

Rational Business Developer



VisualAge Generator to EGL マイグレーション・ガイド

バージョン 7 リリース 5.1

Rational Business Developer



VisualAge Generator to EGL マイグレーション・ガイド

バージョン 7 リリース 5.1

— ご注意 —

本書および本書で紹介する製品をご使用になる前に、571 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。

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

原典： SC31-6830-07
Rational Business Developer
VisualAge Generator to EGL Migration Guide
Version 7 Release 5.1

発行： 日本アイ・ビー・エム株式会社

担当： ナショナル・ランゲージ・サポート

第7版第1刷 2009.1

© Copyright International Business Machines Corporation 2004, 2008.

はじめに

本書は、VisualAge Generator 4.5 から Enterprise Generation Language (EGL) にマイグレーションするユーザーを対象読者としています。

本書の対象読者

本書は、VisualAge Generator 4.5 から Enterprise Generation Language (EGL) へコードをマイグレーションするプログラマーまたはシステム管理者を対象読者としています。

関連情報

関連文書は、以下の 1 つ以上の形式で提供されています。

- 製品 CD-ROM に付属のオンライン・ブック・ファイル (.pdf)。オンラインで表示したり希望のページを印刷したりする場合は、Adobe® Acrobat Reader を使用してください。
- 製品 CD-ROM に付属の HTML ファイル (.htm)。

本書の最新バージョンは、以下の Web サイトからオンライン・ブック・ファイル (.pdf) として入手可能です。

<http://www.ibm.com/software/rational/cafe/community/egl/vagen?view=documents>

目次

はじめに	iii
本書の対象読者	iii
関連情報	iii

第 1 部 マイグレーションの概要 . . . 1

第 1 章 マイグレーションの概要 . . . 3

本書で使用される用語	3
マイグレーションを必要とする EGL の新機能	4
マイグレーションの計画	5
EGL にマイグレーションできるかどうかの判別	10
EGL で使用できない VisualAge Generator 機能	12
用語の違い	13
参照	19

第 2 章 マイグレーション・ツールの考え方 . . . 23

VisualAge Generator to EGL マイグレーション・ツールの概要	24
マイグレーション・ツールの用語	25
ステージ 1 の詳細	26
ステージ 2 の詳細	29
ステージ 3 の詳細	31
単一ファイル・マイグレーションの概要	32
マイグレーションの考慮事項	35
EGL 構文の厳密さ	36
パーツ名が解決される時期と方法	37
共通コードのシナリオ	38
VisualAge Generator から EGL へのマイグレーション・ツールが使用する技法	42
技法の概要	42
エディターとビルド記述子の設定	42
プログラム・プロパティ	43
EGL のビルド・パスと import ステートメント	45
containerContextDependent プロパティ	48
EGL パーツ名制限事項	49
EGL ファイル内でのパーツの配置	50
プログラムを使用したマイグレーション	54
関連パーツを使用したマイグレーション	55
関連パーツを使用しないマイグレーション	56
マイグレーション・セットの処理順序のコントロール	57
ファイルの上書きとマージ	58
一般規則	61
EGL ソース・コードの編成方法の決定	65
製品のコード編成機能に関する違い	65
マイグレーション・ツールが提供する編成機能	68
EGL ソース・コード編成技法の制限とトレードオフ	69

EGL 5.1.2 以降の VAGen マイグレーション・ツールの新機能	72
EGL 6.0 iFix 以降の VAGen マイグレーション・ツールの新機能	73
EGL 6.0.0.1 以降の VAGen マイグレーション・ツールの新機能	73
EGL 6.0.1 以降の VAGen マイグレーション・ツールの新機能	74
EGL 6.0.1.1 以降の VAGen マイグレーション・ツールの新機能	75
EGL 6.0.1.1 ifix003 以降のマイグレーション・ツールの新機能	76
EGL 7.1 以降の VAGen マイグレーション・ツールの新機能	78
マイグレーション・ツールに関する既知の制限事項	78
ステージ 1	78
ステージ 2 および 3	78
構文のマイグレーション	78

第 3 章 未確定状態の処理 . . . 79

データ項目の未確定状態の処理	79
偶数の長さの PACK データ項目	79
共用編集とメッセージ	81
共用データ項目のマッピング編集ルーチン	83
共用データ項目の充てん文字	85
レコードの未確定状態の処理	86
再定義レコード	86
レコード内のレベル 77 項目	87
代替仕様レコード	89
同じレコード名をもつ異なる定義	91
予約語と UI レコード名	92
テーブルの未確定状態の処理	93
予約語とテーブル名	93
マップ・グループとマップの未確定状態の処理	94
予約語と FormGroup 名	94
マップ・グループと FormGroup の要件	95
浮動域と開始位置	96
マップ名とヘルプ・マップ名	97
数値変数フィールド	99
マップ変数フィールドと編集ルーチン	100
マップ・フィールドと数値ハードウェア属性	101
マップ配列と属性	102
名前なしマップ変数フィールド	103
無保護マップ定数	104
row=0, column=0 にあるフィールド	105
プログラムの未確定状態の処理	107
プログラム名と予約語	107
プログラム内の暗黙データ項目	107
関連プログラム・パーツ	108
呼び出しパラメーター・リストに EZEDLPCB が含まれるプログラム	110

マイグレーションに必要な中間変数	111
関数 (入出力ステートメントを含む) の未確定状態の処理	113
マップの DISPLAY 入出力オプション	113
入出力エラー・ルーチン	115
SQL 入出力ステートメント	116
SQL 入出力と必須 SQL 文節の欠落	119
SQL 入出力と Execution time statement build	123
SQL 入出力と !itemColumnName	125
複数の UPDATE 関数または SETUPD 関数を使用する SQL 入出力	126
DL/I I/O および比較値項目	127
その他のステートメントの未確定状態の処理	128
ステートメントの暗黙データ項目	128
ステートメントのレベル 77 項目	129
ステートメントのテーブル参照	129
単一行テーブルをソースとして含む MOVEA	130
代入ステートメント	131
FIND ステートメント	131
RETR ステートメント	132
SET map PAGE ステートメント	133
SET mapItem 属性	134
IN がリテラルかスカラーかの検査	135
SQL 項目とマップ項目が NULL かどうかの検査	136
入出力エラー値 UNQ および DUP	138
入出力エラー値 LOK	141
EZE ワードの未確定状態の処理	142
EZELTERM	142
EZESYS	142
EZEWAIT	144

第 2 部 VisualAge Generator 4.5 on Java から EGL へのマイグレーション 147

第 4 章 ステージ 1 - Java からの抽出 149

VisualAge for Java へのステージ 1 マイグレーション・ツールのインストール	149
マイグレーション・フィーチャーの追加	150
マイグレーション・データベースの作成	151
ステージ 1 設定の実行	151
「プランの作成」ページ	152
「マッピング」ページ	156
「名前変更」ページ	158
「実行」ページ	159
MigPreferences.xml ファイルのサンプル	162
ステージ 1 ツールを実行する前に - ヒント	164
ステージ 1 マイグレーション・ツールのカスタマイズ	164
文字セット情報の指定	166
パフォーマンスの向上	166
ワークスペースの保管	167
ステージ 1 ツールの実行	168

マイグレーション・プランと上位 PLP プロジェクト	169
上位 PLP プロジェクトの作成	170
マイグレーション・プラン・ファイルの手動作成	172

第 3 部 VisualAge Generator 4.5 on Smalltalk から EGL へのマイグレーション 175

第 5 章 ステージ 1 - Smalltalk からの抽出 177

VisualAge Smalltalk へのステージ 1 マイグレーション・ツールのインストール	177
マイグレーション・フィーチャーのロード	178
マイグレーション・データベースの作成	179
ステージ 1 設定の実行	179
「プランの作成」ページ	180
「マッピング」ページ	183
「名前変更」ページ	186
「実行」ページ	187
MigPreferences.xml ファイルのサンプル	189
設定からのファイル名の取得	191
ステージ 1 ツールを実行する前に - ヒント	192
ステージ 1 マイグレーション・ツールのカスタマイズ	192
文字セット情報の指定	193
パフォーマンスの向上	194
イメージの保管	195
ステージ 1 マイグレーション・ツールの実行	195
以前のバージョンのステージ 1 ツールを使用して作成されたマイグレーション・データベースの修復	199
マイグレーション・プランと上位構成マップ	199
上位構成マップの作成	200
マイグレーション・プラン・ファイルの手動作成	202

第 4 部 ステージ 2 および 3 - 共通のマイグレーション・ステップ 205

第 6 章 ステージ 2 - EGL 構文への変換 207

DB2 パフォーマンス情報の設定	207
ワークベンチの設定	207
始動パラメーター	208
必要な EGL 設定	208
推奨設定	210
VAGen マイグレーションの設定	211
その他の推奨設定	217
ステージ 2 VAGen マイグレーション・ファイルの設定	218
ステージ 2 の実行	224
ユーザー・インターフェースからのステージ 2 の実行	224
バッチ・モードでのステージ 2 の実行	225

第 7 章 ステージ 3 - インポート	229
ステージ 3 ツールの実行	229
バッチ・モードでのステージ 3 の実行	233
一時ディレクトリーに書き込まれたマイグレーション・セットの使用	233
第 8 章 単一ファイル・モードでのマイグレーションの実行	235
ユーザー・インターフェースを使用した単一ファイル・マイグレーションの実行	236
バッチ・モードを使用した単一ファイル・マイグレーションの実行	237
第 5 部 マイグレーションの完了	241
第 9 章 マイグレーションの完了	243
ビルド順序の設定	243
設定のエクスポート	244
EGL プロジェクトとパッケージに関するベースラインの保管	245
単一ファイルのマイグレーションを完了するための予備タスク	245
ステージ 1 から 3 までのマイグレーションと単一ファイル・マイグレーションに共通するタスク	246
EGL ソース・コードの検討	246
EGL ビルド記述子パーツの検討	247
EGL リンケージ・オプション・パーツの検討	251
EGL リソース関連パーツの検討	253
テンプレートとして使用するバインド制御パーツの設定	254
プログラム固有のバインド制御パーツの設定	256
リンク・エディット・コマンドの検討	256
ご使用の VGWebTransactions の検討	257
デバッグの準備	259
zSeries に対する EGL サーバー製品のインストール	259
VSE に対する EGL サーバー製品のインストール	261
VAGen 準備テンプレートおよびプロシージャの EGL ビルド・スクリプトへの変換	261
VAGen ランタイム・テンプレートの変換	262
VAGen 予約語ファイルの変換	264
COBOL 生成を行う場合の生成とテスト	264
Java 生成を行う場合の生成とテスト	266
標準の検討	267
ソース・コードの二重メンテナンスの計画	267
VisualAge Generator 互換モードの使用の中止	268
第 6 部 言語と実行時に関する違い	273
第 10 章 言語と実行時に関する違い	275
言語に関する違い	275
実行時の違い	275
一般的な違い	275
SQL サポートに関する違い	277

DL/I サポートに関する違い	280
デバッグに関する違い	282
生成される COBOL に関する違い	284
生成される Java に関する違い	284
ホスト環境とワークステーション環境の違い	285
分散 CICS 環境とネイティブ・ワークステーション環境の違い	286
生成される C++ と生成される Java の違い	289

第 7 部 付録 293

付録 A. 予約語 295

VisualAge Generator マイグレーション・ツールで拡張された予約語	295
EGL 予約語	295
EGL 列挙型ワード	296
SQL 予約語	299
特殊な処理を必要とする SQL 予約語	300
Java 予約語	301

付録 B. VisualAge Generator と EGL の言語要素の関係 303

一般的な構文規則	304
データ項目	305
レコード	313
テーブル	331
マップ・グループ	334
マップ	338
プログラム	356
関数	364
ステートメント	386
EZE ワード	404
プログラム・フローの EZE ワード	404
SQL EZE ワード	405
DL/I EZE ワード	407
日付と時刻の EZE ワード	408
その他のデータの EZE ワード	408
一般的な関数の EZE ワード	411
ストリング EZE ワード	412
数学 EZE ワード	413
ユーザー・インターフェースの EZE ワード	415
オブジェクト・スクリプトの EZE ワード	415
サービス・ルーチン	415
PSB	416
制御パーツ	419
生成オプション・パーツ	421
生成オプション・パーツで使われる変換テーブル名	441
リンケージ・テーブルおよびリソース関連パーツで使用されている変換テーブル名	442
リンケージ・テーブル・パーツ	443
リソース関連パーツ	452
リンク・エディット・パーツ	457
バインド制御パーツ	458
シンボリック・パラメーター	458

他の生成情報	461
準備テンプレートおよびプロシージャ	461
ランタイム・テンプレート	463
他のランタイム情報	466
ランタイム環境変数	466
vgj.properties	468

付録 C. マイグレーション・ツールからのメッセージ 471

VisualAge Generator から EGL マイグレーション・ツールへのメッセージ - ステージ 1	471
ステージ 1 共通メッセージ	471
VisualAge for Java のステージ 1	475
VisualAge Smalltalk のステージ 1	480
VisualAge Generator から EGL マイグレーション・ツールへのメッセージ - ステージ 2	482
VisualAge Generator から EGL マイグレーション・ツールへのメッセージ - ステージ 3	507

付録 D. 「問題」ビューのメッセージ 509

付録 E. 「問題」ビューの IWN.xxx メッセージ 519

.egl ファイルの IWN.VAL メッセージ	519
.eglbld ファイルの IWN.VAL メッセージ	534
JSP の Java メッセージ	537
メッセージの参照情報 - ネーム解決規則および名前の修飾規則	537
VisualAge Generator のネーム解決規則および名前の修飾規則	537
EGL のネーム解決規則および名前の修飾規則	540
ネーム解決規則および名前の修飾規則の違いによる検証メッセージ	542

付録 F. VisualAge Generator に必要な APAR. 545

付録 G. マイグレーション・データベース 547

DB2 マイグレーション・データベースの作成	547
Windows XP 上での DB2 の使用	547
DB2 権限要件	547
マイグレーション・データベースの作成	547

ステージ 1 のマイグレーション・データベースのリセット	549
DB2 を使用したリモート・データベースのカatalog	550
DB2 を使用したリモート・データベースのアンカタログ	551
便利な照会	551
マイグレーション・データベース内のパーツ数の判別	552
EGL ファイル名の検討	553
特定のエラー・メッセージを支援する照会	554
ステージ 2 のマイグレーション・データベースのリセット	555
マイグレーション・データベースのバックアップおよび復元	555

付録 H. マイグレーション・ツールのパフォーマンス 557

プロジェクト、パッケージ、パーツ、およびプログラムの数	558
マイグレーション・セットおよびその他のマイグレーション・オプションの数	559
プロセッサ速度	561
演算部の行数	561
ステージ 1 の新規 Java ワークスペース	562
ディスク・スペース要件	562

付録 I. VisualAge Generator および EGL インターオペラビリティ 565

z/OS CICS における VisualAge Generator および EGL のインターオペラビリティ	565
iSeries における VisualAge Generator および EGL のインターオペラビリティ	565
Web トランザクションでの VisualAge Generator および EGL インターオペラビリティ	568
システム共通プロダクトのインターオペラビリティ	568

特記事項 571

商標	573
----	-----

索引 575

第 1 部 マイグレーションの概要

第 1 章 マイグレーションの概要

Enterprise Generation Language (EGL) コンポーネントを組み込んだ製品は VisualAge® Generator の後継製品です。EGL に移行するには、VisualAge Generator (VAGen) ソース・コードをマイグレーションする必要があります。

本マイグレーション・ガイドでは、マイグレーションの計画、マイグレーション・ツールを使用したソース・コードの変換、およびマイグレーション・ツールの実行後にマイグレーションを完了するために必要なその他の手順について説明します。

本書で使用される用語

EGL は、他の Eclipse ベースの製品と一緒にインストールすることも可能な、IBM® Rational® Business Developer で使用できます。本書では、以下の用語を使用しています。

開発者製品

IBM Rational Business Developer。スタンドアロン製品として使用することも、以下に示すような他の製品とともにインストールすることもあります。

- IBM Rational Application Developer
- IBM Rational Developer for System i®
- IBM Rational Developer for System z®

EGL 開発環境

IBM Rational Business Developer を始動した後に表示されるワークベンチおよび他のウィンドウ。

EGL COBOL ジェネレーター

EGL COBOL 生成サポートを提供する任意の製品または機能。

- System i の場合、EGL COBOL 生成機能は IBM Rational Business Developer に組み込まれています。
- System z の場合、EGL COBOL 生成機能は IBM Rational Business Developer の System z 用の生成フィーチャーに組み込まれています。このフィーチャーをインストールする必要がありますが、IBM Rational Business Developer のライセンスを適用すると、自動的に使用可能になります。
- VSE の場合、EGL COBOL 生成機能は、IBM Rational Business Developer の VSE 用の生成フィーチャーに組み込まれています。このフィーチャーをインストールし、その後 IBM Rational Business Developer Extension for VSE のライセンスを購入して適用することによりこのフィーチャーを使用可能にする必要があります。

EGL ビルド・サーバー

生成済みの COBOL プログラムに EGL ビルド・サーバー・サポートを提供する任意の製品または機能。

- System i の場合、ビルド・サーバーは IBM Rational Business Developer に組み込まれていますが、System i にアップロードしてインストールする必要があります。
- System z の場合、ビルド・サーバーは IBM Rational COBOL Runtime for zSeries® によって提供されます。
- VSE の場合、ビルド・サーバーに相当するものが、VSE 用の生成フィーチャーに組み込まれている準備 JCL テンプレート、および IBM Rational COBOL Runtime for z/VSE™ に組み込まれている準備 JCL プロシージャによって提供されます。

EGL ランタイム・サーバー

生成済みの COBOL プログラムに EGL ランタイム・サポートを提供する任意の製品または機能。

- System i の場合、ランタイム・サーバーは IBM Rational Business Developer に組み込まれていますが、System i にアップロードしてインストールする必要があります。
- System z の場合、ランタイム・サーバーは IBM Rational COBOL Runtime for zSeries によって提供されます。
- VSE の場合、ランタイム・サーバーは IBM Rational COBOL Runtime for z/VSE によって提供されます。

マイグレーションを必要とする EGL の新機能

EGL は、VisualAge Generator からの主要な変更および機能強化が行われています。以下のタイプの変更は、EGL へのマイグレーションに影響する可能性があります。

- VAGen 言語への変更。例えば、新しいデータ型、多次元構造化フィールド配列、動的配列、**case** ステートメント、Web サポートの改良など、数多くの機能拡張を含む。
- コンテンツ・アシスト、パーツ作成用のコード・テンプレート、ほとんどのパーツ型用のテキスト・エディターなど、プログラムの開発に使用するユーザー・インターフェースの変更。
- 生成プロセスや準備プロセスへの変更。例えば、分散プラットフォームの場合に C++ と Java™ を生成する代わりに Java のみを生成し、COBOL 生成の場合に JCL テンプレートを準備する代わりに EGL ビルド・サーバーを使用するなど。
- ランタイム動作への変更。例えば、EGL ランタイム・サーバーの使用など。
- ライブラリー管理への変更。例えば、EGL とのインターフェースに独自のソース・コード・リポジトリを選択する機能など。

VAGen 言語と EGL の違いは広範囲に及びます。以前、システム共通プロダクトまたは VisualAge Generator のバージョンを新バージョンにアップグレードしていたときには、言語の変更は小規模でした。従来のマイグレーション・ツールは、他のパーツと独立して各パーツをマイグレーションできました。一方で、VisualAge Generator から EGL へのマイグレーション・ツールは、これら 2 つの言語の違いが大きいため、他の参照先パーツや関連パーツのコンテキストに従って次の要因を判別し、各パーツをマイグレーションする必要があります。

- 参照先パーツのパーツ型

- 新しい EGL の構文に従って参照側パーツに移動する必要がある情報
- ワークスペース内での参照先パーツのロケーション

このように、1 つのパーツのマイグレーションが他のパーツに依存するという状況を表すために、パーツ間マイグレーション という用語が使用されます。パーツ間マイグレーションは、VAGen 言語から EGL への変換を可能な限り最適に行うために必要です。さらにこのためには、一緒にマイグレーションするパーツのグループを慎重に検討する必要があります。

VisualAge Generator と EGL との相違点、およびパーツ間マイグレーションが必要なことを考慮すると、このマイグレーションはかなりの作業であり、慎重な計画が必要です。

マイグレーションの計画

マイグレーション・プロジェクトを計画する際には、次の作業を検討する必要があります。

- マイグレーションのパイロット・プロジェクトの計画:
 - パイロット・プロジェクトに参加する開発者とシステム・サポート担当者を選出します。
 - パイロット・プロジェクトに使用するソース・コードの小サブセットを選択します。この小サブセットを使用して、環境のセットアップ、およびライブラリー管理の手順とツールを検証します。
 - 修正パッケージ 5 を適用した VisualAge Generator 4.5 にアップグレードします。修正パッケージを入手するには、IBM サポートにご連絡いただくか、VAGen Web サイト (

<http://www.ibm.com/software/awdtools/visgen/support>) にアクセスしてください。

「Download」セクションにあるリンクをたどってください。また、特定の状況に応じて必要になる可能性があるその他の VAGen APAR については、545 ページの『付録 F. VisualAge Generator に必要な APAR』を参照してください。

- DB2® がまだ使用可能になっていない場合はインストールまたはアップグレードします。DB2 は、マイグレーション・データベースに使用する必要があります。マイグレーション・ツールを使用するには、修正パッケージ 15 を適用した DB2 バージョン 8.1 または修正パッケージ 8 を適用した DB2 バージョン 8.2 が必要です。これら 2 つの修正パッケージ・レベルは同等です。DB2 バージョン 9.x は、Smalltalk でのステージ 1、およびステージ 2 および 3 でサポートされています。DB2 バージョン 9.x は Java のステージ 1 ではサポートされません。
- 使用を予定している開発者製品の前提条件を確認します。また、次の例に示すような、ランタイム環境の前提条件を確認します。
 - z/OS® 環境用の COBOL の生成を予定している場合は、IBM Rational COBOL Runtime for zSeries の前提条件を確認します。
 - UNIX® System Services (USS) 環境用の Java の生成と、その環境でのビルドを予定している場合は、IBM Rational COBOL Runtime for zSeries の前提条件を確認します。

- z/VSE 環境用の COBOL の生成を予定している場合は、IBM Rational COBOL Runtime for z/VSE の前提条件を確認します。
- iSeries® 環境での生成を予定している場合はご使用の開発者製品のランタイム・コンポーネントの前提条件を必ず確認してください。
- ワークステーション環境用の Java の生成を予定している場合は、開発者製品の前提条件を確認します。
- 次の例に示すように、パイロット・プロジェクトのスコープに関する主要な決定を行います。
 - 実際のマイグレーション中に VAGen の開発と保守を凍結できるかどうかを判断します。この手法では実動レベルのソース・コードのみをマイグレーションできます。VAGen の開発と保守を凍結できない場合は、パイロット・プロジェクトに次のタスクを組み込んでください。
 - 処理中作業のソースを VisualAge Generator から EGL にマイグレーションするための手順を開発し、テストします。
 - 共通 (共用) パーツの二重保守のための手順を開発し、テストします。
 - バックエンド・ソース・コード・リポジトリを選択します。
- タスク・リスト、リソース割り当て、およびパイロット・プロジェクトのスケジュールを作成します。
- 以下の分野でパイロット・チームの教育を実施します。
 - Developer 製品環境
 - EGL 言語
 - VisualAge Generator to EGL マイグレーション・ツール
 - 新規のソース・コード・リポジトリ
- 次のタスクを実行するためのパイロット・プロジェクト計画を実施します。
 - パイロット・チーム用の開発者製品をインストールします。インストール先のマシンの地域設定値が、VAGen プログラムの開発に使用した設定値と同じであることを確認してください。以下に、特別な要件の下でのインストールの例を示します。
 - ドイツ語マシン上で VAGen プログラムを開発した場合は、ドイツ語マシンに開発者製品をインストールする必要があります。これにより、小数点としてコンマが使用され、ドイツ語のウムラウト文字が正しくマイグレーションされるようになります。
 - 中国語マシン上で VAGen プログラムを開発した場合は、同じコード・ページを使用する中国語マシンに開発者製品をインストールする必要があります。これにより、DBCS 文字が正しくマイグレーションされるようになります。
 - EGL でのソース・コードの編成方法を決定します。この編成を同等の VAGen の編成にマップします。考慮事項については、65 ページの『EGL ソース・コードの編成方法の決定』を参照してください。
 - パイロットのコード・セットに対して VAGen マイグレーション・ツールを実行します。マイグレーション・ツールの詳細については、次のセクションを参照してください。
 - 23 ページの『第 2 章 マイグレーション・ツールの考え方』

- 147 ページの『第 2 部 VisualAge Generator 4.5 on Java から EGL へのマイグレーション』
- 175 ページの『第 3 部 VisualAge Generator 4.5 on Smalltalk から EGL へのマイグレーション』
- 205 ページの『第 4 部 ステージ 2 および 3 - 共通のマイグレーション・ステップ』
- 241 ページの『第 5 部 マイグレーションの完了』
- EGL 開発環境で次のタスクを実行して、ソース・コードをテストします。
 - EGL デバッグ機能を使用するために必要な接続を計画およびインストールします。インタラクティブ・テスト機能 (ITF) を使用する VAGen プログラムをテストするときに以下のいずれかの機能を使用する場合は、同等の EGL デバッグ機能を実現する方法を計画する必要があります。
 - ITF から呼び出す必要がある EGL 以外のプログラム。
 - DB2 データベースへのアクセス。
 - DL/I データベースへのアクセス。
 - VSAM ファイルへのアクセス。
 - デバッグ用の EGL ビルド・パーツを作成します。これにはデバッグのために必要なビルド記述子オプション、リンケージ・オプション、およびリソース関連パーツが含まれます。
 - EGL デバッグ機能を使用してソース・コードをテストします。ホスト環境に対する各タイプの接続を必ずテストしてください。
- 以下のタスクを実行して、ライブラリー管理プロセスを作成します。
 - ソース・コード・リポジトリを選択してインストールします。
 - 開発者のワークステーションからソース・コード・リポジトリへのアクセスを提供します。
 - 企業標準と選択したソース・コード・リポジトリが連動する変更管理手順を定義します。
 - 変更管理手順に必要なすべてのツールを開発します。例えば、以下のプロセスを支援するツールが必要になる場合があります。
 - チェックインおよびチェックアウト手順
 - バージョン管理手順
 - バッチ生成を使用する場合は、ソース・コード・リポジトリからソース・コードを取得し、ワークスペースまたはディレクトリ構造をロードします。
- iSeries COBOL ターゲット環境で以下のタスクを実行します。
 - 「*Rational Business Developer EGL Server Guide for IBM i*」に記載されている指示に従います。
- z/OS COBOL ターゲット環境で以下のタスクを実行します。
 - TCP/IP をインストールして使用可能にします。TCP/IP は COBOL 生成の出力を z/OS ホストに転送する唯一の方法です。
 - IBM Rational COBOL Runtime for zSeries の前提条件 (COBOL コンパイラおよびランタイム環境に対するすべての変更を含む) をインストールします。

- IBM Rational COBOL Runtime for zSeries (最新の PTF を含む) をインストールします。
- COBOL 生成の出力および EGL ビルド・サーバーから受け取る結果を格納するための新しいライブラリーのセットを作成します。
- CICS® または IMS™ を使用する場合は、EGL で生成した COBOL をテストするための新しい領域を作成します。この手法を使用すると、VAGen で生成したコードと EGL で生成したコードを誤って混合することを防止でき、パイロット・プロジェクトを実行しながら VAGen コードの管理を続行できます。
- EGL ランタイム・サーバーをカスタマイズします (すべてのランタイム環境に対するカスタマイズ検証プログラムの実行を含む)。
- EGL ビルド・サーバーおよび擬似 JCL ビルド・スクリプトをカスタマイズします。詳しくは、261 ページの『VAGen 準備テンプレートおよびプロシーチャーの EGL ビルド・スクリプトへの変換』を参照してください。
- VSE COBOL ターゲット環境で以下のタスクを実行します。
 - FTP をインストールして使用可能にします。FTP は COBOL 生成の出力を VSE ホストに転送する唯一の方法です。
 - IBM Rational COBOL Runtime for z/VSE の前提条件 (COBOL コンパイラーおよびランタイム環境に対するすべての変更を含む) をインストールします。
 - IBM Rational COBOL Runtime for z/VSE (最新の PTF を含む) をインストールします。
 - COBOL 生成の出力および EGL ビルド・サーバーから受け取る結果を格納するための新しいライブラリーのセットを作成します。
 - CICS を使用する場合は、EGL で生成した COBOL をテストするための新しい領域を作成します。この手法を使用すると、VAGen で生成したコードと EGL で生成したコードを誤って混合することを防止でき、パイロット・プロジェクトを実行しながら VAGen コードの管理を続行できます。
 - EGL ランタイム・サーバーをカスタマイズします (すべてのランタイム環境に対するカスタマイズ検証プログラムの実行を含む)。
 - 準備 JCL テンプレートおよび準備 JCL プロシーチャーをカスタマイズします。詳しくは、19 ページの『参照』にリストされている VSE 資料を参照してください。
- Java ターゲット環境で以下のタスクを実行します。
 - プラットフォームを変更する場合 (例えば、Windows® CICS から Windows ネイティブへ) は、ランタイム・プラットフォームの相違点を検討します。これによるコード変更を行います。使用するオリジナルおよび新規のランタイム・プラットフォームに応じて、275 ページの『第 10 章 言語と実行時に関する違い』の適切なセクションを参照し、プラットフォームの相違点を確認してください。C++ の生成から Java の生成に変更する場合も、同じ章を参照してください。
 - 現在 ODBC サポートを使用している場合は、ベンダーから JDBC サポートを取得します。
- 以下のタスクを実行して、プログラムを生成し、準備します。

- ビルド・パーツ (ビルド記述子、リンケージ・オプション、リソース関連、リンク・エディット、バインド制御パーツ) を確認し、修正します。
VAGen マイグレーション・ツールによって処理できない変更の詳細については、ランタイム環境で使われるビルド・パーツに応じて、243 ページの『第 9 章 マイグレーションの完了』内の適当なセクションを参照してください。
- VAGen 予約語を変更した場合は、EGL 予約語ファイルを作成します。
- 必要により、EGL バッチ生成サーバー・マシンをビルドします。これには、ソース・コード・リポジトリを使用し、また生成に必要なすべてのパーツが入ったディレクトリーをロードするツールを作成する必要があります。
- 特定のタイプのプログラムをマイグレーションする場合は、VAGen プログラムと EGL プログラムの相互運用性を提供するために、EGL プログラムを生成するか、VAGen プログラムを再リンクする必要があります。詳しくは、565 ページの『付録 I. VisualAge Generator および EGL インターオペラビリティ』を参照してください。
- 以下の手順を使用してテストします。
 - 少なくとも生成したプログラムの代表例をテストして、VisualAge Generator と EGL の間のランタイムの相違点を確実に理解します。相違点のリストについては、275 ページの『実行時の違い』を参照してください。
 - EGL ソース・コードに行う可能性のある標準的な変更を使用して、ライブラリー管理手順とツールをテストします。それぞれのターゲット環境ごとに、共通のコード、フォーム、DataTable、およびプログラムを変更する手順を必ずテストしてください。また、それぞれのターゲット環境ごとに、共通のコード、フォーム、DataTable、およびプログラムを追加する手順もテストしてください。
 - 典型的な変更を使用してパイロット変更サイクルを実行し、計画したライブラリー管理プロセスが複数の開発者にとって受け入れ可能であることを確認してください。
 - ソース・コード・リポジトリのバックアップとリカバリーの手順を計画し、テストします。
- パイロット・プロジェクトの結果に基づいて、ライブラリー管理手順およびツールを改善します。
- 以下の事項を含むパイロット・プロジェクトでの調査結果を文書化します。
 - 必要なコード変更 (特に、ターゲット環境を変更する場合)。
 - 開発者がすべての個人用ビルド記述子パーツに行う必要のある変更。
 - パイロット・プロジェクト中に発生した問題に基づいた、開発者に特に役立つマイグレーション・ガイドのセクションの参照。
 - マイグレーション後にユーザーが認識すると考えられるランタイム動作の変更。
 - 最終的なライブラリー管理および変更制御プロセス。
- パイロット・プロジェクトで分かったことに基づいて、マイグレーションを完了するための計画の作成
- 以下のトピックについて残りの開発者への教育を実施します。

- Developer 製品環境
- EGL 言語
- EGL のソース・コード編成 (コードを EGL プロジェクト、パッケージ、およびファイルへ構造化する方法を含む)
- 新規のソース・コード・リポジトリ
- 新規のライブラリー管理プロセス
- 新規の生成プロセス
- 必要に応じて、開発の最初の数週間に渡る指導

EGL にマイグレーションできるかどうかの判別

EGL は Rational Developer 製品の戦略的コンポーネントであり、VisualAge Generator のお客様はこのコンポーネントにマイグレーションする必要があります。EGL サポートは、VisualAge Generator Developer 4.5 がサポートするすべての機能とプラットフォームを完全に置き換えるものではありません。ターゲット環境および VisualAge Generator で開発したプログラムのタイプによっては、他の代替製品を検討しなければならない場合があります。

EGL ベース製品は、以下の VisualAge Generator ターゲット環境に対する Java 生成をサポートします。

- Unix システム・サービス (VAGen OS/390® ターゲット環境)
- iSeries (VAGen OS/400® ターゲット環境)
- Windows ネイティブ
- Intel® プラットフォーム上の Linux®
- AIX® ネイティブ
- HP-UX
- Solaris

EGL ベース製品は、次の VisualAge Generator ターゲット環境に対する COBOL 生成もサポートします。

- iSeries (VAGen OS/400 ターゲット環境)

IBM Rational Business Developer の System z 用の生成フィーチャーは、以下の VisualAge Generator ターゲット環境に対する COBOL 生成機能をサポートします。

- IMS BMP
- IMS/VS
- z/OS パッチ (VAGen MVS™ パッチ・ターゲット環境)
- z/OS CICS (VAGen MVS CICS ターゲット環境)

IBM Rational Business Developer の VSE 用の生成フィーチャーは、以下の VisualAge Generator ターゲット環境に対する COBOL 生成機能をサポートします。

- VSE パッチ
- VSE CICS

注: VisualAge Generator は一部のプラットフォーム上で Java と C++ を生成しますが、EGL は Java のみを生成します。

これらのサポート対象環境のその他の考慮事項については、以下の情報を参照してください。

- EGL へのマイグレーションに関する特別な考慮事項 - ファイルおよびデータベース・アクセス (11 ページの表 1)
- EGL へのマイグレーションに関する特別な考慮事項 - ユーザー・インターフェース、12 ページの表 2
- 12 ページの『EGL で使用できない VisualAge Generator 機能』

次の表に、サポート環境に関する特別な考慮事項をリストします。

表 1. EGL へのマイグレーションに関する特別な考慮事項 - ファイルおよびデータベース・アクセス

VAGen ファイルおよびデータベース・アクセス	特別な考慮事項
SQL	EGL でサポートされています。
シリアル、索引付き、および相対レコード	EGL でサポートされています。
メッセージ・キュー・レコード	EGL でサポートされています。
DL/I	z/OS および VSE 環境用の COBOL 生成は EGL でサポートされています。また、データベースが IMS/VS 上にある場合は、デバッグもサポートされています。データベースが CICS または VSE 上にある場合は、デバッグはサポートされませんが、EGL の資料で変更について確認してください。
GSAM	EGL でサポートされています。
IMS メッセージ・キュー	EGL でサポートされています。
Btrieve	EGL ではサポートされていません。
ローカル VSAM	次の場合にサポートされます。 <ul style="list-style-type: none"> • AIX 環境での Java 生成 • COBOL 生成 デバッグ、または他の環境での Java 生成の場合はサポートされません。
リモート VSAM	次の場合にサポートされます。 <ul style="list-style-type: none"> • リモート・ファイルが z/OS 上にある場合のデバッグ。 • リモート・ファイルが z/OS 上にある場合の、Windows 用の Java 生成。 • z/OS および VSE における CICS 用の COBOL 生成。

表 2. EGL へのマイグレーションに関する特別な考慮事項 - ユーザー・インターフェース

VAGen ユーザー・インターフェース	特殊な考慮事項
テキスト・ユーザー・インターフェース (印刷を含む)	COBOL 生成と Java 生成のどちらの場合も、EGL でサポートされます。
Web トランザクションおよびユーザー・インターフェース (UI) レコード	COBOL 生成と Java 生成のどちらの場合も、環境に応じて EGL でサポートされます。
VAGen Java ラッパーを使用する JSP と Java サブレット	<ul style="list-style-type: none"> 製品に付属の情報を使用して、JSP と Java サブレットを新規の開発者製品にマイグレーションできます。 本書「VAGen マイグレーション・ガイド」を使用して、VAGen サーバー・プログラムを EGL にマイグレーションできます。EGL を使用して、Java ラッパーを生成できます。
フリー・フォーム域で VAGen パーツを使用せず、VAGen Java ラッパーを使用する Java GUI アプリケーションまたはアプレット	<ul style="list-style-type: none"> 製品に付属の VisualAge for Java から Java コードをマイグレーションするための情報を使用して、Java アプリケーションまたはアプレットを新規の開発者製品にマイグレーションできます。 本書「VAGen マイグレーション・ガイド」を使用して、VAGen サーバー・プログラムを EGL にマイグレーションできます。EGL を使用して、Java ラッパーを生成できます。
フリー・フォーム域で VAGen パーツを使用する Java GUI アプリケーションまたはアプレット	サポートされていません。
Smalltalk GUI ビューまたはビジュアル・パーツ	非サポート。VAGen パーツのビューは、Java ベース・ソリューションにマイグレーションする必要があります。EGL には、Smalltalk ベースのソリューションはありません。

EGL で使用できない VisualAge Generator 機能

表 1 および 2 にリストした特別な考慮事項に加え、次のリストのいずれかの機能が必要な場合は、マイグレーションを現時点で行うことと将来的に行うことから受ける影響を評価する必要があります。

- Java または Smalltalk GUI のサポート。
- 以下のような特定のランタイム環境。
 - MVS/TSO。z/OS CICS の検討をお勧めします。
 - VM および VM バッチ。z/OS CICS および z/OS バッチの検討をお勧めします。
 - OS/2® および CICS for OS/2。Windows 用に Java 生成を使用することの検討をお勧めします。
 - 分散プラットフォーム (CICS for Windows、CICS for AIX など) 上の TX シリーズ。同等のネイティブ・ランタイム環境 (Windows、AIX など) 用に Java 生成を使用することの検討をお勧めします。

ただし、Java 生成の使用を予定している場合は、ネイティブ・ランタイム環境に変換できない CICS 固有の機能を使用しているかどうかを判別してください。相違点の詳細については、286 ページの『分散 CICS 環境とネイティブ・ワークステーション環境の違い』を参照してください。

- 生成時に作成されるプログラムのリストなど、特殊なエディターおよびリスト
- 以下のような機能を含む特殊な機能。
 - 選択したパーツのセットから参照を検索し、その参照リストをプログラムおよびプログラムに関連するパーツに限定する機能。
 - パーツ型別またはサブタイプ別によるパーツのフィルター操作。EGL には検索機能が用意されているので、特定のパーツ型またはサブタイプを検索できます。
- 以下のような機能を含む特殊なデバッグ・サポート。
 - データベースが z/OS CICS または VSE CICS 上にある場合は、DL/I データベース I/O。
 - 相対ファイルの索引付け (そのファイルが z/OS 上のリモート VSAM ファイル以外にある場合)。
 - IMS 環境内の、静的リンケージ (PL/I など) を必要とし、IMS または DL/I 呼び出しを実行するプログラムへの呼び出し。静的リンケージを必要とするプログラムへの呼び出しは、呼び出し先プログラム内に IMS または DL/I 呼び出しがなければ許可されます。
 - IMS 環境内の、CBLTDLI または AIBTDLI 以外のもの (PLITDLI または ASMTDLI) を使用して DL/I 呼び出しを実行するプログラムへの呼び出し。
- VisualAge Generator テンプレート。

用語の違い

VisualAge Generator Developer on Java、VisualAge Generator Developer on Smalltalk、および EGL は、すべて異なる用語を使用します。以下の表に VAGen 用語と EGL 用語の関連を示します。

表 3. コード編成用語の相違点

VisualAge Generator on Java	VisualAge Generator on Smalltalk	エンタープライズ開発言語 (EGL)
ワークスペース	イメージ	ワークスペース
プロジェクト	構成マップ	EGL プロジェクト
パッケージ	アプリケーション	1 つ以上の EGL ファイルを含む EGL ソース・フォルダーおよび EGL パッケージ
(同等の概念なし)	(同等の概念なし)	ファイル (一般に、複数のファイルに分割される Java パッケージまたは Smalltalk アプリケーション)。EGL ファイルには、1 つ以上のパーツ型の EGL パーツ (複数可) が格納されます。
クラスまたはタイプ	クラス	EGL パーツ型

表 3. コード編成用語の相違点 (続き)

VisualAge Generator on Java	VisualAge Generator on Smalltalk	エンタープライズ開発言語 (EGL)
メソッドまたはメンバー	メソッド	(同等の概念なし)
VAGen パーツ	VAGen パーツ	ファイル内の EGL パーツ

表 4. VAGen のパーツと概念に関する用語の違い

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
共用データ項目	共用データ項目	DataItem パーツを使用した型定義
非共用データ項目	非共用データ項目	プリミティブ・データ型を使用した型定義
データ項目パーツ	データ項目パーツ	DataItem パーツ
レコード・パーツ	レコード・パーツ	レコード・パーツ 注: マイグレーション制御ツールは、VAGen の振る舞いを保持するために、すべての VAGen のレコード定義を EGL の構造化レコードに変換します。
PSB パーツ	PSB パーツ	PSBRecord ステレオタイプを持つレコード・パーツ。
ユーザー・インターフェース (UI) レコード	ユーザー・インターフェース (UI) レコード	VGUIRecord ステレオタイプを持つレコード・パーツ。
構造項目 (レコード内のフィールドの構造)	構造項目 (レコード内のフィールドの構造)	構造化フィールド
配列 (レコードまたはマップに複数回出現する項目)	配列 (レコードまたはマップに複数回出現する項目)	構造化フィールド配列
テーブル・パーツ	テーブル・パーツ	DataTable パーツ
マップ・グループ・パーツ	マップ・グループ・パーツ	FormGroup
マップ・パーツ: • 表示マップ • プリンター・マップ	マップ・パーツ: • 表示マップ • プリンター・マップ	書式: • テキスト書式 • printForm
演算部	演算部	スタンドアロン演算部。 注: マイグレーション・ツールは、すべての VAGen 演算部を EGL スタンドアロン演算部に変換します。
入出力オプションと入出力オブジェクト	入出力オプションと入出力オブジェクト	EGL 入出力ステートメント

表 4. VAGen のパーツと概念に関する用語の違い (続き)

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
Java アプリケーションまたはアプレット (GUI)	Smalltalk ビューまたはビジュアル・パーツ (GUI)	<ul style="list-style-type: none"> • Smalltalk ビューおよびビジュアル・パーツはサポートされていません。 • Java アプリケーションとアプレットは、フリー・フォーム域で VAGen パーツを使用していない場合 にサポートされます。フリー・フォーム域で VAGen パーツを使用していた場合、Java アプリケーションまたはアプレットはサポートされません。
生成オプション・パーツ	生成オプション・パーツ	ビルド記述子パーツ
生成オプション	生成オプション	ビルド記述子オプション
リンケージ・テーブル・パーツ	リンケージ・テーブル・パーツ	リンケージ・オプション・パーツ

表 5. VAGen と IDE Windows の用語の違い

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
<p>ログ</p> <ul style="list-style-type: none"> • エラー・メッセージを表示します • ログとワークベンチの両方 を閉じた場合のみ、製品は閉じます • 製品を閉じるときに、ワークスペースは常に 保管されます 	<p>システム・トランスクリプト</p> <ul style="list-style-type: none"> • エラー・メッセージを表示します • システム・トランスクリプトまたは VisualAge Organizer のどちらかを閉じると、製品が閉じます • 製品を閉じるときに、イメージはオプションで 保管されます 	<p>コンソール</p> <ul style="list-style-type: none"> • メッセージを表示します。 <p>「問題」ビュー</p> <ul style="list-style-type: none"> • メッセージを表示します (特に構文の検証に関連するメッセージ)。 • 製品を閉じるときに、ワークスペースは常に 保管されます。

表 5. VAGen と IDE Windows の用語の違い (続き)

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
<p>ワークベンチ</p> <ul style="list-style-type: none"> ワークスペース内のプロジェクトとパッケージを表示します。 	<p>VisualAge Organizer</p> <ul style="list-style-type: none"> イメージ内のアプリケーションを表示します。 	<p>EGL および Web パースペクティブ:</p> <ul style="list-style-type: none"> 「ナビゲーター」ビューと「プロジェクト・エクスプローラー」ビューに、ワークスペース内のプロジェクト、ソース・フォルダー、パッケージ、およびファイルが表示されます。 エラー・マーカは「プロジェクト・エクスプローラー」ビューには表示されますが、「ナビゲーター」ビューには表示されません。
スクラップブック	ワークスペース	スクラップブック・ページ・エディター
リポジトリ・エクスプローラー	アプリケーション・エディション・ブラウザー	リポジトリの使用を決定した場合、リポジトリに同等の概念がある場合があります。
<p>VAGen パーツ・ブラウザー</p> <ul style="list-style-type: none"> 3 つのペインに、パッケージ、パーツ型、および VAGen パーツが表示されます。 ブラウザーにはフィルター操作とソートが組み込まれています。 	<p>VAGen パーツ・ブラウザー</p> <ul style="list-style-type: none"> 3 つのペインに、アプリケーション、パーツ型、および VAGen パーツが表示されます。 ブラウザーにはフィルター操作とソートが組み込まれています。 	<p>EGL および Web パースペクティブ:</p> <ul style="list-style-type: none"> 「ナビゲーター」ビューと「プロジェクト・エクスプローラー」ビューに、ワークスペース内のプロジェクト、ソース・フォルダー、パッケージ、およびファイルが表示されます。 「アウトライン」ビューに、ファイル内のパーツが表示されます。 「EGL パーツ・リスト」ビューではフィルター操作とソートを実行できます。
VAGen オプション	VAGen 設定	EGL 設定
VAJava オプション	VASmalltalk 設定	その他の製品設定
「参照 (References)」ツールによる、特定のパーツ名またはテキスト・ストリングを使用するパーツの検索	「参照 (References)」ツールによる、特定のパーツ名またはテキスト・ストリングを使用するパーツの検索	EGL 検索またはファイル検索
「関連 (Associates)」ツールによる、特定のパーツが参照するパーツすべての検索	「関連 (Associates)」ツールによる、特定のパーツが参照するパーツすべての検索	EGL パーツ参照

表 6. VAGen のワークスペース管理に関する用語の違い

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
リポジトリ	ライブラリー	なし。使用する製品によって、CVS および Clear Case LT が提供されます。独自のリポジトリ管理システムを選択できます。
追加/削除	ロード/アンロード	リポジトリの使用を決定した場合、リポジトリに同等の概念がある場合があります。
置換	別のエディションのロード	ローカル・ヒストリーとの置換 注: 使用を決定したリポジトリに追加機能が存在することがあります。
次と比較	変更点のブラウズ	ローカル・ヒストリーとの比較 注: 使用を決定したリポジトリに追加機能が存在することがあります。

表 7. VAGen のリポジトリ管理に関する用語の違い

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
管理者	ライブラリー・スーパーバイザー	リポジトリの使用を決定した場合、リポジトリに同等の概念がある場合があります。
リポジトリ管理: • パージ/復元 • 圧縮	ライブラリー管理: • パージ/サルベージ • クローン	リポジトリの使用を決定した場合、リポジトリに同等の概念がある場合があります。

表 8. VAGen ソース・コード管理の用語の相違点

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
所有権: • プロジェクト所有者 • パッケージ所有者 • クラス所有者	所有権: • 構成マップ管理者 • アプリケーション管理者 • クラス所有者	リポジトリの使用を決定した場合、リポジトリに同等の概念があります。
バージョンとリリース	バージョンとリリース	リポジトリの使用を決定した場合、リポジトリに同等の概念があります。

表 8. VAGen ソース・コード管理の用語の相違点 (続き)

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
<p>プロジェクト:</p> <ol style="list-style-type: none"> 1. プロジェクトは必須。 2. VAGen プロジェクト・リスト・パーツが、プロジェクト間の関係を指定します。 3. パッケージ所有者は、常にパッケージをプロジェクトにリリースできます。 	<p>構成マップ:</p> <ol style="list-style-type: none"> 1. 使用はオプション。 2. 必須のマップが、構成マップ間の関係を指定します。 3. オプションで、アプリケーションのリリースを委任したり、リリースを構成マップ・マネージャーのみに制限したりすることができます。 	<p>プロジェクト:</p> <ol style="list-style-type: none"> 1. プロジェクトは必須。 2. プロジェクトの EGL ビルド・パス・プロパティ。ただし、これによってプロジェクトがまとめて自動的にワークスペースにロードされることはありません。 3. リポジトリに存在しない限り、同等の概念はありません。
<p>パッケージ:</p> <ol style="list-style-type: none"> 1. 同等の概念なし 2. 同等の概念なし 3. 同等の概念なし 4. グループ・メンバー 5. プロジェクトのバージョン管理により、組み込まれたパッケージのバージョン管理が自動的に行われます。 	<p>アプリケーション:</p> <ol style="list-style-type: none"> 1. 前提アプリケーション 2. サブアプリケーション 3. 特権 4. グループ・メンバー 5. 構成マップのバージョン管理を行う前に、アプリケーションのバージョン管理を行う必要があります。 	<p>フォルダーまたはパッケージ:</p> <ul style="list-style-type: none"> • リポジトリの使用を決定した場合、リポジトリに同等の概念がある場合があります。
<p>クラスまたはタイプ:</p> <ul style="list-style-type: none"> • パッケージまたはプロジェクトのバージョン管理により、組み込まれたクラスのバージョン管理が自動的に行われます。 	<p>クラス:</p> <ul style="list-style-type: none"> • アプリケーションのバージョン管理を行う前に、クラスのバージョン管理とリリースを行う必要があります。 	<p>EGL パーツ型</p> <ul style="list-style-type: none"> • EGL に同等の概念はありません。

表 8. VAGen ソース・コード管理の用語の相違点 (続き)

VisualAge Generator on Java	VisualAge Generator on Smalltalk	EGL
<p>VAGen パーツ:</p> <ul style="list-style-type: none"> それぞれのパーツごとに日時スタンプが付いています。 重複するパーツ名を含むパッケージをワークスペースに追加できます。 重複するパーツを見つけるための重複パーツ・ツールが存在します。 	<p>VAGen パーツ:</p> <ul style="list-style-type: none"> それぞれのパーツごとに日時スタンプが付いています。 重複するパーツ名を含むアプリケーションは、イメージにロードできません。 	<p>EGL パーツ:</p> <ul style="list-style-type: none"> パーツは EGL ファイル内にあり、EGL ファイルのみに日時スタンプが付いています。 ワークスペース内に重複するパーツがあっても構いません。EGL は、プロジェクトの EGL ビルド・パス、ファイルの import ステートメント、および containerContextDependent プロパティの組み合わせを使用して、パーツ名への参照を解決するために検索されるネーム・スペースを判別します。パーツ名は、ネーム・スペースの中で固有であることが必要です。プロジェクトの EGL ビルド・パスにより、パーツ名を検索する際に対象になる他のプロジェクトが限定されます。ファイルの import ステートメントにより、パーツ名を検索する際に対象になる、EGL ビルド・パスに含まれる他のパッケージまたはパーツが限定されます。レコードまたは関数の containerContextDependent プロパティは、EGL がレコードまたは関数を含むファイルではなく、プログラムを含むファイルの EGL ビルド・パスと import ステートメントを使用することを指定します。

参照

本マイグレーション・ガイドのほかに、次の資料で追加情報や最新情報を確認してください。

- VisualAge Generator に関する Web サイトとニュース・グループ。Web サイトは以下のアドレスにあります。

<http://www.ibm.com/software/awdtools/visgen/>

- EGL に関する Web サイトとフォーラム。Web サイトは以下のアドレスにあります。

<http://www.ibm.com/software/rational/cafe/community/egl>

- 使用している製品に関する Web サイトとフォーラム。

次の資料にも本マイグレーション・ガイドの範囲外の詳細情報が記載されています。

- EGL のオンライン・ヘルプ・システム。
- オンライン・ヘルプ・システムと同じ情報が記載されている次のボリューム。
 - 「*EGL 言語リファレンス*」は、EGL 言語について説明しています。
 - 「*EGL 生成ガイド*」は、Java 生成と COBOL 生成の両方の生成プロセスについて説明しています。この資料には、EGL で生成した Java コードのデプロイに関する情報も記載されています。
 - 「*EGL プログラマー・ガイド*」は、さまざまなタイプの EGL アプリケーションの開発方法について説明します。
- EGL サーバー・ガイドの中には、ビルド・サーバーおよびビルド・スクリプトのセットアップ方法を説明するほか、ランタイム環境を作成するためのその他の参照資料となるものもあります。使用する COBOL ランタイム環境に応じて、次のいずれかの資料を選択してください。
 - *IBM Rational COBOL Runtime for zSeries ガイド*、バージョン 6.0.1 (SC88-4055)
 - *Rational Business Developer EGL Server Guide for IBM i Version 7.5* (SC31-6841)
- VSE については、以下の資料で、VSE と z/OS サポートの違い、および準備プロセスとランタイム・サーバー・サポートについて説明されています。
 - *Program Directory for Rational COBOL Runtime for z/VSE* (GI10-8803-00)
 - *Rational Business Developer V7.5 Generation for z/VSE feature Reference Manual* (SC19-2539-00)

次のホワイト・ペーパーもマイグレーションに役立ちます。

- Java での VisualAge Generator からのマイグレーションの場合:
 - *How to Consolidate Projects and Packages during Stage 1 of the VisualAge Generator on Java to Enterprise Generation Language Migration Tool*
 - ステージ 1 で使用される EGL ファイル・ロケーション・アルゴリズムの変更方法について詳しくは、164 ページの『ステージ 1 マイグレーション・ツールのカスタマイズ』を参照してください。
- Smalltalk での VisualAge Generator からのマイグレーションの場合:
 - *How to Consolidate Projects and Packages during Stage 1 of the VisualAge Generator on Smalltalk to Enterprise Generation Language Migration Tool*
 - ステージ 1 で使用される EGL ファイル・ロケーション・アルゴリズムの変更方法について詳しくは、192 ページの『ステージ 1 マイグレーション・ツールのカスタマイズ』を参照してください。
- Java での VisualAge Generator または Smalltalk での VisualAge Generator のマイグレーションの場合:
 - *How to Create Records for Implicit Items when Migrating from VisualAge Generator to Enterprise Generation Language*
 - *Using the Rename User Exit in the VisualAge Generator to Enterprise Generation Language Migration Tool*

- *How to Modify the Package Name in the JSP when Migrating Web Transactions from VisualAge Generator to Enterprise Generation Language*
- システム共通プロダクトまたは VisualGen® (バージョン 2.2 およびそれ以前) からのマイグレーションの場合:
 - *Migrating from Cross System Product Version 4.1 to Enterprise Generation Language Version 7.5.1*。このホワイト・ペーパーはシステム共通プロダクトバージョン 4.1 に固有ですが、システム共通プロダクトの古いバージョンからのマイグレーションと、VisualGen (バージョン 2.2 およびそれ以前のバージョン) からのマイグレーションとの違いについて説明した付録が含まれています。

これらのホワイト・ペーパーはすべて、次の Web サイトの EGL Cafe の「VAGen Migration」ハブから入手できます。

<http://www.ibm.com/software/rational/cafe/community/egl/vagen?view=documents>

第 2 章 マイグレーション・ツールの考え方

VisualAge Generator to EGL マイグレーション・ツールは、実際には一連の複数のツールからなります。この章では、これらのツールの概要を説明し、ツールが使用している技法について説明します。

VisualAge Generator to EGL マイグレーション・ツールは、主に次のことを目的として設計されています。

- プログラムの振る舞いを、VisualAge Generator から EGL へのマイグレーション後も保持する。
- Java プロジェクトとパッケージの構造を、VisualAge Generator から EGL へのマイグレーション後も保持する (該当する場合)。
- Smalltalk 構成マップとアプリケーションの構造が、VisualAge Generator から EGL へのマイグレーション後も保持されます (該当する場合)。
- サブシステムのインクリメンタル・マイグレーションを、1 つのサブシステムごとに実行する。
- サブシステムの複数バージョンのマイグレーションを実行する。

VisualAge Generator to EGL マイグレーション・ツールの設計は、次のような副次的な目的も考慮しています。

- 可能な限りバッチ・モード処理を使用し、重要なポイントでは、次のステップに進む前にユーザーがマイグレーションの計画を検討する機会があります。
- 複数のプロジェクト・バージョン、および複数のサブシステム間で情報を保持できるように、計画したマイグレーションに関する情報をデータベースに保管します。またこれにより、中間結果をバックアップとして保管できます。これは、リポジトリに多数のパーツがある場合に重要です。
- VAGen ソースをリポジトリから抽出して、マイグレーション・データベースをロードする、一連のサンプル・プログラムがツールのために用意されています。環境を正確に反映するように、オプションでサンプル・プログラムを作り替えることもできます。

VisualAge Generator to EGL マイグレーション・ツールは、次の前提に基づいて設計されています。

- VisualAge Generator 4.5 からのマイグレーションが、VisualAge Generator 4.5 で生成された外部ソース形式を使用して行われる。
- マイグレーション対象のパーツは、有効な VisualAge Generator パーツである。プログラム、テーブル、およびマップ・グループは、VisualAge Generator 4.5 内で検証または生成 (あるいはその両方) できる。

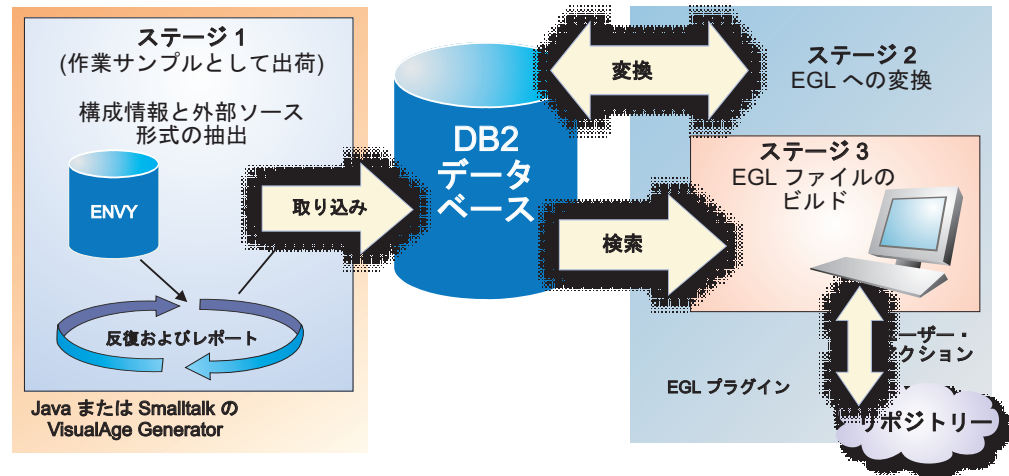
VisualAge Generator to EGL マイグレーション・ツールを使用するには、次の 2 つの方法があります。

- ステージ 1 から 3 のマイグレーション。これについては、24 ページの『VisualAge Generator to EGL マイグレーション・ツールの概要』で説明します。これは、ソース・コードをマイグレーションするための基本手法です。

- 単一ファイル・マイグレーション。これについては、32 ページの『単一ファイル・マイグレーションの概要』で説明します。この手法は、環境が正しく動作することを確認するために、少数のプログラムをマイグレーションする場合に便利です。

VisualAge Generator to EGL マイグレーション・ツールの概要

前述の目的を果たすために、VisualAge Generator to EGL マイグレーション・ツールは実際には複数のツール群で構成され、次の図に示すように 3 つのステージに編成されています。



- ステージ 1 用のツールは、VAGen 環境で実行されます。ステージ 1 ツールは、ソース・コードの編成、およびソース・コード自体に関する情報を、Java リポジトリまたは Smalltalk ライブラリーから抽出します。ステージ 1 ツールはまた、EGL プロジェクト、パッケージ、およびファイルの編成における各パーツの配置も決定します。ステージ 1 ツールは、この情報をマイグレーション・データベースにロードします。VAGen ソース・コードは、外部ソース形式で保管されます。
- ステージ 2 ツールは、EGL 環境で実行します。ステージ 2 ツールは、マイグレーション・データベースに保管されている情報を使用して、ステージ 1 の実行中にマイグレーション・データベースに保管された VAGen パーツの EGL 構文を作成します。ステージ 2 ツールは、得られた EGL ソース・コードをマイグレーション・データベースに保管します。
- ステージ 3 ツールも EGL 環境で実行します。作成するそれぞれの EGL プロジェクトごとに、ステージ 3 ツールはそのプロジェクトに属するパーツの EGL ソースを抽出して、EGL プロジェクトをファイル・システム内に作成します。プロジェクト・セットの 1 つのバージョンのみを使用する場合、オプションでステージ 3 ツールでプロジェクトをご使用のワークスペースにインポートできます。

プロジェクトをワークスペースに配置した後は、使用を決定したソース・コード・リポジトリの提供するツールを使用して、ソース・コードを管理できます。

マイグレーション・ツールの用語

パーツ間マイグレーションを問題なく行うには、パーツ自体だけでなく、そのパーツが参照するパーツすべてを用意する必要があります。例えば、プログラムをマイグレーションする場合は、そのプログラムだけでなく、プログラムが参照するパーツすべてを提供する必要があります。プログラムの場合、プログラムからマイグレーションする際に必要なパーツのセットは、プログラムを VisualAge Generator 内で生成するときに必要なパーツのセットと同じです。このパーツのセットは、プログラムの関連パーツ・リストです。

VisualAge Generator では、以下の共通技法によって、生成のためのパーツがすべて提供されます。

- VisualAge Generator on Java のプロジェクト・リスト・パーツ (PLP)
- VisualAge Generator on Smalltalk の構成マップ

マイグレーション・ツールは、これら 2 つの技法を使用します。このツールは、次の用語を使用します。

- VisualAge Generator on Java からのマイグレーションの場合:
 - 上位 PLP プロジェクト は、プロジェクト・リスト・パーツ (PLP) を含む Java プロジェクトで、他の PLP からは参照されません。
 - マイグレーション・セット は、Java 上位 PLP プロジェクト内で参照されるすべての VAGen プロジェクトからなり、上位 PLP プロジェクトから始まる PLP チェーン全体の VAGen プロジェクトをすべて含みます。
- VisualAge Generator on Smalltalk からのマイグレーションの場合:
 - 上位構成マップ は、他の構成マップに必須マップとしてリストされていない Smalltalk 構成マップです。
 - マイグレーション・セット は、Smalltalk 上位構成マップで必須マップとしてリストされているすべての Smalltalk 構成マップ (上位構成マップをはじめとする、Smalltalk 必須マップのチェーン全体に渡るすべての構成マップを含む) で構成されます。
- マイグレーション・プラン は、1 つ以上のマイグレーション・セットに関する情報を指定するファイルです。ステージ 1 の設定中にマイグレーション・プラン・ファイル名を指定した場合は、リポジトリ・フィルターにマッチングするマイグレーション・セットがすべて同じマイグレーション・プラン・ファイル内に置かれます。マイグレーション・プラン・ファイル名を指定しない場合は、それぞれのマイグレーション・セットが別々のマイグレーション・プラン・ファイルに置かれます。

注: VisualAge Generator on Java からのマイグレーションを行う場合に、PLP プロジェクトを現在使用していなければ、マイグレーションのみに使用する PLP を作成できます。その代わりとして、以下のソリューションを使用することもできます。

- Java プロジェクトのバージョンに関連して、生成に必要なものを指定する情報がデータベースや他のシステム内にある場合は、データベースからマイグレーション・プラン・ファイルを自動的に作成するツールをユーザーが開発できます。
- マイグレーション・プラン・ファイルを手作業で作成できます。

VisualAge Generator on Java からのマイグレーションを行う場合、詳細については 169 ページの『マイグレーション・プランと上位 PLP プロジェクト』を参照してください。

ステージ 1 の詳細

ステージ 1 ツールは、サンプル・プログラムとして EGL デベロッパー製品に付属しています。現在使用している VisualAge Generator Developer 4.5 製品に応じて、VisualAge Generator Developer on Java、または VisualAge Generator Developer on Smalltalk 上で稼働するサンプル・プログラムをインストールします。Java 環境と Smalltalk 環境の違いにより、2 つのサンプル・プログラムは多少異なります。ただし、ステージ 1 サンプル・プログラムを使用するための基本的な手順は、どちらの環境でも同じです。ステージ 1 ツールを実行するには、以下の基本手順を実行する必要があります。

1. ステージ 1 マイグレーションを指示するための規則と設定を定義します。
2. ツールを実行して、以下の結果を 1 つ以上生成します。
 - 1 つ以上のマイグレーション・プラン・ファイル。
 - 検出された問題に関するメッセージを含むログ・ファイル。
 - マイグレーション・データベース。
 - ステージ 2 および 3 で各マイグレーション・プラン・ファイルがどのようにマイグレーションされるかを示したレポート。

ステップ 1

以下の情報を含めて、マイグレーションする対象に関する情報をステージ 1 ツールに提供する規則と設定を定義します。

- マイグレーションするプロジェクトのみが対象となるよう Java プロジェクト名をフィルターに掛ける方法。Smalltalk の場合は、Smalltalk 構成マップ名をフィルターに掛ける方法を指定します。これにより、フィルターにマッチングする Java プロジェクトまたは Smalltalk 構成マップのみをツールが処理するようになるので、ステージ 1 のパフォーマンスが向上します。
 - フィルターにマッチングする Java プロジェクトから、ステージ 1 ツールは上位のプロジェクト・リスト・パーツを含む Java プロジェクトを選択します。Java プロジェクトが他の PLP によって参照されていない場合、その Java プロジェクトは上位のプロジェクト・リスト・パーツ (PLP) を含んでいます。
 - フィルターにマッチングする Smalltalk 構成マップから、ステージ 1 ツールは上位構成マップを選択します。上位構成マップは、他の構成マップに必須マップとしてリストされていないものです。
- フィルターに基づいてマイグレーション可能な対象をすべて反映したマイグレーション・プランを 1 つ作成するか、複数のマイグレーション・プラン (それぞれの Java 上位 PLP プロジェクトのバージョンごと、または Smalltalk 上位構成マップのバージョンごとに 1 つずつ) を作成するか。
- Java プロジェクト名とパッケージ名から、または Smalltalk 構成マップ名とアプリケーション名から、EGL プロジェクト名、パッケージ名、およびファイル名を作成する方法。これには、以下の情報が含まれます。
 - 共通コードを含む Java プロジェクトとパッケージ、または Smalltalk 構成マップとアプリケーションを指示する規則。

- EGL プロジェクト名とパッケージ名を作成する際に使用される名前変更規則。
- 共通パーツ、または未使用パーツを含む EGL ファイルに使用される名前。
- 同じマップ・グループのマップが、複数の Java プロジェクトまたは Smalltalk 構成マップに配置されている場合は、マイグレーション・セット名の接尾部として使用される名前。同様に、同じマップ・グループのマップが、単一プロジェクト内の複数の Java パッケージにある場合、または単一の構成マップ内の複数の Smalltalk アプリケーションにある場合は、これをプロジェクト名の接尾部として使用される名前にすることができます。
- マイグレーション・データベースの名前、およびデータベースにアクセスするために必要なユーザー ID とパスワード。
- ステージ 1 ツールがステップ 2 で作成する出力。すべての出力を単一ステップで作成するように指示することもでき、または出力を順次作成して、長時間かかる次の出力の作成前に規則と設定を検討する機会を設けることもできます。
- ログ・ファイルおよびデバッグ・ファイルの名前、およびデバッグ・ファイルに入れる詳細情報のレベル。

ステップ 2

ステージ 1 ツールは、定義した規則と設定に基づいて、以下のような出力を生成します。

- **マイグレーション・プラン・ファイル (複数可)**。マイグレーション・プラン・ファイルは、マイグレーション・セットを格納しています。それぞれのマイグレーション・セットは、Java リポジトリからの上位 PLP プロジェクトの 1 バージョン、または Smalltalk ライブラリーからの上位構成マップの 1 バージョンです。従属 Java プロジェクトのバージョン、または必須 Smalltalk 構成マップのバージョンが、マイグレーション・セット内で指定されます。
- マイグレーション設定ファイルでマイグレーション・プランの **filename** オプションに値が指定されていない場合は、複数のマイグレーション・プラン・ファイルが作成されます。Java 用の上位 PLP プロジェクトのバージョンごとに、マイグレーション・セットを 1 つ含むマイグレーション・プラン・ファイルが 1 つずつ作成されます。同様に、Smalltalk 用の上位構成マップのバージョンごとに、マイグレーション・セットのバージョンを 1 つ含むマイグレーション・プラン・ファイルが 1 つずつ作成されます。
- マイグレーション設定ファイルでマイグレーション・プラン **filename** オプションの値が指定されている場合は、Java の各上位 PLP プロジェクト・バージョンごとに、単一のマイグレーション・プラン・ファイル内にマイグレーション・セット項目が作成されます。同様に、Smalltalk 用の上位構成マップのバージョンごとに、単一のマイグレーション・プラン・ファイル内にマイグレーション・セット項目が 1 つずつ作成されます。
- 例えば、5 つの Java プロジェクトと、それら 5 つのプロジェクトのバージョンを指定する PLP を含む 6 番目の Java プロジェクトからなる受注システムがあるとします。複数のマイグレーション・プランと 3 つのバージョンを要求すると、ステージ 1 ツールは、PLP パーツを含む Java 受注プロジェクトのバージョンごとに、1 つずつ 3 つのマイグレーション・セットを作成します。同様に Smalltalk の場合は、受注システムを構成するコードを反映する構

成マップのバージョンを 3 つマイグレーションする場合、ステージ 1 ツールはこの上位構成マップのバージョンごとに 1 つずつ、計 3 つのマイグレーション・セットを作成します。

この時点でステージ 1 ツールに停止を指示することにより、マイグレーション・プラン・ファイルを検討して、マイグレーションしたい Java プロジェクトのバージョン、または Smalltalk 構成マップのバージョンがマイグレーション・プラン・ファイルに正しく反映されているかどうか確認できます。

- VAGen プログラム、テーブル、マップ・グループ、または制御パーツの名前が EGL 予約語リストのいずれかと競合する場合は、**ログ・ファイルにメッセージが示されます**。これらのパーツの名前は、マイグレーション中には変更されません。VisualAge Generator 内でパーツの名前を変更するか、EGL へのマイグレーションが済むまで待つことができます。

また、このログ・ファイルには、EGL 予約語リストと矛盾する UI レコード名、あるいは「#」または「@」で始まる UI レコード名に関するメッセージも含まれます。UI レコードは、ステージ 2 のマイグレーション中に名前変更されます。

- マイグレーション・プラン・ファイルに基づいて情報と VAGen ソース・コードがロードされた**マイグレーション・データベース**。データベースのロードに使用するマイグレーション・プランを 1 つ選択することも、ディレクトリー内のマイグレーション・プラン・ファイルをすべて選択することもできます。ステージ 1 マイグレーション・ツールは、データベースに以下の情報をロードします。
 - 選択したマイグレーション・プラン・ファイル (複数可) 内の各マイグレーション・セットに関する情報。
 - マイグレーション・セットに関連した Java プロジェクトまたは Smalltalk 構成マップのセット。
 - Java プロジェクトまたは Smalltalk 構成マップのセットに含まれるそれぞれの VAGen パーツごとの、外部ソース形式の VAGen パーツ定義。
 - それぞれの Java プロジェクト、パッケージ、および VAGen パーツごと、またはそれぞれの Smalltalk 構成マップ、アプリケーション、および VAGen パーツごとの、対応する EGL プロジェクト、パッケージ、およびファイル名。
- **ステージ 2 および 3 で各マイグレーション・セットがどのようにマイグレーションされるかを示したレポート**。このレポートでは、以下の情報が示されます。
 - Java の場合は、それぞれのマイグレーション・セットごとに、組み込まれるプロジェクトのバージョンがリストされます。それぞれのプロジェクト・バージョンごとにパッケージ・バージョンが示され、それぞれのパッケージ・バージョンごとに VAGen パーツのリストが示されます。
 - Smalltalk の場合は、それぞれのマイグレーション・セットごとに、組み込まれる構成マップのバージョンがリストされます。それぞれの構成マップ・バージョンごとにアプリケーション・バージョンが示され、それぞれのアプリケーション・バージョンごとに VAGen パーツのリストが示されます。

VAGen パーツごとに、各パーツに対応し、そのパーツが配置されている EGL プロジェクト、パッケージ、およびファイルの名前が示されます。また、それぞれの VAGen パーツごとに、VisualAge Generator によって作成された関連パーツのリスト、および関連パーツが配置される EGL ファイルも示されます。ステージ

1 から 3 までのマイグレーション中に、VAGen パーツがどのようにファイルに割り当てられるかについては、50 ページの『EGL ファイル内でのパーツの配置』を参照してください。

ステージ 1 ツールは、VisualAge Generator の Java と Smalltalk の両バージョン用サンプル・プログラムとして付属しています。ステージ 1 ツールは「現状のまま」使用でき、ご使用の環境に適合するようにサンプル・プログラムを変更することもできます。例えば、

- 現在は構成情報を Java リポジトリまたは Smalltalk ライブラリーの外部に保管している場合があります。この構成情報は生成に必要なソース・コードのバージョンを指定しているとします。この状況では、サンプル・プログラムをガイドとして使用して、構成情報と Java リポジトリまたは Smalltalk ライブラリーの組み合わせからマイグレーション・データベースをロードするためのツールをユーザーが独自に作成できます。
- パーツ配置アルゴリズムを変更したい場合があります。ステージ 1 マイグレーション・ツールのパーツ配置アルゴリズムの変更が必要になる可能性がある考慮事項については、65 ページの『EGL ソース・コードの編成方法の決定』を参照してください。

ステージ 1 サンプル・プログラムを変更した場合は、マイグレーション・データベースを変更して、コードの分析に役立つ追加情報を組み込むことができます。既存の SQL 表に列を追加したり、マイグレーション・データベースに表を追加したりすることができます。ただし、これらの新しい列と表は、マイグレーションのステージ 2 と 3 では使用されません。また、ステージ 1 サンプル・プログラムを変更する場合は、サンプル・プログラムに示されている情報を SQL 表に必ず取り込む必要があります。取り込まなかった場合は、ステージ 2 と 3 でコードをマイグレーションできません。

ステージ 1 ツールを VisualAge Generator Developer on Java にインストールして実行する場合の詳細については、149 ページの『第 4 章 ステージ 1 - Java からの抽出』を参照してください。ステージ 1 ツールを VisualAge Generator Developer on Smalltalk にインストールして実行する場合の詳細については、177 ページの『第 5 章 ステージ 1 - Smalltalk からの抽出』を参照してください。

ステージ 2 の詳細

ステージ 2 ツールは、Eclipse プラグイン `com.ibm.etools.egl.vagenmigration` に含まれており、EGL 環境で稼働します。この時点で、マイグレーションする情報はマイグレーション・データベース内にあるので、VisualAge Generator on Java または VisualAge Generator on Smalltalk のどちらからマイグレーションするかに関係なく、同じステージ 2 ツールを使用します。ステージ 2 ツールを実行するには、以下の基本手順を実行する必要があります。

1. 以下の情報を含めて、マイグレーション対象をステージ 2 ツールに指示する規則と設定を定義します。
 - EGL ソース・コードの作成方法に関する具体的な詳細。例えば、ステージ 2 マイグレーション・ツールは、VAGen 作業用ストレージ・レコードを次の 2 つの EGL 基本レコードに分割する必要があります。

- オリジナルの作業用ストレージ・レコードと同じ名前の、レベル 77 以外の項目をすべて含むレコード。
- オリジナルの作業用ストレージ・レコードと同じ名前に接尾部が付いた、レベル 77 項目をすべて含む 2 つ目のレコード。

ステージ 2 のマイグレーション設定を使用して、レベル 77 項目を含むレコードを新規作成する際にステージ 2 ツールが使用する接尾部を指定できます。

- マイグレーションするマイグレーション・セット。例えば、3 つの異なるバージョンの受注システムをマイグレーションするために 3 つのマイグレーション・セットを作成していても、最初は 1 つのバージョンのみをマイグレーションしたい場合があります。この機能により、マイグレーションの制限を柔軟に設定でき、マイグレーション・データベース内のものをすべて同時にマイグレーションする必要がなくなります。
 - マイグレーション・データベースの名前、およびデータベースにアクセスするために必要なユーザー ID とパスワード。データベースのユーザー ID とパスワードが明示的に指定されていない場合、ステージ 2 とステージ 3 のマイグレーション・ツールはどちらも、Windows マシンへのログオンに使用されたユーザー ID とパスワードを使用してデータベースにログオンしようとします。
 - ステージ 2 の完了後、ステージ 3 ツールを自動的に開始するかどうか。ステージ 3 を自動的に実行する場合は、EGL プロジェクトの 1 バージョンをワークスペースにロードするか、または後でソース・コード・リポジトリとインターフェースを取ることができるように、EGL プロジェクトを一時ディレクトリにロードするかを選択できます。
2. ステージ 2 ツールは、定義した規則と設定に基づいて、以下の処理を行います。
- データベースから 1 つのマイグレーション・セット用のパーツを検索します。
 - 外部ソース形式のソース・コードを EGL ソース・コードに変換します。
 - EGL ソース・コードをマイグレーション・データベースに保管します。パーツのマイグレーションに関連したメッセージも、マイグレーション・データベースに保管されます。これにより、ステージ 2 のパフォーマンスが向上します。同じパーツ・エディションが別のマイグレーション・セット内で使用されている場合、EGL ソース・コードはすでに使用可能になっており、再変換は行われないからです。
 - 検出された潜在的な問題のログ・ファイルを作成します (例えば、生成可能パーツと EGL 予約語リストの競合や、マイグレーション・ツールが解決できない未確定状態など)。
 - 次に選択されたマイグレーション・セットに対してプロセスを繰り返します。

ステージ 2 マイグレーション・ツールはバッチ・モードで実行できます。EGL 開発環境でのステージ 2 ツールの実行の詳細については、207 ページの『第 6 章 ステージ 2 - EGL 構文への変換』を参照してください。ステージ 2 マイグレーション・ツールを変更することはできません。

ステージ 3 の詳細

ステージ 3 ツールは、ステージ 2 ツールと同じ Eclipse プラグイン (com.ibm.etools.egl.vagenmigration) で出荷され、同様に EGL 環境で稼働します。この時点で、マイグレーションする情報はマイグレーション・データベース内にあるので、VisualAge Generator on Java または VisualAge Generator on Smalltalk のどちらからマイグレーションするかに関係なく、同じステージ 3 ツールを使用します。ステージ 3 ツールを実行するには、以下の基本手順を実行する必要があります。

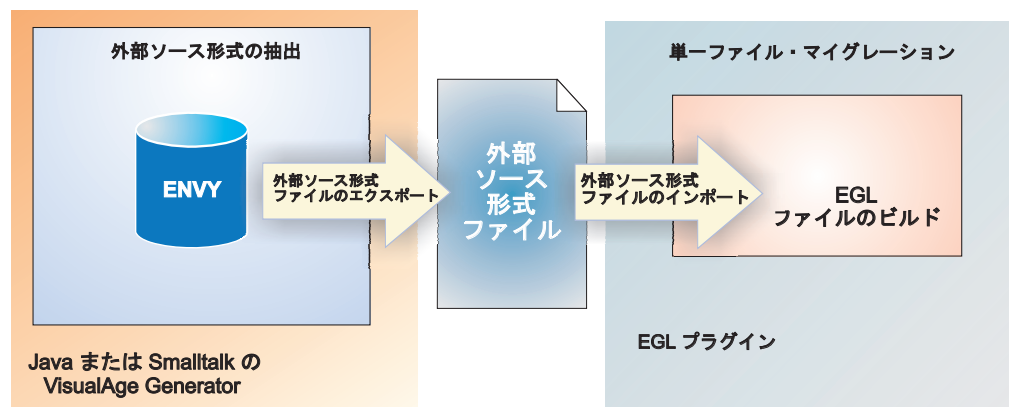
1. 以下の情報を含めて、マイグレーション対象をステージ 3 ツールに指示する規則と設定を定義します。
 - マイグレーションするマイグレーション・セット。例えば、3 つの異なるバージョンの受注システムをマイグレーションするために 3 つのマイグレーション・セットを作成していた場合、ステージ 2 ツールで 3 つのバージョンすべてをマイグレーションしていても、ステージ 3 ツールでは 1 つのバージョンのみをマイグレーションしたい場合があります。ステージ 3 で 1 バージョンだけを処理する理由として最も一般的な状況は、ソース・コード・リポジトリ内でこのコードのバージョン管理を行い、その後でステージ 3 ツールを使用して次のバージョンをマイグレーションし、ソース・コード・リポジトリ内でそのバージョンを管理したい場合です。
 - マイグレーション・データベースの名前、およびデータベースにアクセスするために必要なユーザー ID とパスワード。
2. ステージ 3 ツールは、定義した規則と設定に基づいて、以下の処理を行います。
 - マイグレーション・セットの「to do」リストを作成します。この「to do」リストは、ステージ 2 で生成されたメッセージを統合したリストで、マイグレーションを完了するために必要になる可能性がある追加作業を示しています。
 - ステージ 1 でマイグレーション・データベースに保管された情報に基づいて、EGL プロジェクトとパッケージの構造をワークスペースに作成します。
 - ステージ 2 で保管された VAGen パーツの EGL ソース・コードに基づき .egl ソース・ファイルを作成します。この .egl ソース・ファイルには、EGL パーツ参照を解決するのに必要な、ほとんどの **import** ステートメントが組み込まれています。**import** ステートメントについて詳しくは、45 ページの『EGL のビルド・パスと import ステートメント』を参照してください。
 - ステージ 2 で保管された VAGen 制御パーツの EGL XML ソースに基づいて、.eglbld ファイルを作成します。制御パーツは、生成オプション (EGL ビルド記述子パーツ)、リンケージ・オプション、リソース関連、バインド制御、およびリンク・エディット・パーツです。
 - EGL プロジェクトのビルド順序を最適化するツールを呼び出します。
 - EGL 検査が実行されるよう、ワークスペースをリフレッシュします。
3. この時点では、以下の手順を実行する必要があります。
 - a. オプションで、EGL プロジェクトをソース・コード・リポジトリにバージョン管理またはコミットして、マイグレーションされたときとまったく同じコードを反映するベースラインを確立する。
 - b. ワークスペースの「問題」ビューにあるメッセージを検討して、検証エラーがないかどうか確認する。この確認は、ステージ 2 で生成されたログ、またはステージ 3 で作成された「to do」リストを併用して行うことができます。

- c. ターゲット環境へのマイグレーションが正しく行われるように、すべてのプログラムと DataTable を（準備せずに）生成する。プログラムを生成する場合は、すべての FormGroup が作成されるよう、必ず **genFormGroup** と **genHelpFormGroup** ビルド記述子オプションを使用します。このステップはオプションですが、準備プロセスのオーバーヘッドを伴わずに生成の問題を早期に検出するのに役立ちます。
- d. EGL プロジェクトをソース・コード・リポジトリにバージョン管理またはコミットし、問題を解決するために行ったすべてのコード変更を反映する新しいベースラインを確立する。
- e. マイグレーション済みコードを生成し、テストする。使用しているランタイム環境に応じて、このステップは必須またはオプションになります。生成とテストの利点は、このステップが、コードの正しいバージョンをマイグレーションしたこと、そのコードが正しくマイグレーションされたこと、およびそのコードが VAGen ランタイム環境と同じように実行することを確認するのに役立つ点です。この時点で生成とテストを実行しておかないと、プログラムに初めて変更を加えたときの動作に違いが生じ、その違いが、行ったばかりの変更によるものなのか、マイグレーション・プロセスによるものなのかを判別するのが難しくなります。

ステージ 3 マイグレーション・ツールはバッチ・モードで実行できます。EGL 開発環境でのステージ 3 ツールの実行の詳細については、229 ページの『第 7 章 ステージ 3 - インポート』を参照してください。ステージ 3 マイグレーション・ツールを変更することはできません。

単一ファイル・マイグレーションの概要

EGL に慣れてきて初めて環境をセットアップする場合、少数のプログラムのみをマイグレーションして使用する環境を検証し、生成および準備が正しく動作することを確認して、EGL に関するランタイム環境が正しく構成されていることを確認できます。この場合に、ステージ 1 から 3 のマイグレーションを完全に行う必要はありません。マイグレーション・ツールには、下図に示すように、単一ファイル・マイグレーションを使用していくつかのプログラムをマイグレーションするための仕組みが備わっています。



単一ファイル・マイグレーションは、ステージ 1 から 3 のマイグレーションに比べて手操作の部分が多いプロセスです。単一ファイル・モードでは、VisualAge Generator を使用して、外部ソース形式のソース・コードをファイルにエクスポート

します。その後、EGL 開発環境を使用して EGL プロジェクトと EGL パッケージを作成します。「インポート」ウィザードを使用して、外部ソース形式ファイルを EGL ソースにインポートできます。単一ファイル・マイグレーション・ツールの機能は以下のとおりです。

- 存在しない場合は、ターゲット EGL ソース・ファイルを作成します。そのファイルが存在する場合は、上書きするか、ファイルに追加するかを選択できます。使用している設定と、外部ソース形式ファイルに含まれるパーツによっては、追加の EGL ファイルがマイグレーション・ツールによって作成される場合があります。
- 外部ソース形式のソース・コードを EGL ソース・コードに変換します。
- 検出された潜在的な問題のログ・ファイルを作成します (例えば、生成可能パーツと EGL 予約語リストの競合や、マイグレーション・ツールが解決できない未確定状態など)。

単一ファイル・マイグレーション中に行われる外部ソース形式から EGL への変換は、ステージ 1 から 3 のマイグレーションのステージ 2 で行われる構文変換と本質的には同じです。ただし、単一ファイル・マイグレーションにはいくつかの制限があり、大規模マイグレーションには適していません。次のような制限があります。

- 単一の外部ソース形式ファイルにあるパーツのみが、マイグレーション中に処理されます。実現可能な最良のマイグレーションを行うには、プログラムとその関連パーツをすべて外部ソース形式ファイルに組み込みます。
- VAGen パーツのファイルへの配置は、ステージ 1 から 3 のマイグレーションの配置とは異なります。単一ファイル・モードでは、ターゲット EGL ファイル名として *targetFile.egl* を指定したと想定すると、マイグレーション・ツールは以下の方法でそれらのパーツをファイル内に配置します。
 - *targetFile* という名前のファイルに、すべての制御パーツが配置されます。
 - 各 UI レコードはそれぞれ単独で *uiRecordName.egl* (*uiRecordName* は UI レコードの名前) という名前のファイルに格納されます。
 - 生成可能パーツを複数の EGL ファイルに分割する設定を選択していない場合、残りのパーツはすべて *targetFile.egl* という名前のファイルに格納されます。
 - 生成可能パーツを複数の EGL ファイルに分割する設定を選択すると、マイグレーション・ツールはそれらのパーツを以下の方法でファイルに格納します。
 - それぞれのプログラム・パーツは、*programName.egl* という名前のファイルに格納されます (ここで、*programName* はプログラムの名前)。
 - それぞれのテーブル・パーツは、*tableName.egl* という名前のファイルに格納されます (ここで、*tableName* はテーブルの名前)。
 - それぞれのマップ・グループと、マップ・グループ内のすべてのマップは、*mapGroupName.egl* という名前のファイルに格納されます (ここで、*mapGroupName* はマップ・グループの名前)。
 - 残りのパーツはすべて、*targetFile.egl* という名前のファイルに格納されます。残りのパーツをすべてプログラムと同じファイルに格納したい場合は、*targetFile* を *programName.egl* と同じ名前にすることができます。単一ファイル・マイグレーション・ツールは、複数の生成可能パーツが共用しているパーツを判別しません。

- すべてのファイルが、同じ EGL プロジェクト、ソース・フォルダー、およびパッケージに配置されます。
- すべての出力ファイルは同じ EGL パッケージに配置されるので、マイグレーション・ツールは **import** ステートメントを組み込みません。また、すべてのパーツが同じ EGL パッケージに配置されるので、オリジナルの Java プロジェクトとパッケージの構造は保持されません。同様に、オリジナルの Smalltalk 構成マップとアプリケーションの構造は保持されません。
- 同じパーツが外部ソース形式ファイルに複数回現れる場合は、最後の定義のみがマイグレーションされます。
- 単一ファイル・モードでマイグレーションを行う場合に、共通パーツを処理するための技法を 4 つの中から選択できます。いずれかを選択する前に、それぞれの技法の欠点について理解してください。以下の技法を使用できます。
 - 複数のプログラムとその関連パーツを含む大きな外部ソース形式ファイルをマイグレーションする場合は、1 つのターゲット・ファイル名しか指定できません。マイグレーション設定をパーツを **EGL ファイルに分離** に選択した場合、マイグレーション・ツールはデータ項目、機能、PSB、および非 UI レコードを単一のターゲット・ファイルに格納します。複数のプログラムがパーツを共有しているため同じ関連が複数回このファイルに現れたとしても、マイグレーション・ツールがマイグレーションするのは各パーツについて 1 つの定義のみです。したがって、この技法を使用することによってパーツが重複することはありません。ただし、複数のプログラム内に非常に多くの関連パーツがある場合は、ターゲット・ファイルがきわめて大きくなる可能性があります。
 - 2 つの外部ソース形式ファイルを同じ EGL パッケージにマイグレーションする場合、それら 2 つのファイルに、異なる ターゲット・ファイル名を指定した同じパーツが含まれていると、マイグレーション・ツールは重複するパーツを作成します。プログラム 1 とプログラム 2 がいくつかの共通パーツを共有している場合に、以下の手順を使用してマイグレーションするとどうなるかを考えてみます。
 1. プログラム 1 とその関連パーツを含むファイルをターゲット・ファイルにマイグレーションします。
 2. プログラム 2 とその関連パーツを含む 2 番目のファイルを別のターゲット・ファイルを使用してマイグレーションします。

この場合、マイグレーション・ツールは共通パーツを両方のターゲット・ファイルに格納します。両方のターゲット・ファイルを同じパッケージに格納すると、パッケージ内のパーツの名前が重複するため、EGL はパーツ名を解決できません。それぞれのプログラムとその関連パーツを別々の EGL パッケージにマイグレーションすれば、この問題を回避できます。この場合もワークスペース内に重複パーツが生じますが、これらのパーツは別々のパッケージにあるので、EGL はパーツ参照を解決できます。
 - 2 つの外部ソース形式ファイルを同じ EGL パッケージにマイグレーションする場合、それら 2 つのファイルに、**同じ** ターゲット・ファイル名を指定した同じパーツが含まれていると、ワークスペース内の既存のターゲット・ファイルを上書きするかどうかを尋ねるプロンプトが出されます。
 - 既存のターゲット・ファイルを上書きしないことを指定した場合、2 番目のインポート内のデータ項目、関数、PSB、および非 VGUI レコードは、その

ターゲット・ファイルに追加されます。 2 番目のインポート内の共通パーツは、ターゲット・ファイル内で重複パーツになります。

- 既存のターゲット・ファイルに上書きすることを指定した場合、2 番目のインポートのデータ項目、関数、PSB、および非 VGUI レコードは、既存のターゲット・ファイルを完全に置き換えます。 このため、最初のインポートに含まれていて、2 番目のインポートに含まれていないパーツはすべて失われます。
- マイグレーション設定として「パーツを EGL ファイルに分離」を選択した場合、マイグレーション・ツールは、プログラム、FormGroup、および DataTable 用に作成されたファイルを上書きします。この設定を選択しなかった場合、これらのパーツはターゲット・ファイルに格納され、上書き確認プロンプトに対する応答に従って追加または上書きされます。
- マイグレーション・ツールは、VGUI レコード用のファイルと .eglbld ファイルを常に上書きします。

最初の技法と同様に、この問題は、それぞれのプログラムとその関連パーツを別々の EGL パッケージにマイグレーションすることにより回避できます。この場合もワークスペース内に重複パーツが生じますが、これらのパーツは別々のパッケージにあるので、EGL はパーツ参照を解決できます。

- 共通パーツを別の外部ソース形式ファイルに分割すると、単一ファイル・ベースで VisualAge Generator から EGL へのマイグレーションを正常に行うために必要な情報がすべてそろわない場合があります。例えば、1 つの外部ソース形式ファイルに SQL レコードがあり、そのレコードの変更された SQL を使用する関数が別のファイルにある場合、マイグレーション・ツールはその関数の入出力ステートメントのビルドを完全に行うことができません。また、共通パーツが別のパッケージに存在する場合、共通パッケージを参照する必要がある各ファイルに EGL **import** ステートメントを追加する必要があります。
- ステージ 2 および 3 のマイグレーションで組み込まれる次の処理は、単一ファイル・モードのマイグレーションでは組み込まれません。
 - FormGroup 内の書式のネスト。
 - 出力ファイル内のパーツ間の、複数のブランク行の組み込み。

単一ファイル・マイグレーションとステージ 1 から 3 のマイグレーションとの違いについて詳しくは、35 ページの『マイグレーションの考慮事項』、および 42 ページの『VisualAge Generator から EGL へのマイグレーション・ツールが使用する技法』を参照してください。

マイグレーションの考慮事項

ソース・コードを作成および管理するためのアプローチには、VisualAge Generator と EGL の間でいくつかの相違点があります。特に、次の相違点はマイグレーションには重要です。

- EGL の構文は VisualAge Generator よりも厳密な場合がある
- パーツ参照が解決される時期と方法の違い
- 共通コードの処理の違い

次に、これらの相違点について詳しく説明します。

EGL 構文の厳密さ

2 つの言語の構文は大きく異なりますが、VAGen 言語の大部分は、オリジナルの VAGen プログラムと同じ振る舞いを保持しながら EGL 言語にマイグレーションできます。ただし、状況によっては、EGL の構文が VisualAge Generator よりも厳密または制限的になることがあります。このような状況は、標準的なプログラムではあまり起こりません。ただし、この状況が発生した場合、マイグレーション・ツールはパーツ間マイグレーションを行って、VisualAge Generator 内で必要だった振る舞いを保持する正確な EGL 構文を判別する必要があります。パーツ間マイグレーションとは、マイグレーション・ツールが現行パーツを正しくマイグレーションするには、参照を受けている他のパーツを 1 つ以上使用可能にする必要があることを意味します。以下にその例を示します。

- VisualAge Generator の場合は、表示 (テキスト) マップとプリンター・マップの両方に `DISPLAY` 入出力オプションを使用します。EGL は、テキスト書式用の **display** ステートメントと印刷書式用の **print** ステートメントを提供しています。VisualAge Generator からのマイグレーションを容易にするために、VisualAge Generator との互換性が必要であることを示す EGL 設定があります。VisualAge Generator 互換の設定を使用すると、印刷書式に **display** ステートメントを使用できます。マイグレーション中に、プログラム、そのマップ・グループ、およびマップがすべて使用可能であれば、マイグレーション・ツールは **display** または **print** のどちらのステートメントにマイグレーションするかを判別できます。ただし、`DISPLAY` 関数がプログラムを使用せずにマイグレーションされる場合、マイグレーション・ツールは **display** または **print** のどちらの EGL ステートメントを使用するかを明確に判別することはできません。この場合、マイグレーション・ツールは、VisualAge Generator 互換モードで印刷書式に使用することが容認されている **display** ステートメントを使用します。
- VisualAge Generator では、表示 (テキスト) マップおよびプリンター・マップの両方に対して `SET map PAGE` ステートメントを使用します。このため、次の `CONVERSE` または `DISPLAY` の対象が表示マップである場合は画面が消去され、次の `DISPLAY` の対象がプリンター・マップである場合はページ替えが行われます。EGL は、テキスト書式用には `clearScreen()` システム・ライブラリー関数を、印刷書式用には `pageEject()` システム・ライブラリー関数を提供しています。VisualAge Generator 互換性の設定は、`clearScreen()` または `pageEject()` の使用には影響しません。マイグレーション中に、プログラム、そのマップ・グループ、およびマップがすべて使用可能であれば、マイグレーション・ツールは、`clearScreen()` または `pageEject()` のどちらのシステム・ライブラリー関数にマイグレーションするかを判別できます。ただし、プログラムを使用せずにマイグレーションされている関数で `SET map PAGE` ステートメントを使用すると、マイグレーション・ツールは `clearScreen()` または `pageEject()` のどちらのシステム・ライブラリー関数を使用するかを判別できません。この場合、マイグレーション・ツールは意図的に無効な EGL 構文である `EZE_SETPAGE` を使用します。この結果、エラーが「問題」ビューに表示されるので、関数の訂正が必要であることが分かります。
- VisualAge Generator の場合は、マップ変数フィールドの編集ルーチンとして、編集テーブルまたは編集関数のどちらかを指定できます。両方は指定できません。EGL の場合は、**validatorDataTable** および **validatorFunction** の両方のプロパティを指定できます。マイグレーション時に編集テーブルまたは編集関数が使用できる場合、マイグレーション・ツールは **validatorDataTable** または

validatorFunction のどちらのプロパティを設定するかを判別できます。ただし、編集ルーチンによって指定されているパーツが使用不可の場合、マイグレーション・ツールは EGL の **validatorDataTable** または **validatorFunction** のどちらのプロパティを設定するかを確実に判別することはできません。この場合、マイグレーション・ツールは、編集ルーチン名の長さ、編集メッセージの存在などの情報を使用して、編集ルーチンがテーブルまたは関数のどちらであるか判別を試みます。それでも判別できない場合、マイグレーション・ツールは **validatorFunction** プロパティを使用します。**validatorFunction** が関数ではない場合、または検索できない場合にのみ、EGL 検査によって「問題」ビューにエラー・メッセージが表示されます。

マイグレーション・ツールは、マイグレーション・セット内で使用可能なパーツをすべて使用して、未確定状態を解決します。このような未確定状態を最小限に抑えるために、マイグレーションの際には必ずすべての関連パーツを組み込んでください。例えば、プログラムをマイグレーションする場合は、VisualAge Generator 内でプログラムを生成するために必要なパーツをすべて組み込みます。これにより、可能なかぎり最適なパーツのマイグレーション結果が得られます。マイグレーション・ツールが未確定状態を解決する方法の概要については、次に示す項を参照してください。

- 54 ページの『プログラムを使用したマイグレーション』
- 55 ページの『関連パーツを使用したマイグレーション』
- 56 ページの『関連パーツを使用しないマイグレーション』

正しいマイグレーションを行うためにマイグレーション・ツールがパーツ間マイグレーションを行う必要があるすべての状況のリストと、追加のパーツが使用できない場合にマイグレーション・ツールがより良い選択を試行するために使用する技法については、79 ページの『第 3 章 未確定状態の処理』を参照してください。

パーツ名が解決される時期と方法

定義時に、VisualAge Generator はすべてのパーツの存在を必要としません。プログラム構造ダイアグラムの中で、VisualAge Generator は欠落しているマップ、レコード、テーブル、および関数に疑問符 (?) を付けて示します。ただし、共用データ項目を使用する場合など、他の場所では現在パーツが存在していないことを示すものはありません。パーツを VisualAge Generator に保管するときに基本的な構文の検証が行われますが、パーツ間での検証は、テスト、検証、または生成のときまで行われません。EGL の場合は、ファイルを保管するたびにより幅広い検証 (すべてのパーツ名を解決可能な検証を含む) が行われます。このため、問題があれば可能なかぎり早期に警告が示されます。

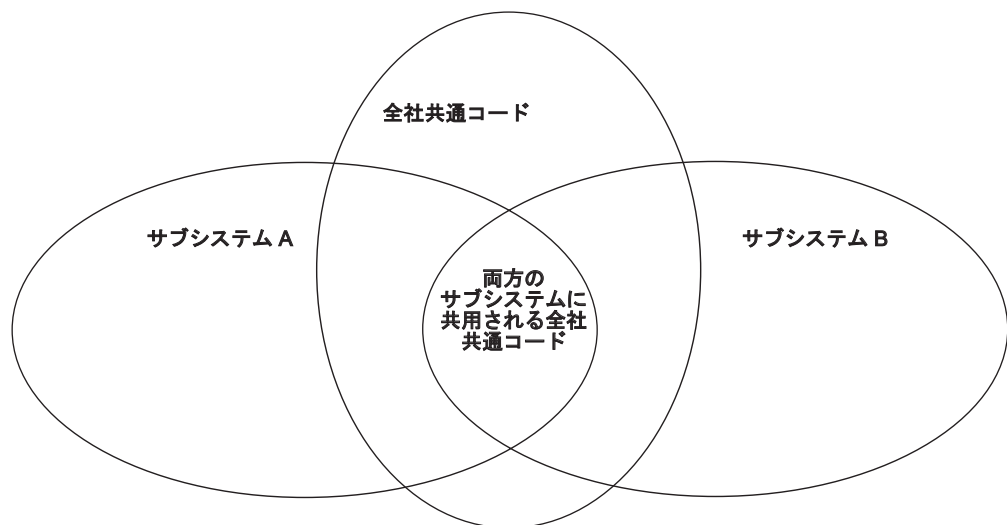
VisualAge Generator は、特定のパーツ名を見つけるために、ワークスペース内のパーツすべてを検索します。VisualAge for Java 内に重複するパーツ名がある場合は、重複パーツの問題が修正されるまで、テストと生成は停止します。VisualAge for Smalltalk の場合は、イメージに重複するパーツをロードすることは許されません。EGL の場合は、ワークスペース内で重複するパーツ名が許されます。EGL は、プロジェクトの EGL ビルド・パス、ファイルの **import** ステートメント、およびレコードと関数の **containerContextDependent** プロパティの組み合わせによって、使用するパーツの定義を判別します。

ステージ 1 から 3 のマイグレーションを使用してマイグレーションを行う場合、マイグレーション・ツールは、マイグレーション・セット内の使用可能なパーツに基づいて、プロジェクトの EGL ビルド・パスを設定し、ファイルに **import** ステートメントを組み込みます。正しい EGL ビルド・パスと **import** ステートメントが得られるように、マイグレーションの際には必ずすべての関連パーツを組み込んでください。例えば、プログラムをマイグレーションする場合は、VisualAge Generator 内でプログラムを生成するために必要なパーツをすべて組み込みます。これにより、可能なかぎり最適なパーツのマイグレーション結果が得られます。詳しくは、次に示す項を参照してください。

- 45 ページの『EGL のビルド・パスと import ステートメント』
- 48 ページの『containerContextDependent プロパティ』

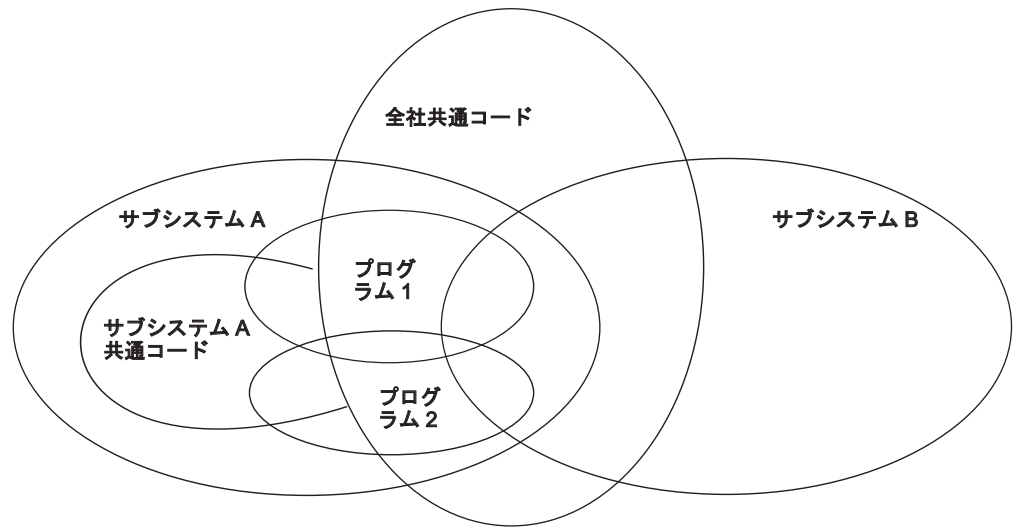
共通コードのシナリオ

共通コードは、サブシステムまたはプログラム間で共用されます。次の図に、2 つのサブシステム間で共用される共通コードを示します。



この例では、全社共通コードを含む Java プロジェクトまたは Smalltalk 構成マップが 1 つ以上あります。これらのプロジェクトまたは構成マップ内のコードは、複数のサブシステム間で共用できます。この例では、サブシステム A とサブシステム B が共通コードのサブセットを使用しており、全社共通コードの一部が両方のサブシステムによって使用されています。例えば、全社共通コードには、多数のサブシステム間で使用される SQL レコード定義を組み込むことができます。

次の図に、2 つのサブシステム間で全社共通コードを共用する基本的な方法を、サブシステム A の詳細とともに示します。



サブシステム A には、サブシステム A 共通コードがあります。このコードは、サブシステム A 内の複数のプログラムによって使用されますが、サブシステム A 内のプログラム以外には使用されません。この例では、プログラム 1 とプログラム 2 がそれぞれ、サブシステム A 共通コードの一部を全社共通コードの一部とともに使用しています。2 つのプログラム間で、サブシステム A 共通コードと全社共通コードの両方がオーバーラップする部分があり、サブシステム B が使用する全社共通コードとのオーバーラップもあります。例えば、サブシステム A 共通コードには、サブシステム A 内のプログラムのみが使用する SQL レコード定義を組み込むことができます。さらに、サブシステム A 内の複数のプログラムが使用するマップ・グループ定義も、サブシステム A 共通コードに組み込むことができます。

共通コードと VisualAge Generator

共通コードを使いやすくするために、VisualAge Generator はソース・コードの特定部分の解釈方法をテストと生成の時点で決定します。この利点は、それぞれのサブシステムまたはプログラムがコードに小規模の変更を加えるときに、プログラムが使用する特定のマップ・グループを変更するか、生成時にワークスペース内にあるデータ項目またはレコード定義を変更するだけで済むという点です。次の例は、この考えを説明するものです。

- 端末と対話するオンライン・プログラムと、類似したレポートを印刷するバッチ・プログラムとの間では、次の例のように同じロジックの大部分を共用できます。
 - プログラム A は、HEADER、DETAIL、および TRAILER という名前の表示マップを含む MapGrpA を使用する、メインのトランザクション・プログラムです。プログラム A は、部分的な HEADER マップを表示して、DETAIL 行を浮動域に表示し、ユーザーが次のレポートを要求するために使用する入力フィールドを含む TRAILER マップと対話します。プログラム A は、SET HEADER PAGE ステートメントを使用して画面を消去します。
 - プログラム B は、HEADER、DETAIL、および TRAILER という名前のプリンター・マップを含む MapGrpB を使用する、メインのバッチ・プログラムです。プログラム B は、プログラム A が端末に表示するものと同じレポートのハードコピー版を作成します。プログラム B は、部分的な HEADER マップ

を表示して、DETAIL 印刷行を浮動域に表示し、TRAILER マップをページの下部に表示します。プログラム B は、SET HEADER PAGE ステートメントを使用してページ替えを強制します。

- 浮動域の行数は、メインのトランザクション・プログラムとメインのバッチ・プログラムの間で異なります。ただし、データ検索、データ操作、および HEADER マップと DETAIL マップを表示するためのロジックは両プログラムとも同じです。このため、プログラム A とプログラム B は、共通関数を使用してデータベースからデータを検索し、データを操作し、HEADER マップと DETAIL マップを表示するように設計されています。
- VisualAge Generator 内では、同じ DISPLAY 入出力オプションを表示マップとプリンター・マップの両方に使用できるので、この共通コードの技法が有効です。また、表示マップとプリンター・マップに同じ SET HEADER PAGE ステートメントを使用できます。VisualAge Generator は、テストまたは生成している特定プログラムに基づいて、DISPLAY 入出力オプションと SET map PAGE ステートメントを解釈します。
- EGL では、表示書式と印刷書式に対して異なるステートメントが必要とされます。
 - テキスト書式の場合は **display** と **converseLib.clearScreen()**
 - 印刷書式の場合は **print** と **converseLib.pageEject()**

VisualAge Generator 互換モードでは、印刷書式に対して **display** ステートメントを使用することが許容されます。ただし、VisualAge Generator 互換モードであっても、**clearScreen()** はテキスト書式のみに適用され、**pageEject()** は印刷書式のみに適用されます。

- 一般的ではありませんが、SET-MESSAGE-TEXT という名前の共通エラー・ハンドラー関数を使用する例があります。この関数は、MSGTBLE という名前の VAGen テーブルからメッセージ・テキストを検索し、MESSAGE-TEXT という名前の関数仮パラメーターに格納します。ここで、MESSAGE-TEXT は共用データ項目です。
 - サブシステム A とサブシステム B が、異なる CICS 領域内で稼働しているとします。この場合、2 つのサブシステムが MSGTBLE の定義と、関数仮パラメーターとして使われる MESSAGE-TEXT 共有データ項目の定義を独自に提供することがあります。これは、サブシステムそれぞれのマップ定義に、異なるサイズのエラー・メッセージ・フィールドがある場合に起こります。
 - VisualAge Generator は、プログラムを生成するときに、ワークスペースに現在ロードされている定義を使用します。テストまたは生成の前に、それぞれのサブシステムが独自の MESSAGE-TEXT データ項目の定義を必ずワークスペースにロードするので、VisualAge Generator はそのサブシステムに適した定義を使用します。この技法の欠点は、生成時にワークスペース内にある定義を管理する必要があることと、両方のサブシステムが同時にワークスペース内に存在できないことです。
 - EGL の場合は、ワークスペース内に両方のサブシステムが同時に存在することが可能です。この状況では、EGL は EGL ビルド・パス、**import** ステートメント、および SET-MESSAGE-TEXT 関数の **containerContextDependent** プロパティの組み合わせを使用して、MESSAGE-TEXT DataItem パーツ定義への参照を解決します。

- 少し異なる例として、MESSAGE-TEXT2 という名前の共用データ項目を含む、ERROR-RECORD という名前の共通エラー・レコードを使用する場合があります。
- サブシステム A とサブシステム B には、異なる MESSAGE-TEXT2 の定義があるとして。この状況は、両サブシステムが異なる画面サイズ用のメッセージ・テキストを作成する必要がある場合に起こります。
- VisualAge Generator は、プログラムを生成するときに、ワークスペースに現在ロードされている定義を使用します。それぞれのサブシステムが独自の MESSAGE-TEXT2 の定義をロードする限り、VisualAge Generator はそのサブシステムに適した定義を使用します。この技法の欠点は、SET-MESSAGE-TEXT 関数の例の場合と同じです。生成時にワークスペース内にある定義を管理する必要があり、両方のサブシステムが同時にワークスペース内に存在することはできません。
- EGL の場合は、ワークスペース内に両方のサブシステムが同時に存在することが可能です。この状況では、EGL は EGL ビルド・パス、**import** ステートメント、および ERROR-RECORD の **containerContextDependent** プロパティの組み合わせを使用して、MESSAGE-TEXT2 DataItem パーツ定義への参照を解決します。

共通コードとマイグレーション・ツール

共通コードは、複数のプログラムで使用されます。共通コードはマイグレーション・ツールに次のような影響を及ぼすので、すべてのマイグレーション・セットに共通コードを組み込む必要があります。

- 共通コードが使用できれば、マイグレーション・ツールは大部分の未確定状態を解決できます。これにより、手作業で行う必要があるコードの変更が最小限あるいは不要になります。例えば、SQL レコードが、VAGen 共通ファイルに格納され、多くの異なるサブシステム・プロジェクトの関数で使用されているとします。マイグレーション・ツールが SQL 関数を変換するとき、正しく SQL 関数を変換するためには、そのマイグレーション・ツールが I/O オプションである SQL レコードを参照できなければなりません。したがって、SQL レコードを使用するサブシステム・プロジェクトをマイグレーションするときは常に、VAGen 共通プロジェクトを組み込む必要があります。
- 共通コードが使用可能であれば、マイグレーション・ツールはプロジェクトの EGL ビルド・パスを正しく設定でき、EGL ファイルに正しい **import** ステートメントを組み込むことができます。これにより、EGL ビルド・パスを変更したり、**import** ステートメントを追加する必要性が最小限になります。
- マイグレーション・ツールがパーツ・バージョンを初めてマイグレーションするとき、ツールはパーツに対して作成された EGL をデータベースに保管します。また、オリジナルの外部ソース形式もマイグレーション・データベースに保存されます。別のマイグレーション・セットが同じパーツ・バージョンを使用している場合、その別のマイグレーション・セット内で新規パーツに対する EGL を作成するときには、マイグレーション・ツールはオリジナルの外部ソース形式を参照して使用し、パーツの EGL への変換は再度行いません。その別のマイグレーション・セット用の EGL プロジェクト、パッケージ、およびファイルを作成する際にも、マイグレーション・ツールはそのパーツ・バージョン用の EGL を使用します。この技法により、パーツ間マイグレーション時に未確定状態を解決す

るために必要な参照情報がマイグレーション・ツールに提供され、さらにそれぞれのパーツ・バージョンが一度に 1 つだけマイグレーションされるのでパフォーマンスが向上します。

可能な限り最適な結果が得られるように、サブシステムをマイグレーションする際には、全社共通コードとサブシステム共通コードをマイグレーション・セットに必ず組み込んでください。

VisualAge Generator から EGL へのマイグレーション・ツールが使用する技法

技法の概要

マイグレーション・ツールは、以下の技法を使用して対応する EGL 構文を判別し、VisualAge Generator の振る舞いを保持します。

- エディターとビルド記述子の設定
- プログラム・プロパティ
- EGL のビルド・パスと **import** ステートメント
- **containerContextDependent** プロパティ
- EGL パーツ名制限事項
- EGL ファイル内でのパーツの配置
- プログラムを使用したマイグレーション
- 関連パーツを使用したマイグレーション
- 関連パーツを使用しないマイグレーション
- マイグレーション・セットの処理順序のコントロール
- ファイルの上書きとマージ

これらの技法について、次に説明します。また、マイグレーション・ツールの動作を規定する一般規則がいくつかあります。

エディターとビルド記述子の設定

マイグレーションのステージ 2 を開始する前に、使用するワークスペースの VisualAge Generator との互換性の設定をオンにします。EGL VisualAge Generator との互換性の設定により、次の VAGen の振る舞いがサポートされます。

- EGL ビルド・パスでのサイクル。つまり、ProjectB がそのビルド・パスで ProjectA を参照すると同時に、ProjectA がそのビルド・パスで ProjectB を参照できます。
- ハイフン (-) と各国語文字 @ および # を含むパーツ名の使用。ただし、これらの文字は VisualAge Generator 互換モードでも、名前の最初の文字には使用できません。
- プリミティブ・データ型 NUMC および PACF。
- 1 次元の構造化フィールド配列の場合に、添え字をデフォルトで 1 に設定します。
- DataTable の **use** 宣言に対する **deleteAfterUse** プロパティ (VAGen Keep After Use の置換表現)。

- **display printForm** ステートメントは、**print printForm** ステートメントと同じようにインプリメントされます。
- 書式フィールドの初期値は、値が割り当てられていないフィールドを画面に表示するときのみ使用されます。この設定により、ストレージ内のフィールドの初期値は設定されません。
- プリミティブ型 **DECIMAL** の項目に偶数の長さを指定すると、その項目が **WHERE** 節または **EGL prepare** ステートメントの **SQL** ホスト変数として使用される場合を除き、**EGL** はその長さを 1 だけ増やします。

EGL VisualAge Generator との互換性の設定により、**EZE** データ・ワードが次のように置換されます。

- **converseVar.segmentedMode** システム変数 (**EZESEGM** の置換表現)。
- **EZESYS** の以前の **VAGen** 値を提供する **vgLib.getVAGSysType** システム関数。
- **vgVar.sqlIsolationLevel** システム変数 (**EZESQISL** の置換表現)。

EGL VisualAge Generator 互換の設定により、**EZE** 機能語が次のように置換されます。

- **vgLib.connectionService** システム関数 (**EZECONCT** の置換表現)。

VAGen マイグレーション・ツールは、**EGL** ビルド記述子パーツにマイグレーションするすべての **VAGen** 生成オプション・パーツで、**vagCompatibility** ビルド記述子オプションを自動的に **YES** に設定します。**vagCompatibility** ビルド記述子オプションは、生成機能に対して、**VisualAge Generator** との互換性の設定と同じサポートを提供するように指示します。

注: 将来、**VisualAge Generator** 互換モードの使用を中止することを検討している場合は、マイグレーションの前に、268 ページの『**VisualAge Generator** 互換モードの使用の中止』で詳細を参照してください。例えば、ハイフン、@、または # をパーツ名から除去する必要がある場合は、マイグレーション時に名前変更ユーザー出口を使用できます。

マイグレーション・ツールの **VisualAge Generator** 互換モードの使用を最小化できるマイグレーション設定があります。例えば、マイグレーション・ツールが **vagCompatibility** = "YES" ビルド記述子オプションをあらゆる **VAGen** 生成オプション・パーツに追加しないように指定できます。詳しくは、211 ページの『**VAGen** マイグレーションの設定』を参照してください。

設定内容に関わらず、マイグレーション・ツールはワークスペースをリフレッシュするときに、**VisualAge Generator** 互換モードを常に オンにします。

プログラム・プロパティー

マイグレーション・ツールは次のプログラム・プロパティーをすべてのプログラムに組み込みます。

- **includeReferencedFunctions** = YES。関数をプログラム内でネストする必要がないように、マイグレーション・ツールはこのプログラム・プロパティーを常に組み込みます。これにより、別個のプロジェクトまたはパッケージ内で共通関数のコピーを 1 つだけ保持してインポートでき、それぞれのプログラムに共通関数を組み込まずに済みます。ステージ 1 から 3 のマイグレーションを使用する場合、

マイグレーション・ツールは、プログラムとは別のパッケージ内の関数に必要なすべての **import** ステートメントも組み込みます。

- **allowUnqualifiedItemReferences** = YES。フィールド (VAGen データ項目) の参照を修飾する必要がないように、マイグレーション・ツールはこのプログラム・プロパティを常に組み込みます。非修飾フィールドの EGL 規則は VAGen 規則に似ています。ほとんどの場合、非修飾フィールドは VisualAge Generator 内にあるものと同じレコード、DataTable (VAGen テーブル)、または書式 (VAGen マップ) に解決されます。マイグレーション・ツールは修飾を追加しません。ただし、マップ上の VAGen 非共用項目またはフィールドがプログラム、マップ、テーブルまたは関数と同じ名前である場合は、競合が発生することがあります。詳しくは、537 ページの『メッセージの参照情報 - ネーム解決規則および名前の修飾規則』を参照してください。
- **throwNrfEofExceptions** = YES。NRF (noRecordFound) と EOF (endOfFile) がエラー条件として扱われるように、マイグレーション・ツールはこのプログラム・プロパティを常に組み込みます。EGL の場合、NRF と EOF は通常はエラー条件として扱われません。このため、VAGen の振る舞いを保持するために **throwNrfEofExceptions** = yes が必要です。
- **handleHardIOErrors** = NO。vgVar.handleHardIOErrors のデフォルト値が 0 に設定されるように、マイグレーション・ツールは常にこのプログラム・プロパティを組み込みます。vgVar.handleHardIOErrors は、EZEFECE の置換表現です。EGL の vgVar.handleHardIOErrors のデフォルト値は、通常は 1 です。ただし、VAGen の EZEFECE のデフォルト値は 0 です。このため、VAGen の振る舞いを保持するために **handleHardIOErrors** = no が必要です。
- **V60ExceptionCompatibility** = YES。例外が発生する関数の外にその例外が伝搬しないように、マイグレーション・ツールは常にこのプログラム・プロパティを組み込みます。V60ExceptionCompatibility プロパティを YES に設定すると、追加的な副次作用があります。これらの副次作用のすべては、VAGen の振る舞いと整合性があります。
- **I4GLItemsNullable** = NO。NO がデフォルト値ですが、マイグレーション・ツールはこのプログラム・プロパティを常に組み込みます。この NO 値は、VAGen の振る舞いおよびパフォーマンスを保持するために指定する必要があります。
- **textLiteralDefaultIsString** = NO。テキスト・リテラルがリテラル内のデータの型によって固定長の CHAR、DBCHAR、または MBCHAR フィールドとして扱われるように、マイグレーション・ツールはこのプログラム・プロパティを常に組み込みます。これによって、EGL の生成した COBOL プログラムのパフォーマンスが向上し、さらに、テキスト・リテラルが VisualAge Generator 内で行われるように非 EGL プログラムに渡されるようになります。
- **localSQLScope** = YES。YES がデフォルト値ですが、マイグレーション・ツールはこのプログラム・プロパティを常に組み込みます。マイグレーション・ツールが結果セット ID と **prepare** ステートメント ID の作成に使用する命名規則は、プログラム間での固有性を保証しません。このため、VAGen の振る舞いを保持するために **localSQLScope** プロパティを YES に設定する必要があります。

マイグレーション・ツールはすべての DL/I プログラムまたは IMS プログラム用の @DLI complex プロパティを組み込んでおり、以下のプロパティ・フィールドを設定します。

- **psb** = *psbVariableName*。マイグレーション・ツールは、このプログラム・プロパティを組み合わせ、PSBRecord パーツ名を提供する変数の名前を指定します。マイグレーション・ツールは常に "psb" を変数名として使用します。
- **callInterface** = **DLICallInterfaceKind.CBLTDLI**。マイグレーション・ツールは、このプログラム・プロパティを組み合わせ、CBLTDLI を呼び出しインターフェースとして使うようにします。CBLTDLI は、VisualAge Generator が使用するのと同じ呼び出しインターフェースを備えています。EGL は AIBTDLI をデフォルトの呼び出しインターフェースとして使用します。AIBTDLI を使用する場合は、PCB 名情報を IMS PSB と EGL PSBRecord パーツに追加する必要があります。
- **handleHardDLIErrors** = NO。**dliVar.handleHardDLIErrors** のデフォルト値が 0 に設定されるように、マイグレーション・ツールは常にこのプログラム・プロパティを組み合わせます。**dliVar.handleHardDLIErrors** は、EZEDLERR の置換表現です。EGL の **dliVar.handleHardDLIErrors** のデフォルト値は、通常は 1 です。ただし、VAGen の EZEDERR のデフォルト値は 0 です。このため、VAGen の振る舞いを保持するために **handleHardDLIErrors** = NO を指定する必要があります。
- **psbParm** = *psbData*。VAGen プログラムが EZEDLPSB を呼び出し先パラメーターとして組み合わせる場合、マイグレーション・ツールは、PSB 全体がそのプログラムに渡されることを示すために、**psbParm** プロパティを組み合わせます。
- **pcbParms** = [*PCB* パラメーターのリスト]。VAGen プログラムが EZEDLPCB[n] (*n* は数値リテラル) を呼び出し先パラメーターとして組み合わせる場合、マイグレーション・ツールは、入力 PCB パラメーターをプログラムの PSBRecord パーツ内にある PCB にマッピングするために、**pcbParms** プロパティを組み合わせます。

UI レコードの編集ルーチンは入出力を実行する関数または別のプログラムを呼び出す関数を含んでいることがあるため、マイグレーション・ツールは、次のプロパティを常に組み込んで、VGUI レコードの作成時に VAGen の振る舞いを保持します。

- **throwNrfEofExceptions**
- **handleHardIOErrors**
- **V60ExceptionCompatibility**
- **I4GLItemsNullable**
- **textLiteralDefaultIsString**
- **localSQLScope**

EGL のビルド・パスと import ステートメント

EGL を使用すると、ワークスペース内に複数のパーツ名の定義を同時に設定できます。プロジェクトの EGL ビルド・パスにより、パーツ名を検索する際に対象となるその他のプロジェクトが限定されます。ファイル内の **import** ステートメントにより、パーツ名を検索する際に対象になる、EGL ビルド・パスに含まれるパッケージ (現行パッケージ以外) とパーツが決定されます。

ほとんどの場合、パーツ参照の解決には EGL ビルド・パスと **import** ステートメントがあれば十分です。例えば、プログラムのレコード宣言の中でレコードを型定

義として使用する場合は、EGL ビルド・パスとプログラムの **import** ステートメントがあれば、レコード名の解決には十分です。また、レコード定義、関数ローカル・ストレージ、または関数仮パラメーター・リストに対して使用できる **DataItem** パーツの定義が 1 つであれば、**DataItem** パーツ参照の解決には EGL ビルド・パスと **import** ステートメントがあれば十分です。

例えば、サブシステム A とサブシステム B を使用していて、これらに 2 つの異なる **RECORDX** の定義が存在するとします。サブシステム A 内のプログラムはすべて、**RECORDX** のサブシステム A による定義を使用する必要があります。EGL では、ビルド・パスおよび **import** ステートメントを次のように指定する必要があります。

- サブシステム A 内のプロジェクトに対する EGL ビルド・パスのプロパティに、サブシステム A の **RECORDX** の定義を提供するプロジェクトを組み込む必要があります。
- レコードの型宣言として **RECORDX** を使用するサブシステム A 内のプログラム用のファイルに、サブシステム A 内で **RECORDX** の定義を含むパッケージに **import** ステートメントを組み込む必要があります。

サブシステム A のプロジェクトに対する EGL ビルド・パスのプロパティにより、検索対象になるプロジェクトは、サブシステム A 内にあるプロジェクトと共通プロジェクトのみに限定されます。サブシステム A 内にあるファイルの **import** ステートメントにより、EGL ビルド・パスの中で検索対象になるパッケージが限定されます。**RECORDX** が **DataItem** パーツ **ITEM1** を型定義として使用する場合に、2 つのサブシステムの **ITEM1** の定義が異なっている場合、**ITEM1** への参照を解決するためには、EGL のビルド・パスと **import** ステートメントがあれば十分です。それぞれのサブシステム内の **RECORDX** を含むプロジェクトは、**ITEM1** の対応するサブシステム定義を含むサブシステム・プロジェクトを組み込んで、EGL ビルド・パスのプロパティを指定する必要があります。各サブシステム内の **RECORDX** を含むファイルには、**ITEM1** の対応するサブシステム定義を含むサブシステム・パッケージを指定する **import** ステートメントが必要です。

ステージ 1 から 3 のマイグレーションを使用する場合は、マイグレーション・セット内のパーツに基づいて、マイグレーション・ツールは次の処理を実行します。

- マイグレーション・ツールは、プロジェクトが他のプロジェクト内で参照する必要があるパーツに基づいて、それぞれのプロジェクトごとに EGL ビルド・パスを設定します。
- マイグレーション・ツールは、それぞれのファイルごとに、そのファイルを含むプロジェクトの EGL ビルド・パス内で、ファイルが参照する必要がある他のパッケージ内のパーツに基づいて、**import** ステートメントの大部分を組み込みます。これらの **import** ステートメントは、マイグレーションのステージ 1 で **VisualAge Generator** によって判別された関連パーツに基づいています。
- マイグレーション・ツールは、**DataItem** パーツに関する **import** ステートメントを追加します。ステージ 2 のマイグレーション中に、マイグレーション・ツールは **DataItem** パーツに編集ルーチンがあるかどうかを判別します。編集ルーチンとして指定されたテーブルまたは関数がマイグレーション・セットに含まれている場合、マイグレーション・ツールはマイグレーション・データベースを更新して、編集ルーチンとして指定されているパーツをデータ項目の関連パーツとして組み込みます。

- マイグレーション・ツールは、UI レコードに関する **import** ステートメントを追加します。ステージ 2 のマイグレーション中に、マイグレーション・ツールは UI レコード内のいずれかのフィールドがプログラム・リンク情報を指定しているかどうかを判別します。UI レコード内のフィールドがプログラム・リンク情報を指定していて、かつ参照先プログラムと最初の UI レコードがマイグレーション・セットに組み込まれている場合、マイグレーション・ツールはマイグレーション・データベースを更新して、参照先プログラムと最初の UI レコードを UI レコードの関連パーツとして組み込みます。
- マイグレーション・ツールは、関数に関する **import** ステートメントを追加します。ステージ 2 のマイグレーション中に、マイグレーション・ツールによって、関数内のいずれかのステートメントがテーブルを明確に参照しているかどうかを判別されます。関数内のステートメントがテーブルを参照していて、かつ参照先テーブルがマイグレーション・セットに組み込まれている場合、マイグレーション・ツールによってマイグレーション・データベースは更新され、参照先テーブルが関数の関連パーツとして組み込まれます。
- 次に示す状況では、マイグレーション・ツールは **import** ステートメントを追加しません。これは、VisualAge Generator 内ではこれらが関連パーツではないからです。
 - CALL、DXFR、または XFER のいずれかのステートメントを使用してプログラムに転送される関数。Java 用の生成を行っている場合は、その関数を含むファイル内でそのプログラムを含むパッケージの **import** ステートメントを追加するか、プログラム名をパッケージ名によって完全に修飾する必要があります。あるいは、リンケージ・オプション・パーツ内のエントリを使用して、プログラムがあるパッケージの名前を指定することも、**programPackageName** ビルド記述子オプションを使用して、生成されたすべての Java プログラムが強制的に同じランタイム・パッケージ内に配置されるようにすることもできます。
 - .eglbld ファイル内のビルド・パーツの場合。生成オプション・パーツなどの VAGen 制御パーツは関連パーツをリストしないので、マイグレーション・ツールは情報を入手できません。また、EGL がビルド記述子パーツを処理する方法が原因で、**nextBuildDescriptor** 値の再配列が必要になる可能性があります (VAGen /OPTIONS)。さらに、この再配列を行うには、マイグレーション・ツールが行ったインポートを変更する必要があります。

注: ステージ 1 マイグレーション・ツールは、マイグレーション・セット内のパーツを分析して、各パーツの関連パーツを判別します。マイグレーション・セット内のパーツのみが分析対象に含まれるように、上位 PLP プロジェクトによって指定されたマイグレーション・セットをロードする前に、ステージ 1 マイグレーション・ツールはすべての Java プロジェクトをワークスペースから削除します。同様に、上位構成マップによって指定されたマイグレーション・セットをロードする前に、ステージ 1 マイグレーション・ツールはすべての Smalltalk 構成マップを削除します。関連パーツの分析対象はマイグレーション・セットに限定されるので、マイグレーション・ツールは、マイグレーション・セットに含まれない EGL プロジェクトを指定する EGL ビルド・パス・プロパティを設定しません。また、マイグレーション・ツールは、マイグレーション・セットに含まれない EGL パッケージの **import** ステートメントを組み込みません。

ステージ 2 および 3 のマイグレーションで組み込まれる次の処理は、単一ファイル・モードのマイグレーションでは組み込まれません。

- EGL ビルド・パスの設定。これは、単一ファイルのマイグレーションではすべてのパーツが同じプロジェクト内に配置されるためです。
- **import** ステートメントの組み込み。これは、単一ファイルのマイグレーションではすべてのパーツが同じパッケージ内に配置されるためです。

containerContextDependent プロパティ

注: ここで説明するのは、EGL バージョン 7.1 に部分的にのみインプリメントされている機能です。関数に **containerContextDependent** プロパティを指定すると、その関数内での関数呼び出しの解決は（開発時でなく）生成時に行われ、呼び出し側関数を使用するプログラムのネーム・スペースへの参照を組み込みます。現時点では、レコード・パーツ名または **DataItem** パーツ名の解決に対して **containerContextDependent** プロパティの効果はありません。

以降の説明は、最終インプリメンテーションとして意図されている内容を反映しています。

45 ページの『EGL のビルド・パスと **import** ステートメント』で説明したとおり、一般には EGL ビルド・パスと **import** ステートメントがあれば、必要なパーツ名の解決には十分です。ただし EGL は、ファイルを保管するたびにパーツ名参照すべての解決を期待します。パーツ名を解決できない場合、EGL は「問題」ビューにエラー・メッセージを追加します。アーキテクチャーによっては、レコードまたは関数の **containerContextDependent** プロパティを使用する必要があることもあります。

RECORDX が、FUNCTIONY 内で関数仮パラメーターの型定義として使用されるという状況を考えます。RECORDX と FUNCTIONY が異なるプロジェクトとパッケージの中にあるとすると、EGL は次の情報を期待します。

- FUNCTIONY を含むプロジェクトの EGL ビルド・パスには、RECORDX の定義を含むプロジェクトが組み込まれている。
- FUNCTIONY を含むファイルには、RECORDX を含むパッケージの **import** ステートメントが組み込まれている。

すべてのサブシステムにある RECORDX の定義が同じであれば、EGL ビルド・パスと **import** ステートメントがあれば十分であり、EGL は FUNCTIONY を含むファイルを保管するときにいつでも RECORDX のパーツ参照を解決できます。

一方で、サブシステム A とサブシステム B が両方とも FUNCTIONY を使用し、ただし異なる RECORDX の定義を使用するという状況を考えましょう。この状況では、EGL ビルド・パスと **import** ステートメントは両方のサブシステムを同時に指定できません。EGL は、関数の **containerContextDependent** プロパティをサポートします。この状況では、FUNCTIONY に対して **containerContextDependent** プロパティを YES に設定できます。このプロパティは、関数仮パラメーターとローカル・ストレージのパーツ名参照が、プログラム内で FUNCTIONY が使用されるまで解決されないように指定します。FUNCTIONY を使用するプログラムをテストまたは生成するとき、プログラムを含むプロジェクトの EGL ビルド・パスと、プログラムを含むファイルの **import** ステートメントによって、RECORDX の

定義を検索する場所が決まります。**containerContextDependent** プロパティを YES に設定すれば、関数に対して VisualAge Generator の機能と同じ柔軟性が得られます。サブシステム内の各プロジェクトの EGL ビルド・パスと、サブシステム内のプログラムを含むファイルの **import** ステートメントが、そのサブシステムの RECORDX の定義を指示します。

containerContextDependent プロパティは、レコード・パーツにも対応しています。例えば、SubsystemA と SubsystemB が両方とも RECORDZ の同じ定義を使用しているとします。ただし、RECORDZ は ITEM1 という名前の DataItem パーツを参照する型定義を使用します。各サブシステムの ITEM1 の定義は異なります。この場合は、RECORDZ に対して **containerContextDependent** = yes を指定すれば、RECORDZ がプログラム内で使用されるまで、EGL の検証機能は ITEM1 の解決を試行しません。プログラムを含むプロジェクトの EGL ビルド・パスと、プログラムを含むファイルの **import** ステートメントによって、ITEM1 の定義を検索する場所が決まります。

マイグレーション・ツールは、ユーザー用の **containerContextDependent** プロパティを設定しません。これは、マイグレーション・ツールが必ずしもすべてのサブシステムを同時にマイグレーションするとは限らず、重複するパーツ定義があることを判別するために全パーツの定義すべてを完全に分析しないからです。定義時 (EGL の場合のように) でなく、テストおよび生成時 (VisualAge Generator の場合のように) に解決する必要がある重複パーツ名があるとユーザーが判断した場合に、必要に応じて **containerContextDependent** プロパティを追加できます。

EGL パーツ名制限事項

EGL のパーツおよび変数名には、VAGen のパーツおよびフィールド名よりも多くの制限事項があります。EGL の制限事項は次のとおりです。

- EGL には予約語リストがあります。EGL パーツおよび変数には、EGL 予約語とは異なる名前を付ける必要があります。
- VisualAge Generator 互換の設定が選択されている場合でも、EGL では EGL パーツ名の最初の文字として # または @ の記号を使用することができません。
- EGL パーツ名または変数名が EGL プロパティ、注釈、列挙、またはライブラリーの名前と同じである場合、EGL ネーム・レゾリューションではパーツ名または変数名が優先されます。

競合を最小限にとどめるために、マイグレーション・ツールは、すべての EGL プロパティ、注釈、列挙、およびライブラリーの名前のほか、すべての EGL 予約語を組み込んだ拡張予約語リストを作成します。VAGen パーツ名またはフィールド名がこの拡張予約語リストにあるか、# または @ 記号で始まる場合、マイグレーション・ツールは次の規則を使用してそのパーツ名またはフィールド名をパーツ型に応じて変更します。

- マイグレーション・ツールは、プログラム、マップ・グループ、またはテーブルの名前を変更しません。これは、これらのパーツが、VAGen 以外のプログラム、またはランタイム環境を参照する場合が多いからです (例えば、CICS PROGRAM 定義)。
- マイグレーション・ツールは、パーツ名に名前変更接頭部を付けることによって、データ項目、レコード、マップ、および関数の名前を変更します。名前変更

接頭部は、ステージ 2 または単一ファイル・モードのマイグレーションに関して指定できる、VAGen マイグレーション設定の 1 つです。同様に、マイグレーション・ツールは、接頭部に**名前変更接頭部**を付けてフィールド名を変更します。

注: 名前変更について、マイグレーション・ツールは UI レコードを他のレコードと同じに扱います。さらに、UI レコードの名前を変更する必要がある場合は、ステージ 3 マイグレーション・ツールはその UI レコードを含む .egl ファイルの名前を変更して UI レコードの変更後の名前と一致するようにします。マイグレーション・ツールは、VGUI レコードの **alias** プロパティを元の VAGen UI レコード名に設定し、EGL が生成したレコード名を元の VAGen UI レコードに一致させます。

- マイグレーション・ツールは、次の場合を除いて制御パーツの名前を変更しません。
 - マイグレーション・ツールは、バインド制御パーツ名の末尾から .BND 接尾部を除去します。
 - マイグレーション・ツールは、リンク・エディット・パーツ名の末尾から .LKG 接尾部を除去します。
 - マイグレーション・ツールは、制御パーツ名に含まれるその他のドットを下線に変更します。さらにこのツールは、/OPTIONS、/RESOURCE、および /LINKEDIT 生成オプション内で参照されている制御パーツ名のドットを下線に変更します。

ステージ 1 マイグレーション・ツールには、マイグレーション・ツール拡張予約語リストと競合するプログラム名、マップ・グループ名、テーブル名、および制御パーツ名のリストがあります。マイグレーション前にこれらのパーツの名前を変更しなければ、ステージ 2 マイグレーション・ツール (または単一ファイル・モード) からもエラー・メッセージが出されます。EGL 検証の結果、「問題」ビューにもエラー・メッセージが表示されます。プログラム、FormGroup または DataTable の名前を変更し、オプションで EGL **alias** プロパティを使用することで、EGL の問題を訂正できます。

注: ステージ 2 マイグレーション・ツールは、そのツールで名前変更される UI レコードすべてについて警告メッセージを発行します。ステージ 3 マイグレーション・ツールも .egl ファイル名を変更するため、UI レコードの「問題」ビューにエラーは表示されません。

EGL ファイル内でのパーツの配置

ステージ 1 から 3 のマイグレーションを使用してマイグレーションを行う場合、それぞれの Java パッケージまたは Smalltalk アプリケーションは、ステージ 1 の名前変更規則に基づいて、対応する EGL パッケージにマイグレーションされます。オリジナルの Java パッケージまたは Smalltalk アプリケーション内の VAGen パーツは、次の条件に基づいて、対応する EGL パッケージ内で 1 つ以上の EGL ファイルに配置されます。

- パーツのタイプ:
 - 生成可能パーツ (プログラム、テーブル、マップ・グループ、または UI レコード)

- 制御パーツ (生成オプション、リソース関連、リンケージ・テーブル、リンク・エディット、またはバインド制御)
- その他のマイグレーション可能パーツ (データ項目、マップ、関数、PSB、および UI レコード以外のレコード)
- 共通パーツを含む Java プロジェクトまたはパッケージ名の識別を可能にするステージ 1 設定。同様に、共通パーツを含む構成マップまたはアプリケーション名の識別を可能にする、Smalltalk 用の設定もステージ 1 にあります。
- パーツが他のパーツによって使用されているかどうか。ステージ 1 マイグレーション・ツールは、次の条件に基づいてパーツが使用されているかどうかを判断します。
 - パーツがマイグレーション・セット内の生成可能パーツの VAGen 関連リストにあれば、パーツは「使用されている」。
 - ステージ 1 設定に指定されているとおり、パーツが共通 Java プロジェクトまたはパッケージ内、または共通 Smalltalk 構成マップまたはアプリケーション内にあれば、パーツは「使用されている」。

ステージ 1 マイグレーション・ツールは、すべてのパーツの配置を決定します。ステージ 1 マイグレーション・ツールは、単一の Java パッケージまたは Smalltalk アプリケーション内にある VAGen パーツを、対応する EGL パッケージ内の EGL ファイルに、次の規則に従って配置します。

- すべての制御パーツが、*eglPackageName.eglbld* という名前の単一ファイルに配置されます (ここで、*eglPackageName* は対応する EGL パッケージの名前)。
- それぞれのプログラム・パーツは、*programName.egl* という名前のファイルに格納されます (ここで、*programName* はプログラムの名前)。
- それぞれのテーブル・パーツは、*tableName.egl* という名前のファイルに格納されます (ここで、*tableName* はテーブルの名前)。
- それぞれのマップ・グループと、マップ・グループ内のすべてのマップは、*mapGroupName.egl* という名前のファイルに格納されます (ここで、*mapGroupName* はマップ・グループの名前)。マップ・グループ・パーツがない場合、ステージ 1 マイグレーション・ツールはダミーのマップ・グループ・パーツを作成します。マップ・グループと、そのマップ・グループ内にあるマップはすべて同じファイルに配置する必要があるため、これらのパーツはグループとして考える必要があります。このため、一部のパーツが元は同じ Java パッケージまたは Smalltalk アプリケーション内になかった場合、そのパーツは異なる EGL パッケージまたはプロジェクトに移動されることがあります。マイグレーション・ツールは、*mapGroupName.egl* ファイルを配置する場所を、次の規則に従って決定します。
 - マップ・グループとそのマップすべてが同じ Java パッケージ内にある場合、*mapGroupName.egl* ファイルは対応する EGL パッケージに配置されます。同様に、マップ・グループとそのマップすべてが同じ Smalltalk アプリケーション内にある場合、*mapGroupName.egl* ファイルはその Smalltalk アプリケーションに対応する EGL パッケージに配置されます。この場合、マイグレーション・ツールはプログラムやテーブル・ファイルと同じ方法で *mapGroupName.egl* ファイルを処理します。これは最もよく起こる状態です。
 - マップ・グループとそのマップが、1 つのプロジェクト内で複数の Java パッケージに分散している場合は、プロジェクト名に接尾部を付加した名前が、

mapGroupName.egl ファイルを格納する新規 EGL パッケージ名の作成に使用されます。この新規 EGL パッケージは、オリジナルのプロジェクト内に配置されます。同様に、マップ・グループとそのマップが 1 つの構成マップ内で複数の Smalltalk アプリケーションに分散している場合は、構成マップ名に接尾部を付加した名前が、*mapGroupName.egl* ファイルを格納する新規 EGL パッケージ名の作成に使用されます。Java と Smalltalk のどちらの場合も、ステージ 1 設定を使用して接尾部を制御できます。

- マップ・グループとそのマップが、複数の Java プロジェクトに分散している場合は、マイグレーション・セット名に接尾部を付加した名前が、*mapGroupName.egl* ファイルを格納する新規 EGL プロジェクト名の作成に使用されます。同様に、マップ・グループとそのマップが複数の Smalltalk 構成マップに分散している場合は、マイグレーション・セット名に接尾部を付加した名前が、*mapGroupName.egl* ファイルを格納する新規 EGL プロジェクト名の作成に使用されます。Java と Smalltalk のどちらの場合も、ステージ 1 設定を使用して接尾部を制御できます。
- それぞれの UI レコードは、*uiRecordName.egl* という名前のファイルに配置されます (ここで、*uiRecordName* は UI レコードの名前)。
- その他のパーツはすべて、次の規則に従って配置されます。
 - マイグレーション・セット内の**ただ 1 つのプログラムによって使用されている**パーツは、次の規則に従ってファイル内に配置されます。
 - パーツがプログラムと同じパッケージ内にある場合、パーツはプログラムと同じファイルに配置されます。例えば、プログラムの `main` 関数 (`ProgramA-MAIN`) は、その関数が他のプログラムまたは他の生成可能パーツ内で使用されていなければ、プログラム (`ProgramA`) と同じファイルに配置されます。このファイルは、プログラムに合わせて `ProgramA.egl` という名前になります。
 - パーツがそのパーツを使用するプログラムとは異なるパッケージ内にある場合、パーツは、そのパーツのオリジナル・パッケージ内のファイルに配置されます。このファイルは、デフォルトでは `commonParts.egl` と命名されますが、共通パーツ・ステージ 1 の設定によって名前を変更することができます。
 - マイグレーション・セット内の**複数のプログラムまたは複数の生成可能パーツによって使用されている**パーツは、オリジナル・パッケージ内で `commonParts.egl` という名前のファイルに配置されます。例えば、`ProgramA` が `ProgramB` を呼び出して `RecordR` を渡す場合、`RecordR` が配置される先のファイルは、`RecordR` を含むオリジナルの Java パッケージまたは Smalltalk アプリケーションに対応する EGL パッケージ内にある、`commonParts.egl` という名前のファイルです。
 - マイグレーション・セット内のプログラムによって**使用されていない**パーツは、次の規則に従ってファイルに配置されます。
 - パーツが共通 Java プロジェクトまたはパッケージ内にある場合は、そのパーツを含むオリジナル Java パッケージに対応する EGL パッケージ内で、`commonParts.egl` という名前のファイルにパーツが配置されます。同様に、パーツが共通 Smalltalk 構成マップまたはアプリケーション内にある場合

は、そのパーツを含むオリジナル Smalltalk アプリケーションに対応する EGL パッケージ内で、commonParts.egl という名前のファイルにパーツが配置されます。

- 共通 Java プロジェクトまたはパッケージ内に存在しないパーツは、そのパーツを含むオリジナル Java パッケージに対応する EGL パッケージ内のファイルに配置されます。同様に、共通 Smalltalk 構成マップまたはアプリケーション内に存在しないパーツは、そのパーツを含むオリジナル Smalltalk アプリケーションに対応する EGL パッケージ内のファイルに配置されます。このファイルは、デフォルトでは unusedParts.egl と命名されますが、未使用パーツ・ステージ 1 の設定によって名前を変更することができます。
- 次の特別な考慮事項が該当します。
 - マイグレーション・ツールによって、編集ルーチンとして使用される関数が commonParts.egl ファイル内のマップ、UI レコード、またはデータ項目パーツに配置されます。また、マイグレーション・ツールは commonParts.egl ファイル内の編集ルーチン関数の関連パーツをすべて配置します。この技法によって、プログラム以外のパーツが使用する編集ルーチン関数を、確実にプログラム・ファイルの外部で表示できるようになります。プログラム内の関数を後でネストすることにした場合、編集ルーチンとして使用される関数の可視性を気にせずに、プログラム・ファイル内の最後の関数の後にプログラムの **end** ステートメントを移動することができます。
 - マイグレーション・ツールは、テーブル内や UI レコードで使用されるすべての共用項目を commonParts.egl ファイルに配置します。マイグレーション・ツールは、テーブルまたは UI レコードと同じファイル内に共用項目を配置しません。

注: マイグレーション・データベースを消去せずに、複数のマイグレーション・セット、または複数バージョンのマイグレーション・セットをマイグレーションした場合は、いずれかのパーツ・エディションを含む、ステージ 1 で最初に処理されたマイグレーション・セットのバージョンによって、EGL パーツのプロジェクト、パッケージ、およびファイル名が制御されます。マイグレーション・セットのバージョンごとの定義に従ってパーツが配置されるようにするには、バージョンが変わるたびにマイグレーション・データベースを消去する必要があります。また、ステージ 1 における最新バージョンのマイグレーション・セットをマイグレーションすることにより、前バージョンのマイグレーション・セット以降変化していないすべてのパーツ・エディションが、そのパーツが現在使用されているかどうかを基準にして EGL ファイルに配置されるようにします。例えば、次のようになります。

- 以前は 1 つのプログラムだけが使用していたパーツを、複数のプログラムが使用するようになっていくとします。この場合、最新バージョンのマイグレーション・セットを最初にマイグレーションすると、パーツは、そのパーツの元の (そして唯一の) ユーザーであったプログラムではなく、共通パーツ EGL ファイル内に配置されます。
- 以前はどのプログラムも使用していなかったパーツを、現在は 1 つ以上のプログラムが使用しているとします。最新バージョンのマイグレーション・セットを最初にマイグレーションすると、パーツは、unusedParts.egl ファイル内ではなく、そのパーツの現在の使用状況に応じて配置されます。

Java および Smalltalk 用のステージ 1 マイグレーション・ツールは、サンプル・コードとして提供されています。お客様独自のライブラリー管理方針に基づいてパーツを配置するように、ステージ 1 マイグレーション・ツールを変更できます。例えば、次のようになります。

- ProgramX が ProgramY を呼び出して、レコード ProgramY-Parm と Common-Parm を渡す場合は、ProgramY-Parm を ProgramY と一緒のファイルに配置し、Common-Parm を commonParts ファイルに配置できます。命名規則が分かっている場合は、ステージ 1 マイグレーション・ツールを修正してファイル配置アルゴリズムを変更できます。
- 大規模なパッケージの場合、パーツをパーツ型別、またはパーツ名の最初の数文字別に複数ファイルに分割できます。
- 同じパーツ・エディションがマイグレーション・セットの複数バージョンに存在し、ただしマイグレーション・セットのバージョンに応じて、このパーツ・エディションを異なる EGL プロジェクト、パッケージ、またはファイルに配置する必要がある場合は、マイグレーション・セットのいずれかのバージョンを処理するたびに、パーツごとの新しい EGL プロジェクト、パッケージ、およびファイル名に応じてマイグレーション・データベースを更新できます。この変更を行う場合は、それぞれのマイグレーション・セット・バージョンをステージ 1 から 3 まで完全に処理してから、次のマイグレーション・セット・バージョンのマイグレーションを開始するようにしてください。
- Java パッケージごとに 1 つのプログラム、そしてプロジェクトごとに 1 つのパッケージを配置する方針をとっていた場合、複数の Java パッケージまたはプロジェクトを結合して、ソース・コード・リポジトリで管理する必要のある EGL パッケージおよびプロジェクトの数を減らすことができます。同様にまた、Smalltalk アプリケーションごとに 1 つのプログラム、そして構成マップごとに 1 つのアプリケーションを配置する方針をとっていた場合、EGL プロジェクトおよびパッケージを作成するとき複数の Smalltalk アプリケーションまたは構成マップを結合することができます。

ステージ 1 ツールの変更の例については、19 ページの『参照』にリストされた、ステージ 1 のホワイト・ペーパーを参照してください。さらに、ステージ 1 ツールには、標準装備のカスタマイズ機能がいくつかあります。Java のステージ 1 ツールの詳細は、164 ページの『ステージ 1 マイグレーション・ツールのカスタマイズ』を参照してください。Smalltalk のステージ 1 ツールの詳細は、192 ページの『ステージ 1 マイグレーション・ツールのカスタマイズ』を参照してください。

プログラムを使用したマイグレーション

通常、マイグレーションの際には、グループとしてマイグレーションする必要があるすべての Java プロジェクトまたは Smalltalk 構成マップを識別するマイグレーション・セットを指定します。マイグレーション・ツールは、マイグレーション・セットを使用して、プログラムとその関連パーツを最初にマイグレーションします。これによって、EGL 言語が VisualAge Generator よりも厳密または制限的である場合に、ツールがその状態を解決するための支援情報として特定プログラムのコンテキストを使用できます。最初にマイグレーションするプログラムとその関連パーツによって、そのプログラムまたは関連パーツの中で未確定状態に対応する EGL 構文が決まります。プログラムが異なると、共用データ項目、共用レコード、マップ、テーブル、または関数の同じ未確定状態に対して、異なる解決結果が得られる

場合があります。パーツ・バージョンのマイグレーションは一度限り行われるので、共通パーツを最初に使用するプログラムが、その関連パーツの未確定状態の解決を制御します。

例として、ProgramA が表示マップを使用するメインのトランザクション・プログラムであり、ProgramB がプリンター・マップを使用するメインのバッチ・プログラムである場合を考えます。これらのプログラムは、HEADER マップと DETAIL マップを表示する共通の関数を共有します。共通関数はまた、SET map PAGE ステートメントを使用して、画面のクリアまたは強制的なページ替えを行います。この例では、ProgramA を先にマイグレーションする場合、マイグレーション・ツールは、**display** ステートメントと **converseLib.clearScreen()** システム・ライブラリー関数を使用する EGL ソースを関数用に作成します。ProgramB を先にマイグレーションする場合、マイグレーション・ツールは、**print** ステートメントと **converseLib.pageEject()** システム・ライブラリー関数を使用する EGL ソースを作成します。

プログラムとその関連パーツをマイグレーションするときには必ず、共用データ項目、共通レコード、マップ、テーブル、または関数を最初に使用するプログラムが、EGL コードの生成結果を制御します。ほとんどの場合、プログラムは共通コードを同じ方法で使用するので、この技法によって VAGen ソースの最適なマイグレーションが実現します。ただし、この例から分かるように、共通コードの実行内容として意図された特性が、結果の EGL ソースには反映されない場合があります。この例では、どのプログラムを最初にマイグレーションするかに関係なく、2 番目にマイグレーションするプログラムをテストしたり生成したりすることはできません。VisualAge Generator 互換モードでは、**display** ステートメントを使用して、入出力ステートメントに関する問題を解決できます。ただし、**clearScreen()** または **pageEject()** を選択することで問題を解決する場合には、新しい変数 TEXT-OR-PRINT を追加する必要があります。それぞれのプログラムがこの変数を初期化し、共通関数がこの変数をテストして、**clearScreen()** または **pageEject()** システム・ライブラリー関数を実行するかどうかを判別します。

関連パーツを使用したマイグレーション

プログラムと関連パーツが使用不可の場合、マイグレーション・ツールはマイグレーション・セット（また、単一ファイル・モードでマイグレーションしている場合には、外部ソース形式ファイル）内で使用可能なすべてのパーツを使用します。この場合、パーツ間マイグレーションに必要な追加パーツが使用可能ならば、マイグレーション・ツールは正しい選択肢を高い確率で判断できます。

例えば、マップ変数フィールドが編集ルーチンを指定している場合を考えます。編集ルーチンと同じ名前の VAGen テーブルが同じマイグレーション・セット（または外部ソース形式ファイル）内にあれば、マイグレーション・ツールはこれが常に使用されるテーブルであると想定して、**validatorDataTable** プロパティにマイグレーションします。編集ルーチンと同じ名前の関数が存在すれば、マイグレーション・ツールは **validatorFunction** プロパティにマイグレーションします。どちらの場合も、編集ルーチンと同じ名前のパーツが存在するので、マイグレーション・ツールが正しい選択を行った確率は高くなります。編集ルーチンと同じ名前のテーブルまたは関数がなければ、マイグレーション・ツールは、関連パーツを使用せずにマイグレーションを行う場合と同じようにマップ変数フィールドを処理します。

多くのケースでは、関連パーツを使用してマイグレーションを行えば、プログラムを関連パーツとともにマイグレーションした場合とほぼ同様のマイグレーションを実現できます。プログラムを使用しないマイグレーションの欠点は、単一の関数内であっても、マイグレーション対象である特定のステートメント、およびマイグレーション・セットに含まれているその他のパーツに応じて、関連パーツを使用したマイグレーションから、関連パーツを使用しないマイグレーションに切り替わりやすいことです。

関連パーツを使用しないマイグレーション

場合によっては、プログラムが使用可能であっても、その関連パーツの一部がマイグレーション・セットに含まれていないことがあります。あるいは、マイグレーションする共通パーツが、過去にサブシステムによって使用されていたものの、現在は使用されていない場合もあります。このような場合、マイグレーション対象のパーツに関連するものが使用できなくなる可能性があります。それでもマイグレーション・ツールは、次のいずれかの技法を使用してパーツを変換します。

- **EGL 構文の柔軟性。**例えば、関連マップがない状態で **DISPLAY** 入出力オプションをマイグレーションするとします。この場合、マイグレーション・ツールは **display** ステートメントの使用を選択し、マイグレーション・ログに警告メッセージを書き込みます。マイグレーション・ツールの予測が誤っていたとしても、VisualAge Generator 互換モードを使用していれば、書式が印刷書式であっても **display** ステートメントは受け入れられます。
- **インテリジェント予測。**例えば、マップ変数フィールドで編集ルーチンが指定されているが、その編集ルーチンと同じ名前のパーツがマイグレーション・セットに存在しないとします。この場合、マイグレーション・ツールは他の情報を使用します。ツールは、**validatorDataTable** プロパティと **validatorFunction** プロパティのどちらを使用するかを判別する際に、以下の要因を考慮します。
 - 編集ルーチン名の長さ。8 文字以上ならば、編集ルーチンが関数であることを示しています。
 - 編集ルーチン名が **EZEC10** または **EZEC11** である。これは、編集ルーチンが関数であることを示しています。
 - 編集メッセージも指定されている。メッセージは、編集テーブル、**EZEC10**、または **EZEC11** に対してのみ使用できます。

これらの要因のいずれかが存在していると、マイグレーション・ツールは、**validatorDataTable** または **validatorFunction** のどちらのプロパティを設定するか、より良い選択ができます。決定的な選択要因がなければ、マイグレーション・ツールは **validatorFunction** プロパティを使用し、エラー・メッセージをマイグレーション・ログに書き込みます。マイグレーション・ツールの予測が誤っている場合は、「問題」ビューにもエラーが示されます。

- **意図的な無効構文。**例えば、関連マップがない状態で **SET map PAGE** をマイグレーションするとします。この場合、マイグレーション・ツールはテキスト書式用の **EGL converseLib.clearScreen()** 関数を使用するか、印刷書式用の **EGL converseLib.pageEject()** 関数を使用するかを選択できます。ただし、どちらの選択項目も等しくありえます。このため、マイグレーション・ツールは意図的に誤った構文を作成し、**converseLib.EZE_SETPAGE** に変換します。この結果、「問題」ビューでエラーが発生するので、問題の訂正が必要であることが分かります。

- 関連パーツの欠落による、情報が無い状態での直接変換。(関連パーツが欠落していると、マイグレーション・ツールが検出できない問題が生じる可能性があります。) 例えば、RecordA が RecordB のストレージを再定義するように指定しているとします。VisualAge Generator の場合、再定義情報は RecordA のレコード定義に格納されています。生成時には、RecordA と RecordB が使用可能であることが必要で、RecordA の再定義がプログラム内で行われます。EGL の場合、再定義情報はプログラムのみに格納されます。プログラムのマイグレーション時に RecordA が使用できなければ、RecordA がプログラム内に **redefines** プロパティを組み込む必要があるということを、マイグレーション・ツールが検出する手段はありません。**redefines** プロパティがなければ、EGL のデバッグおよび生成機能は、RecordA と RecordB を別個のデータ域として扱います。プログラムは、VisualAge Generator の場合とは異なる動作をします。データが正しく初期化されず、異常終了が発生する可能性があります。このため、マイグレーション済みのプログラムを生成し、テストすることを強くお勧めします。

マイグレーション・セットの処理順序のコントロール

ステージ 1 マイグレーション・ツールでは以下の順序でマイグレーション・セットを処理します。

- .pln ファイルまたは .pln ファイルを含むディレクトリーを指定する場合、ステージ 1 マイグレーション・ツールでは、単一の .pln ファイル内でリストされている順序でマイグレーション・セットが処理されます。ディレクトリー内に複数の .pln ファイルがある場合、ステージ 1 マイグレーション・ツールではそれらのファイルはアルファベット順で処理されます。
- .pln ファイルまたは .pln ファイルを含むディレクトリーを指定しない場合、ステージ 1 マイグレーション・ツールでは、仕様に一致する Java の上位 PLP プロジェクトがアルファベット順に処理されます。同じ Java 上位 PLP プロジェクトの複数のバージョンが要求されている場合、ステージ 1 マイグレーション・ツールでは、それらのバージョンが日付/タイム・スタンプに基づいた順序で処理されます。同様に、ステージ 1 マイグレーション・ツールでは、仕様に一致する Smalltalk の上位構成マップがアルファベット順に処理されます。同じ Smalltalk の上位構成マップの複数のバージョンが要求されている場合、ステージ 1 マイグレーション・ツールでは、それらのバージョンが日付/タイム・スタンプに基づいた順序で処理されます。
- マイグレーション・セットがマイグレーション・データベースに追加される順序をさらにコントロールする必要がある場合、ステージ 1 ツールを複数回実行します。

ステージ 2 および 3 のマイグレーション・ツールでは、以下の順序でマイグレーション・セットを処理します。

- ステージ 2 および 3 マイグレーション・ツールでは、.vgmig ファイルでリストされているのと同じ順序でマイグレーション・セットが処理されます。デフォルトですべてのマイグレーション・セット名が固有である場合、この順序は VAGen マイグレーション・ウィザードでマイグレーション・セットがリストされる順序と同じであり、マイグレーション・セットがステージ 1 のマイグレーション・データベースに追加された順序と同じです。
- 順序を変更する必要がある場合、「今すぐマイグレーションを実行」を選択解除して .vgmig ファイルを保存します。.vgmig ファイルをダブルクリックして順序

を変更し、ファイルを保存します。 .vgmig ファイルを右クリックして、「**マイグレーションの開始**」をクリックします。あるいはステージ 2 および 3 を複数回実行し、実行するたびに、マイグレーションを実行する順序になるように 1 つのマイグレーション・セットのみを指定します。

注: オンライン・モードではマイグレーション・セットの複数のバージョンを処理することはできません。

ファイルの上書きとマージ

ステージ 2 および 3 のマイグレーション・ウィザードは、同じマイグレーション・セットの複数バージョンの処理を制御する、次の関連設定を行います。

- 残りの VAGen パーツをマイグレーションする
- ワークスペースにインポート (「既存ファイルを上書き」の指定あり、または指定なし)
- マイグレーションしたファイルを一時ディレクトリーに保管

「**残りの VAGen パーツをマイグレーションする**」は、マイグレーション・ツールがマイグレーション・セット内のパーツすべてを EGL に変換するかどうかを制御します。

- **残りの VAGen パーツをマイグレーションする** 設定については、UI レコードは他のレコードと同じように扱われます。この設定では、UI レコードを生成可能パーツとはみなされません。
- 「**残りの VAGen パーツをマイグレーションする**」を選択しない場合は、生成可能パーツとその関連パーツのみが EGL に変換され、マイグレーション・データベースに保管されます。データ項目、レコード、および関数は、1 つ以上の生成可能パーツに関連していない限りは変換されません。制御パーツは変換されません。単一のマイグレーション・セット内の 1 つのサブシステム・プロジェクトおよび 1 つの共通プロジェクトをマイグレーションしている場合は、「**残りの VAGen パーツをマイグレーションする**」の設定を選択解除しておくとう良い場合があります。この場合、マイグレーション・ツールは、次のようにプロジェクトをマイグレーションします。
 - サブシステム・プロジェクトの場合は、サブシステム内で実際に使用されているパーツのみが変換されます。
 - 共通プロジェクトの場合は、生成可能パーツとその関連パーツが変換されます。また、サブシステムによって使用されているデータ項目、レコード、および関数も変換されます。それ以外の、他のサブシステムに使用されていても現行サブシステムに使用されていないデータ項目、レコード、および関数は、EGL に変換されません。

「**残りの VAGen パーツをマイグレーションする**」を選択解除する場合の利点は、次の 2 つです。

- サブシステム・プロジェクトの場合は、実際に使用されているパーツのみがマイグレーション・ツールによって変換されるので、コードをクリーンアップする機会になります。
- 共通プロジェクトの場合は、パーツが別のサブシステムによって実際に使用されるまで、パーツの変換を遅らせることができます。別のサブシステム用のマイグレーション・セットに共通プロジェクトを組み込むと、このサブシステム

によって使用されている追加パーツが EGL に変換され、マイグレーション・データベースに格納されます。この機能は特に、共通プロジェクトがさまざまなサブシステムの関連パーツを含んでいる場合や、さまざまなサブシステム内の生成可能パーツに関連するパーツを含んでいる場合に便利です。サブシステムがパーツを使用するまで共通パーツのマイグレーションを遅らせることにより、「関連パーツを使用して」共通パーツをマイグレーションできます。共通プロジェクトを含む次のマイグレーション・セットをマイグレーションするときに、マイグレーション元の共通パーツ・ファイルに対して行われる処理は、「既存ファイルを上書き」の選択項目によって制御されます。

- 「残りの VAGen パーツをマイグレーションする」設定を選択した場合は、生成可能パーツとその関連パーツが EGL に変換され、マイグレーション・データベースに保管されます。その後、生成可能パーツに関連していないデータ項目、レコード、および関数すべてが EGL に変換されます。また、制御パーツもすべて EGL に変換されます。「残りの VAGen パーツをマイグレーションする」を選択する場合の利点は、次の 2 つです。
 - サブシステム・プロジェクトの場合は、使用されているかいないかに関係なく、すべてのパーツが EGL に変換されます。このことは、コードを履歴として保持しておく必要がある場合に便利です。
 - 共通プロジェクトの場合は、将来マイグレーションする予定のサブシステム・プロジェクト内に、共通プロジェクト内のパーツに関連するものがないことが分かっている場合に、「残りの VAGen パーツをマイグレーションする」の選択が特に役立ちます。すべての共通パーツを一度に変換でき、マイグレーション・データベースに EGL を格納できます。その後、共通プロジェクトが他のサブシステム用のマイグレーション・セットに含まれている場合に、EGL はすでに変換済みであり、ワークスペースにインポートしたり、新規サブシステムの一時的ディレクトリーに保管したりできます。

マイグレーション・セットの最初のバージョンに対して「残りの VAGen パーツをマイグレーションする」を選択した場合は、マイグレーション・セットの他のバージョンに対しても、引き続き「残りの VAGen パーツをマイグレーションする」を選択する必要があります。さらに、「既存ファイルを上書き」を選択する必要があります。これら両方のオプションを選択することにより、マイグレーション・セット内のすべてのパーツが EGL ファイルに含まれるようになります。

「ワークスペースにインポート」設定は、マイグレーション・ツールが EGL プロジェクト、パッケージ、およびファイルをワークスペース内にビルドするかどうかを制御します。「ワークスペースにインポート」を選択すると、選択可能なその他のオプションが示されます。

- マイグレーション・セットの複数バージョンをマイグレーションする場合は、マイグレーションの終了時に、ワークスペースにインポートするバージョンを選択できます。「最新バージョン」(最近のバージョン)、または「最も古いバージョン」のどちらかを選択できます。最新バージョンを選択する場合の利点は、このバージョンが後続のテスト用に生成したいものである可能性が最も高いことです。最も古いバージョンを選択する場合の利点は、最も古いバージョンに対応する EGL プロジェクト、パッケージ、およびファイルが、ソース・コード・リポジトリに最初に格納されるようになることです。
- 現行マイグレーションによって作成されている EGL ファイルがワークスペースにすでに存在する場合に、その状態にどのように対処するか指定できます。

- 「**既存ファイルを上書き**」設定を選択すると、EGL ファイルは、現行マイグレーション・セット内のパーツのみを含む新規ファイルに置き換えられます。VAGen パーツ・エディションが前のマイグレーション・セットのためにすでに変換済みの場合、マイグレーション・ツールはパーツ・エディションを再度変換しません。ただし、パーツが現行マイグレーション・セットに含まれている場合、マイグレーション・ツールはそのパーツ・エディション用の EGL をファイルに組み込みます。VAGen マイグレーション設定またはデータベースの I/O 設定を変更する場合、あるいは**名前変更ユーザー**出口設定を変更する場合は、「**既存ファイルを上書き**」を選択してください。この場合は、データベースをステージ 1 の終了時の状態に復元し、新しい設定を使用してステージ 2 と 3 を再度実行できます。「**既存ファイルを上書き**」設定を選択すれば、EGL ワークスペースを最初に消去せずにステージ 2 と 3 を実行できます。「**残りの VAGen パーツをマイグレーションする**」設定を選択した場合にも、「**既存ファイルを上書き**」を選択すると便利です。この場合、マイグレーション・セットの別のバージョンをマイグレーションすると、EGL ファイルは、マイグレーション・セットの現行バージョンに含まれるパーツ・エディションを格納している新規ファイルに置き換えられます。
- 「**既存ファイルを上書き**」設定を解除した場合は、既存の EGL ファイルが、現行マイグレーション・セット内にある追加パーツをすべて格納するように変更されます。EGL ファイル内に既に存在しているパーツは、現行マイグレーション・セットが別のパーツ・エディションを使用していても、変更されません。「**残りの VAGen パーツをマイグレーションする**」設定を選択解除する場合に、マイグレーションする共通プロジェクトのバージョンが 1 つだけであるが、対象とするサブシステムが複数であるときに限って、「**既存ファイルを上書き**」設定を選択解除すると便利です。この場合、それぞれのサブシステムをマイグレーションしながら、共通プロジェクト用の EGL ファイルを段階的に徐々に構築できます。初期の EGL ファイルに含まれているものは、最初のサブシステムによって使用されている共通パーツのみです。2 番目のサブシステムをマイグレーションするときに、マイグレーション・ツールは 2 番目のサブシステムに必要な追加パーツを EGL ファイルに追加します。マイグレーション・ツールは、オリジナル・ファイルと追加パーツのマージを行って、引き続きパーツ型の中でパーツがアルファベット順に編成されるようにします。

マイグレーション・セットの複数のバージョンをマイグレーションする場合は、「**ワークスペースにインポート**」設定を選択できます。ただし、「**ワークスペースにインポート**」を選択解除し、代わりに「**マイグレーションしたファイルを一時ディレクトリーに保管**」を選択する技法の方をお勧めします。一時ディレクトリーを使用すると、マイグレーション・ツールはすべてのマイグレーション・セット・バージョンを作成できます。

「**マイグレーションしたファイルを一時ディレクトリーに保管**」設定を選択すると、マイグレーション・セットの複数のバージョンをマイグレーションし、すべてのバージョンをワークスペースの外部に保管できます。このオプションでは、EGL 開発環境の複数のインスタンスを同時に使用する必要があります。したがって、大量のメモリー・リソースが必要になるので、このオプションはバッチ・モードでのみ使用してください。「**マイグレーションしたファイルを一時ディレクトリーに保管**」を選択する際には、上位ディレクトリーも指定する必要があります。マイグレーション・ツールは、マイグレーション・セットのバージョンごとに、この上位ディレクトリー内にサブディレクトリーを作成します。「**マイグレーションしたファ**

イルを一時ディレクトリーに保管」は、「残りの VAGen パーツをマイグレーションする」と一緒に指定した場合に特に効果を発揮します。この状態では、1 つのマイグレーション・セット・バージョンに対応するそれぞれのサブディレクトリーに、そのマイグレーション・セット・バージョンに含まれる VAGen プロジェクト・バージョンのパーツがすべて保管されます。

一般規則

ソース・コードをマイグレーションする際にマイグレーション・ツールが行う処理には、いくつかの一般規則が適用されます。次のリストは、他のセクションで説明されている重要な規則の要約です。

- VisualAge Generator から EGL への変換を成功させるには、パーツ間マイグレーションが必要です。そのため、それぞれのマイグレーション・セットに共通 (共用) パーツを組み込む必要があります。パーツ間マイグレーションは、パーツのマイグレーションに次のような影響を及ぼします。
 - ステージ 1 では、パーツの配置が制御されます。データ項目、レコード、PSB、または関数に対してパーツ編集が発生する最初のマイグレーション・セットが、そのマイグレーション・セットに組み込まれた他のパーツに基づいて、パーツをプログラムと同じファイル内、共通パーツ・ファイル内、または未使用のパーツ・ファイル内のいずれに配置するかを決定します。このパーツ編集のための配置は、マイグレーション・データベースを消去しない限り、その後のすべてのマイグレーション・セットが使用します。
 - ステージ 2 では、EGL へのパーツの変換が制御されます。パーツ編集が発生する最初のマイグレーション・セットが、そのマイグレーション・セットに組み込まれた他のパーツに基づいて、パーツ間マイグレーションを使用した、EGL ソースへの変換を決定します。このパーツ編集のための変換は、そのパーツ編集のマイグレーション情報をリセットしない限り、その後のすべてのマイグレーション・セットが使用します。
- ステージ 3 では、マイグレーションの終了時に開始されるビルドの前に、常に、VisualAge Generator との互換性の設定がオンになり、「自動的にビルド」設定がオフになります。マイグレーション・ツールは、これらの設定を元の値に復元しません。
- ステージ 3 の場合、複数のバージョンのマイグレーション・セット、またはバージョンが異なる同じプロジェクトを含むマイグレーション・セットのマイグレーションは、バッチ・モードでのみサポートされます。

以下の一般的な規則も適用されます。

- マイグレーション・ツールは、少数の新規項目変数をそれぞれのプログラムに追加します。EZE ワードまたはその他のステートメントの EGL 置換をサポートするには、これらの変数が必要です。変数をプログラムに追加すると、プログラム内の関数は変数を使用できるようになります。詳しくは、111 ページの『マイグレーションに必要な中間変数』を参照してください。
- システム共通プロダクトや VisualAge Generator の旧リリースから VisualAge Generator 4.5 にマイグレーションされたパーツの中で、VisualAge Generator 4.5 内では変更されなかったパーツが存在する可能性があります。場合によっては、外部ソース形式から情報が欠落していたり、EGL ではサポートされない方法で情報が指定されていたりすることがあります。マイグレーション・ツールは、欠落や誤りに関する情報を、以下の例のように提供します。

- メインのトランザクションの場合、VAGen のセグメンテーション情報が欠落していれば、マイグレーション・ツールは EGL セグメント化プロパティをデフォルトで NO に設定します。
- SQL レコードの場合、SQL データ・コードが欠落していれば、マイグレーション・ツールはその項目を固定長項目に変換します。
- SQL 関数の場合、欠落している必須 SQL 文節があれば、マイグレーション・ツールは入出力オブジェクトとして指定されたレコードに基づいて、その文節の作成を試みます。
- 関数内では、マイグレーション・ツールはすべてのステートメントを何らかに変換します。これにより、IF / ELSE / END と WHILE / END のロジックは保持されます。ただし、得られるステートメントが構文的には正しくない可能性があります。例えば、次のようになります。
 - EZESYS 値が EGL ではサポートされない場合、マイグレーション・ツールは VAGen 値を使用します。これによって、サポートされない値が保持され、サポートされない環境 (TSO など) からサポート対応の EGL 環境 (ZOSCICS など) へ変更する場合に、ロジック更新が必要な可能性のある場所が見つけやすくなります。
 - EZESCRPT 特殊機能語は、EZE_SCRIPT へ変換されます。対応する置換表現は EGL には存在しません。EZESCRPT は IF、WHILE、または TEST ステートメント内では使用できないので、関数のロジック構造は保持されます。マイグレーション・ツールはエラー・メッセージを出し、EGL 検証の結果、「問題」ビューにエラー・メッセージは表示されません。
- プログラムを使用せずにパーツ参照の解決を試みる際に、マイグレーション・ツールはマイグレーション・セット内のパーツを調べます。このため、プロジェクトのグループに含めるマイグレーション・セットは、使用に適した組み合わせになるように定義する必要があります。例えば、次のようになります。
 - 競合するパーツ名をもつ複数のサブシステムからのプロジェクトを組み込まない。
 - サブシステムをマイグレーションする場合は、共通 Java プロジェクト、または共通 Smalltalk アプリケーションを組み込む。
- マイグレーション・ツールがマイグレーション時に行わないことがいくつかあります。
 - 次に示す状況では、マイグレーション・ツールは **import** ステートメントを追加しません。これは、VisualAge Generator 内ではこれらが関連パーツではないからです。
 - CALL、DXFR、または XFER のいずれかのステートメントを使用してプログラムに転送される関数。Java 用の生成を行っている場合は、その関数を含むファイル内でそのプログラムを含むパッケージの **import** ステートメントを追加するか、プログラム名をパッケージ名によって完全に修飾する必要があります。あるいは、リンケージ・オプション・パーツ内のエントリーを使用して、プログラムがあるパッケージの名前を指定することも、
programPackageName ビルド記述子オプションを使用して、生成されたすべての Java プログラムが強制的に同じランタイム・パッケージ内に配置されるようにすることもできます。

- .eglbld ファイル内のビルド・パーツの場合。生成オプション・パーツなどの VAGen 制御パーツは関連パーツをリストしないので、マイグレーション・ツールは情報を入手できません。また、EGL がビルド記述子パーツを処理する方法が原因で、**nextBuildDescriptor** 値の再配列が必要になる可能性があります (VAGen /OPTIONS)。さらに、この再配列を行うには、マイグレーション・ツールが行ったインポートを変更する必要が生じます。

注: ステージ 1 マイグレーション・ツールは、マイグレーション・セット内のパーツを分析して、各パーツの関連パーツを判別します。マイグレーション・セット内のパーツのみが分析対象に含まれるように、上位 PLP プロジェクトによって指定されたマイグレーション・セットをロードする前に、ステージ 1 マイグレーション・ツールはすべての Java プロジェクトをワークスペースから削除します。同様に、上位構成マップによって指定されたマイグレーション・セットをロードする前に、ステージ 1 マイグレーション・ツールはすべての Smalltalk 構成マップを削除します。関連パーツの分析対象はマイグレーション・セットに限定されるので、マイグレーション・ツールは、マイグレーション・セットに含まれない EGL プロジェクトを指定する EGL ビルド・パス・プロパティを設定しません。また、マイグレーション・ツールは、マイグレーション・セットに含まれない EGL パッケージの **import** ステートメントを組み込みません。

- EGL は、暗黙項目を許容しません。VisualAge Generator は暗黙項目を許可しますが、暗黙項目を実際を使用することは、一般的に好ましくありません。暗黙項目は、プログラム内で使用される項目ですが、そのプログラムによって使用されるレコード、テーブル、またはマップ内では明示的に定義されていないものです。非修飾のデータ項目をすべて評価して、暗黙項目かどうか判別する処理はパフォーマンスに影響を及ぼすため、マイグレーション・ツールは暗黙項目の定義を作成しません。マイグレーションの前に、ユーザーが暗黙項目の定義を提供する必要があります。マイグレーションの前に問題を解決するには、VisualAge Generator 内でプログラムを検証して、プログラムが実際に暗黙項目を使用しているかどうか判別してください。暗黙項目が使用されている場合、VAGen 検証機能はその項目の正しい定義を示すメッセージを出します。マイグレーションの前に暗黙項目の定義を作成しない場合は、EGL 検証によって「問題」ビューにエラー・メッセージが示されます。この問題は EGL で訂正できます。マイグレーションのステージ 1 の実行前に暗黙項目を作成する場合に役立つホワイト・ペーパーについての詳細は、19 ページの『参照』を参照してください。
- マイグレーション・ツールは、**containerContextDependent** プロパティの設定を行いません。このプロパティは、サブシステムによって提供される他のパーツを参照する必要がある、特定の共通レコードまたは共通関数に対して後で追加できます。レコードまたは関数に対してこのプロパティを使用する方法について詳しくは、48 ページの『containerContextDependent プロパティ』を参照してください。
- マイグレーション・ツールは、VAGen 構文が有効であることと、マイグレーション・セットに含まれるパーツを使用するプログラムが VisualAge Generator 内で正常に検証できることを前提とします。マイグレーション・ツールは、無効な構文の修復を試行しません。例えば、次のようになります。

- マップ配列の要素が異なる編集特性や属性を備えている場合は、配列の先頭要素の特性によって、EGL にマイグレーションされる内容が決まります。マイグレーション・ツールはメッセージを出しません。
- レコード内の共用データ項目の長さが変更されたために、親項目の長さがその副構造内にある項目の長さの合計と一致しない場合、マイグレーション・ツールは項目の長さを変更せず、メッセージを出しません。EGL 検証によって、「問題ビュー」に、子の長さの合計が親の長さとは一致しないことを示すエラー・メッセージが出されます。
- マイグレーション・ツールは、EGL の構文エラーが発生する場合であっても、効率が良くないコードの改良を試みません。例えば、次のようになります。
 - プログラムのテーブルおよび追加レコードのリストに同じレコードが 2 回リストされている場合、マイグレーション・ツールはそのレコードを除去せず、メッセージも出しません。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。同様に、プログラムのテーブルおよび追加レコードのリストに同じテーブルが 2 回リストされている場合、マイグレーション・ツールは余分なテーブルを除去せず、メッセージも出しません。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。
 - レコードがプログラム内で使用されていないにもかかわらず、プログラムのテーブルおよび追加レコードのリストにある場合、マイグレーション・ツールはそのレコードを除去せず、メッセージも出しません。EGL 検証の結果、「問題」ビューにエラー・メッセージは表示されません。同様に、テーブルがプログラム内で使用されていないにもかかわらず、プログラムのテーブルおよび追加レコードのリストにある場合、マイグレーション・ツールはそのテーブルを除去せず、メッセージも出しません。EGL 検証の結果、「問題」ビューにエラー・メッセージは表示されません。
 - 作業用ストレージ・レコードがプログラムのテーブルおよび追加レコードのリストにあり、レコードがレベル 77 項目のみを含んでいる場合、マイグレーション・ツールはそのレコードを除去せず、メッセージも出しません。EGL 検証の結果、レコードを検出できないことを示すエラー・メッセージが「問題」ビューに表示されます。これは、現時点で存在するレコードが、設定した**レベル 77 接尾部**をレコード名の一部として含むレコードのみであるためです。
- VAGen プログラムがマップ・グループまたはヘルプ・マップ・グループを含んでいても、必ずしも実際にマップ・グループ・パーツが存在していたとは限りません。例えば、プログラムがマップの表示 (DISPLAY) や変換 (CONVERSE) を行ったことがなく、またプログラムがマップを呼び出し先パラメーターとして使用したことがない場合、実際のマップ・グループ・パーツは存在する必要がなかったことになります。この状態では、マイグレーション・ツールはプログラムに FormGroup の **use** ステートメントを組み込みますが、**import** ステートメントは組み込みません。これは、マップ・グループが VisualAge Generator 内でプログラムの関連パーツではなかったからです。マイグレーション・ツールはエラー・メッセージを出しません。EGL は実際の FormGroup パーツを必要とします。FormGroup パーツが存在しない場合、または FormGroup パーツがプログラムと同じパッケージに入っていない場合は、プログラムの **use** 宣言に指定されている FormGroup を検出できないことを示すエラー・メッセージが、EGL 検証によって「問題」ビューに表示されます。

- 文書化されていなかった VisualAge Generator の機能の使用に関しては、同等な EGL サポートが存在しなくても、マイグレーション・ツールがそのことを検出したり、警告メッセージを出したりするとは限りません。例えば、次のようになります。
- CALL ステートメントが非修飾項目を引数として指定していて、プログラム内にこの項目名の定義が複数ある場合、VAGen はプログラムの 1 次作業用ストレージ・レコード内のレベル 77 項目を優先します。EGL は、項目が修飾されていることを必要とします。マイグレーション・ツールによって修飾が追加されることはなく、警告メッセージも出されません。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。

EGL ソース・コードの編成方法の決定

EGL 用のソース・コードを編成する前に、次の概念を理解する必要があります。

- VisualAge Generator 製品と EGL 製品の違い (特に次の領域において)。
 - 製品がコードの編成用に提供している機能。
 - 製品が開発、テスト、生成の対象となるパーツを決定する方法。
 - 製品がパーツに対する変更を追跡する方法。
- 最終的に必要な編成を EGL で実現するためにマイグレーション・ツールによって提供されている支援機能。
- EGL でのさまざまなソース・コード編成技法の制限とトレードオフ。

製品のコード編成機能に関する違い

VisualAge Generator と EGL では、ソース・コードを編成する方法が異なります。Java での VAGen はプロジェクトとパッケージを使用します。Smalltalk での VAGen は構成マップとアプリケーションを使用します。EGL はプロジェクト、パッケージ、およびファイルを使用します。

Java での VAGen のコード編成

Java での VAGen は、ソース・コードを Java プロジェクトおよびパッケージに編成します。例えば、SubsystemA に固有のコードを含むすべてのパッケージを組み込んだ ProjectA、SubsystemB に固有のコードを含むすべてのパッケージを組み込んだ ProjectB、複数のサブシステムで共用されるすべてのパッケージを組み込んだ ProjectCommon を持つように編成できます。

ワークスペースに追加する Java プロジェクトによって、VAGen プログラムを開発、テスト、または生成する際に対象となるソース・コードが決定します。プロジェクト・リスト・パート (PLP) を含むプロジェクトを使用して、同時にワークスペースに追加する必要がある他のプロジェクトを特定できます。例えば、SubsystemA に固有のコードを含むすべてのパッケージを指定するほか、ProjectCommon が VAGen の必須プロジェクトであることを指定する PLP を ProjectA に組み込むことができます。VAGen に必須のプロジェクトを指定すると、ProjectA をワークスペースにロードするたびに、正しいバージョンの ProjectCommon およびそれに含まれるすべてのパッケージ・バージョンもロードされるので、開発、テスト、および生成に必要なすべてのパーツを確実に保持できます。重複するパーツ名を含むプロジェクトをワークスペースに追加することはできますが、ワークスペースに重複するパーツがあると、テストまたは生成はできません。

パーツに変更を加えると、VisualAge Generator はパーツの新しいエディションをワークスペースおよび ENVY リポジトリに作成します。VisualAge Generator はバージョン管理と呼ばれる技法を使用して、コードを既知のレベルで凍結します。ENVY リポジトリには、Java プロジェクトまたはパッケージのすべてのバージョンが保管されますが、ワークスペースにロードできるのは一度に 1 つのバージョンだけです。ツールは、ワークスペース内のバージョンを ENVY リポジトリ内の以前のバージョンと比較し、プロジェクト、パッケージ、またはパーツ・レベルで変更内容を確認する手段を提供します。変更を追跡するためには、同じ Java プロジェクトの異なる複数のバージョンを、開発、テストの各レベル、または実動に使用できます。そのほかに、開発、テストの各レベル、および実動用にそれぞれ 1 つずつプロジェクトを用意して、変更を追跡する方法もあります。

Smalltalk での VAGen のコード編成

Smalltalk での VAGen は、ソース・コードを Smalltalk 構成マップおよびアプリケーションに編成します。例えば、SubsystemA に固有のコードを含むすべてのアプリケーションを組み込んだ ConfigurationMapA、SubsystemB に固有のコードを含むすべてのアプリケーションを組み込んだ ConfigurationMapB、複数のサブシステムで共用されるすべてのアプリケーションを組み込んだ ConfigurationMapCommon を持つように編成できます。

イメージにロードする Smalltalk 構成マップによって、VAGen プログラムを開発、テスト、または生成する際に対象となるソース・コードが決定します。構成マップを使用すると、イメージにロードする Smalltalk アプリケーションを簡単に指定できます。例えば、SubsystemA に固有のコードを含むすべての Smalltalk アプリケーションを指定するほか、ConfigurationMapA は ConfigurationMapCommon が必須の構成マップであることを指定できます。必須の構成マップを指定すると、ConfigurationMapA をイメージにロードするたびに、正しいバージョンの ConfigurationMapCommon およびそれに含まれるすべてのアプリケーション・バージョンもロードされるので、開発、テスト、および生成に必要なすべてのパーツを確実に保持できます。2 つのアプリケーションまたは構成マップが重複するパーツ名を含む場合、それらをイメージにロードすることはできません。

パーツに変更を加えると、VisualAge Generator はパーツの新しいエディションをイメージおよび ENVY マネージャーに作成します。VisualAge Generator はバージョン管理と呼ばれる技法を使用して、コードを既知のレベルで凍結します。ENVY マネージャーには Smalltalk 構成マップまたはアプリケーションのすべてのバージョンが保管されますが、イメージにロードできるのは一度に 1 つのバージョンだけです。ツールは、イメージ内のバージョンを ENVY マネージャー内の以前のバージョンと比較し、構成マップ、アプリケーション、またはパーツ・レベルで変更内容を確認する手段を提供します。変更を追跡するためには、同じ Smalltalk 構成マップの異なる複数のバージョンを開発、テストの各レベル、または実動に使用できます。そのほかに、開発、テストの各レベル、および実動用にそれぞれ 1 つずつ Smalltalk 構成マップを用意して、変更を追跡する方法もあります。

EGL のコード編成

EGL では、ソース・コードをプロジェクト、パッケージ、およびファイルに編成します。例えば、SubsystemA に固有のコードを含むすべてのパッケージを組み込んだ ProjectA、SubsystemB に固有のコードを含むすべてのパッケージを組み込んだ ProjectB、複数のサブシステムで共用されるすべてのパッケージを組み込んだ

ProjectCommon を持つように編成できます。これは Java での VAGen と似ていますが、次の 2 つの重要な手法が異なります。

- EGL には Java での VAGen の PLP と同様の概念はありません。代わりに、開発、デバッグ、および生成に必要なすべてのパーツを使用できるようにするため、ワークスペースにロードするプロジェクトを決定する必要があります。
- EGL ファイルはソース・コードのさらに詳細な編成を指定します。例えば、ProjectA の中に、SubsystemA のプログラムが共用するデータ用に 1 つのパッケージを作成できます。このパッケージには、SubsystemA のすべてのデータ項目パーツと、SubsystemA 内の複数のプログラムが使用するレコードを入れることができます。このパッケージをファイルに編成する方法は、次のスキームを含め、いくつかあります。
 - すべてのデータ項目とレコードを含む 1 つのファイル。
 - すべてのデータ項目を含む 1 つのファイルと、すべてのレコードを含むもう 1 つのファイル。
 - 文字 A から M で始まるデータ項目を含む 1 つのファイルと、文字 N から Z で始まるデータ項目を含むもう 1 つのファイル、およびすべてのレコードを含む 1 つのファイル。

EGL では、Program、DataTable、および FormGroup の各パーツが固有のファイルに入っている必要がありますが、そのファイルに他のパーツを入れることもできます。例えば、ProgramX のパーツとともに ProgramX に固有の関数とレコードを含む ProgramX 用のファイルを作成できます。

EGL プロジェクトを作成するとき、EGL ビルド・パスを使用して、EGL プログラム、DataTable、または FormGroup を開発、テスト、または生成する際に対象となるその他のプロジェクトを指定します。プロジェクトの EGL ビルド・パスにより、パーツ名を検索する際に対象となる他のプロジェクトが限定されます。また、EGL は、パーツ名を検索する際に、各ファイル内の **import** ステートメントを使用して、EGL ビルド・パスにリストされたプロジェクトの中からどのパッケージを組み込むかを判別します。ワークスペース内に重複するパーツ名があってもかまいませんが、EGL ビルド・パス内および一連の **import** ステートメント内のパーツ名は固有でなければなりません。

パーツに変更を加えてファイルを保存すると、EGL はそのファイルを以前のファイルと置き換えてファイル・システムに保管します。ソース・コード・リポジトリを使用してコードの複数のバージョンを維持します。ソース・コード・リポジトリには、チェックアウトとチェックイン、バージョン管理のほか、ワークスペース内のコードのバージョンを、ソース・コード・リポジトリ内のそのコードの他のバージョンと比較できる比較ツールなどのツールが用意されています。ソース・コード・リポジトリを使用すると、開発者が変更内容を共有することもできます。EGL で使用できるソース・コード・リポジトリは多数あります。例として、CVS と IBM Rational ClearCase® があります。どのソース・コード・リポジトリを選択するかに関係なく、ワークスペースに一度にロードできるのはプロジェクト、パッケージ、またはファイルの 1 つのバージョンだけです。

マイグレーション・ツールが提供する編成機能

ステージ 1 マイグレーション・ツールは、EGL プロジェクト、パッケージ、およびファイルの編成における各パーツの配置を決定します。デフォルトでは、Java のステージ 1 マイグレーション・ツールは、Java プロジェクトの各 VAGen を EGL プロジェクトに変換し、Java パッケージの各 VAGen を EGL パッケージに変換することによって、Java プロジェクトおよびパッケージ構造における VAGen を維持します。同様に、Smalltalk のステージ 1 マイグレーション・ツールは、Smalltalk 構成マップの各 VAGen を EGL プロジェクトに変換し、Smalltalk アプリケーションの各 VAGen を EGL パッケージに変換することによって、Smalltalk 構成マップおよびアプリケーション構造における VAGen を維持します。このデフォルトの維持ポリシーに対する唯一の例外は、マップおよび対応するマップ・グループが複数の Java プロジェクトやパッケージ、または複数の Smalltalk 構成マップやアプリケーションに存在する場合に発生します。このような例外の場合、マイグレーション・ツールは、マップおよびマップ・グループを、それぞれの元の配置に応じて、新しい EGL パッケージまたはプロジェクトにマージします。この例外の詳細や、ステージ 1 マイグレーション・ツールが VAGen パーツをファイルに割り当てる方法については、50 ページの『EGL ファイル内でのパーツの配置』を参照してください。

69 ページの『EGL ソース・コード編成技法の制限とトレードオフ』で説明しているように、ステージ 1 マイグレーション・ツールが使用するデフォルトとは異なる方法で、EGL プロジェクト、パッケージ、およびファイルを編成したい場合があります。このような理由から、ステージ 1 マイグレーション・ツールはサンプル・プログラムとして付属しています。ステージ 1 ツールは、各自の環境に合わせて変更できます。以下の資料とホワイト・ペーパーには、ステージ 1 ツールに対する変更のサンプルが掲載されています。

- ファイルのロケーションに関するセクション:
 - Java のステージ 1 ツールについて詳しくは、164 ページの『ステージ 1 マイグレーション・ツールのカスタマイズ』を参照してください。
 - Smalltalk のステージ 1 ツールについて詳しくは、192 ページの『ステージ 1 マイグレーション・ツールのカスタマイズ』を参照してください。
- プロジェクト統合に関する資料:
 - *How to Consolidate Projects and Packages during Stage 1 of the VisualAge Generator on Java to Enterprise Generation Language Migration Tool.*
 - *How to Consolidate Projects and Packages during Stage 1 of the VisualAge Generator on Smalltalk to Enterprise Generation Language Migration Tool*

ファイルのロケーションに関する 2 つのセクションに、共通パーツおよび未使用パーツのパーツ配置アルゴリズムを変更し、パーツ型およびパーツ名の先頭文字に基づいて commonParts.egl および unusedParts.egl ファイルを小さいファイルに分割する例が記載されています。すべてのデータ項目パーツを新しいプロジェクトに移動する方法に関する情報も含まれています。プロジェクト統合に関する 2 つのホワイト・ペーパーには、複数の Java プロジェクトまたは Smalltalk 構成マップを単一の EGL プロジェクトにマージする例と、複数の Java パッケージまたは Smalltalk アプリケーションを単一の EGL パッケージにマージする例が記載されています。

これらのホワイト・ペーパーは、次の Web サイトの EGL Cafe の「VAGen Migration」ハブから入手できます。

<http://www.ibm.com/software/rational/cafe/community/egl/vagen?view=documents>

EGL ソース・コード編成技法の制限とトレードオフ

EGL はプロジェクトを機能ラインに従って編成した場合に最も効果的です。この技法は、ワークスペース内に必要な EGL プロジェクトの数を最小限に抑え、その結果パフォーマンスを向上させるのに役立ちます。Java プロジェクトおよびパッケージでの VAGen や、Smalltalk 構成マップおよびアプリケーションでの VAGen は、既に機能ラインに従って編成されている場合があります。その場合は、デフォルトのステージ 1 マイグレーション・ツールを使用できることがあります。ただし、デフォルトの EGL プロジェクトおよびパッケージの戦略、またはステージ 1 マイグレーション・ツールが使用するデフォルトのファイル配置アルゴリズムの変更を必要とする可能性のある他の要因について考慮する必要があります。

考慮すべき主要な要因は、**EGL プロジェクト、パッケージ、およびファイルのサイズが問題になる**ということです。サイズは次のいずれかの領域で、EGL のパフォーマンスに影響する可能性があります。

- ファイルを開く、または保管するのに必要な時間。
- パーツを検証するために必要な EGL ビルド時間。ワークベンチの設定で自動ビルドをオンにした場合、ファイルを保管するとき常に EGL ビルドが実行されます。この設定をオンにしない場合、EGL ビルドは要求したときに実行されます。少なくともコードをデバッグまたは生成する前には EGL ビルドを実行する必要があります。
- Java または COBOL ソース・コードを生成するために必要な EGL 生成時間。

次の表は、プロジェクト、パッケージ、およびファイルのサイズに関する制限とトレードオフを示しています。

表 9. プロジェクト、パッケージ、およびファイルのサイズに関する制限とトレードオフ

	制限	より小さい場合の利点	より大きい場合の利点
プロジェクト	<ul style="list-style-type: none"> 最大 1500 プロジェクト。 一部のソース・コード・リポジトリでは最大値がこれより小さい可能性があります。 	<ul style="list-style-type: none"> プロジェクトのサブセットのみが必要な場合は、ワークスペースは小さくなり、EGL ビルド時間が短縮されることがあります。 プロジェクトが小さいほど、ソース・コード・リポジトリからのロードが速くなります。 2 人の開発者が同じプロジェクトに対して同時に変更を行うことが必要とされる機会が最小限になります。 	<ul style="list-style-type: none"> EGL ビルド・パスのプロジェクトが少なくなります。 EGL ビルド・パスでサイクルが発生する可能性が減り、サイクル回数が減少します。 スクロールするプロジェクト数が少なくなります。 ソース・コード・リポジトリからロードするプロジェクト数が減ります。
パッケージ	<ul style="list-style-type: none"> パーツ名の長さが 8 文字以下である場合、FAT32 としてフォーマットされたドライブ上でのパーツの論理上の最大数は 65,000。パーツ名がそれより長いと、この最大値が小さくなります。 	<ul style="list-style-type: none"> import ステートメント <code>import package.*</code> のような書式である場合、含まれるパーツ名が少なくなるため、EGL ビルド時間が短縮される可能性があります。 2 人の開発者が同じパッケージに対して同時に変更を行うことが必要とされる機会が最小限になります。 	<ul style="list-style-type: none"> 各ファイル内の import ステートメント数が少なくなります。 スクロールするパッケージ数が少なくなります。

表 9. プロジェクト、パッケージ、およびファイルのサイズに関する制限とトレードオフ (続き)

	制限	より小さい場合の利点	より大きい場合の利点
ファイル	<ul style="list-style-type: none"> ファイル・サイズが大きい場合はファイルを開く、または保管するパフォーマンスが低下します。 最大サイズではありませんが、現実的なサイズは 200K 未満です。 	<ul style="list-style-type: none"> ファイルを保管するときに変更の有無を確認するパーツの数が減るため、EGL ビルド時間が短縮される場合があります。 適切な命名規則を使用することで、特定のパーツを含むファイルを素早く検索できます。 2 人の開発者が同じファイルに対して同時に変更を行うことが必要とされる機会が最小限になります。 	<ul style="list-style-type: none"> スクロールするファイル数が少なくなります。

表 9 に説明したトレードオフを考慮し、巨大な EGL プロジェクト、パッケージ、またはファイルを作成することは避けてください。逆に、多数の小さい EGL プロジェクト、パッケージ、またはファイルを作成することも避けてください。これらの両極端な状態はパフォーマンスを低下させる原因になります。例えば、30,000 個のデータ項目を含むファイルは非常にサイズが大きくなり、開いたり保管したりするのに数分を要する可能性があります。一方、データ項目が 1 つしかないファイルが 30,000 個ある場合は、ファイル数が多すぎて各種ワークベンチ・ビューで通常のようにスクロールすることはできません。妥協案をとることをお勧めします。例えば、考えられる妥協案の 1 つは、データ項目名の先頭文字としてアルファベットがまんべんなく使われている場合、複数のファイルを作成し、各ファイルに同じ文字から始まるデータ項目をすべて入れるという方法です。この技法を使用すると、ファイルのサイズが比較的小さくなるため開くのに時間がかからず、しかもファイルの合計数が最小限になります。

さらに、EGL プロジェクト、パッケージ、およびファイルの構造を決定するときに以下の要因を考慮してください。

- ソース・コード・リポジトリの考慮。選択したソース・コード・レジストリーに基づいて考慮しなければならない場合があるいくつかの要因を以下のリストに示します。
 - ソース・コード・リポジトリがプロジェクト、パッケージ、またはファイルのレベルでチェックアウトとチェックインをサポートするかどうか。
 - ソース・コード・リポジトリがプロジェクト、パッケージ、またはファイルのレベルでバージョン管理をサポートするかどうか。
 - 複数の開発者が同じプロジェクト、パッケージ、またはファイルに対して同時に変更を行うことが必要になった場合に、そのような状況をソース・コード・

リポジトリがどのように処理するか。また、使用している EGL プロジェクト、パッケージ、およびファイルの編成から考えてそのような状況が発生する可能性はどれぐらいか。

- ソース・コード・リポジトリを使用する場合、そのソース・コード・リポジトリが所有権またはアクセス制御の概念に対するサポートを備えている場合があります。その場合は、以下の要因を考慮する必要があります。
 - 所有権の戦略が開発と保守の責任分担を反映していること。1 人の担当者または 1 つのグループが機能領域の開発と保守を担当する場合は、機能ラインに従ってソース・コードを編成すると便利です。
 - 特定のプログラムまたはパーツに対するアクセスを制限する制約があるかどうか。例えば、給与計算チェックを作成するプログラムへのアクセスを 1 人か 2 人の開発者に限定する場合があります。この場合、そのプログラムを単独で EGL プロジェクトに配置して、そのプロジェクトへのアクセスを制限できるようにする必要があります。

EGL 5.1.2 以降の VAGen マイグレーション・ツールの新機能

注: EGL 5.1.2 を使用してマイグレーション・データベースを作成した場合は、547 ページの『DB2 マイグレーション・データベースの作成』に記載されている手順に従ってデータベースを再度作成する必要があります。

次に示す機能は、EGL 5.1.2 の一般出荷開始日以降に新しく追加されたか、または改良されたものです。ご使用の WebSphere® Developer 5.1.2 製品の保守レベルによっては、ステージ 1 の一部の機能がすでに強化されている場合があります。

- ステージ 1 は次のように変更されました。
 - マイグレーション・データベース用の DDL が更新されました。
SetupTables.bat を再実行し、ステージ 1 を再実行して、マイグレーション・データベースにソース・コードを配置する必要があります。
 - EGL 予約語リストを外部化し、予約語リストのバージョンをログ情報に組み込みました。
 - エラーがないかどうかステージ 1 の結果を検査する、新しい checkStage1.bat ファイルを追加しました。
- ステージ 2 および 3 と単一ファイル・モードの両方のマイグレーションに、次の変更が組み込まれました。
 - 新しい EGL 構文のサポート。
 - オプションのユーザー出口を呼び出して、ステージ 2 マイグレーション時、または単一ファイル・マイグレーション時にパーツの名前を変更できます。例えば、ハイフン (-) を下線 (_) に変更するユーザー出口を作成できます。
 - 新しい VAGen 構文マイグレーション設定により、SQL レコード内のフィールドに対して次のオプションを指定できます。
 - SQL 列名が項目名と同じである場合に、マイグレーション・ツールが **column** プロパティを省略する。
 - マイグレーション・ツールは、通常 **isSQLNullable = YES** プロパティを省略する。

- SQL レコードに対して SQL 表が 1 つだけ指定されており、VAGen の「読み取り専用」プロパティが yes に設定されている場合に限り、マイグレーション・ツールが **isReadOnly** = YES プロパティを組み込む。
- 新しい VAGen 構文マイグレーション設定により、10 進数のコンマを小数点に変更するようにマイグレーション・ツールに指定できます (ご使用のワークステーションが、小数点をデフォルトに設定するロケールを使用している場合であっても)。
- ステージ 2 から 3 へのマイグレーションは、次の処理も行います。
 - **validatorFunction** プロパティまたは **validatorDataTable** プロパティを指定する DataItem パーツに対して、**import** ステートメントを追加します。
 - パーツ型の中で、EGL パーツ名を基準にファイル内のパーツをソートします。

次に示す EGL の変更により、マイグレーションされた VAGen プログラムのサポートが改良されます。

- スtring連結に数値変数を使用できます。これにより、EGL の **prepare** ステートメントにマイグレーションされる、VAGen の SQL I/O Execution time statement build オプションがサポートされます。
- VisualAge Generator に備わっていた機能と同様に、Java ランタイム環境で EGL 製品メッセージを変更できます。

EGL 6.0 iFix 以降の VAGen マイグレーション・ツールの新機能

注: EGL 6.0 iFix を使用してマイグレーション・データベースを作成した場合は、547 ページの『DB2 マイグレーション・データベースの作成』に記載されている手順に従ってデータベースを再度作成する必要があります。

次に示す機能は、EGL 6.0 iFix 以降に新しく追加されたか、または改良されたものです。

- ステージ 2 および 3 と単一ファイル・モードの両方のマイグレーションに、次の変更が組み込まれました。
 - ユーザー出口を呼び出して、ステージ 2 マイグレーション時、または単一ファイル・マイグレーション時にパーツの名前を変更できる機能をさらにサポートします。
 - 新しい VAGen マイグレーション設定により、VisualAge Generator 互換モードの使用を最小化できます。

EGL 6.0.0.1 以降の VAGen マイグレーション・ツールの新機能

注: EGL 6.0.0.1 を使用してマイグレーション・データベースを作成した場合は、547 ページの『DB2 マイグレーション・データベースの作成』に記載されている手順に従ってデータベースを再度作成する必要があります。

次に示す機能は、EGL 6.0.0.1 以降に新しく追加されたか、または改良されたものです。

- Web トランザクション・プログラムおよび UI レコードをマイグレーションするためのサポート。 Web トランザクションに関連する生成オプションのマイグレーションもサポートします。
- DL/I レコード、DL/I ファンクション I/O、PSB パーツ、EZEDL*、特殊機能語、CSPTDLI 特殊機能語をマイグレーションするためのサポート。生成オプション、リソース関連オプション、IMSVS/IMSBMP 環境に関連する関連リンケージ・テーブルのマイグレーションもサポートします (GSAM およびメッセージ・キューのサポートを含みます)。

EGL 6.0.1 以降の VAGen マイグレーション・ツールの新機能

注: EGL 6.0.1 を使用してマイグレーション・データベースを作成した場合は、547 ページの『DB2 マイグレーション・データベースの作成』に記載されている手順に従ってデータベースを再度作成する必要があります。

次に示す機能は、EGL 6.0.1 以降に新しく追加されたか、または改良されたものです。

- ステージ 1 Smalltalk ではサブアプリケーションをサポートしています。新しい Collapse サブアプリケーション設定の詳細については、183 ページの『「マッピング」ページ』を参照してください。
- 構文マイグレーションの改良:
 - 単一ファイル・モードでは、パーツは各ファイルのパーツ型の中でパーツ名順にソートされるようになりました。
 - プログラム内のレコード宣言はレコード名順にソートされるようになりました。
 - NOT 符号を使用した SQL 比較演算は、コード・ページの独立演算子に変換されるようになりました。
 - DL/I レコード宣言は、そのプログラムの PSB に基づいたプログラムには、自動的に追加されません。プログラムに追加される DL/I レコード宣言は、そのプログラムに使用されている入出力オプションにとって必要なものだけに限られます。
 - 配列を含むマップは現在、**indexOrientation**、**columns**、**linesBetweenRows**、および **spacesBetweenColumns** プロパティを使用して、標準配列に対する位置情報を提供します。これにより、マップ用の EGL フォーム・エディターが使用可能になります。
 - データ項目、マップ上のフィールド、および UI レコードで共有しないフィールドをマイグレーションする際、マイグレーション・ツールで常に **sign** プロパティが設定されます。
- ステージ 2 および 3:
 - 関数でテーブルが参照される場合、マイグレーション・ツールで **import** ステートメントが追加されます。
 - 複数のマイグレーション・セットを同時にマイグレーションする、または「ユーザー出口の名前変更」設定を使用してパーツを名前変更すると、ロジックのマージが向上します。
 - パフォーマンス改善。

EGL 6.0.1.1 以降の VAGen マイグレーション・ツールの新機能

注: EGL 6.0.1.1 を使用してマイグレーション・データベースを作成した場合は、547 ページの『DB2 マイグレーション・データベースの作成』に記載されている手順に従ってデータベースを再度作成する必要があります。

「マイグレーション・ガイド」のこの暫定リリース (6.0.1.1 ifix3) では、マイグレーション・ツール自体は大きく変更されていません。しかし、「マイグレーション・ガイド」は次のような新しい情報で更新されています。

- 第 1 章は次のように変更されました。
 - 3 ページの『本書で使用される用語』のセクションを、IBM Rational COBOL Generation Extension for zSeries および IBM Rational COBOL Runtime for zSeries が使用可能かどうかに基づいて更新および拡張しました。
 - 5 ページの『マイグレーションの計画』のセクションを更新および拡張しました。
 - 19 ページの『参照』のセクションを追加しました。このセクションには、VisualAge Generator から EGL へのマイグレーションに関連して提供されているホワイト・ペーパーのリストも記載しています。
- 第 2 章に以下のセクションが追加されました。
 - 65 ページの『EGL ソース・コードの編成方法の決定』
- 第 9 章に以下のセクションが追加されました。
 - 261 ページの『VAGen 準備テンプレートおよびプロシージャの EGL ビルド・スクリプトへの変換』
 - 262 ページの『VAGen ランタイム・テンプレートの変換』
 - 264 ページの『VAGen 予約語ファイルの変換』
 - 259 ページの『zSeries に対する EGL サーバー製品のインストール』
 - 267 ページの『ソース・コードの二重メンテナンスの計画』
- 第 10 章に以下のセクションが追加されました。
 - 277 ページの『SQL サポートに関する違い』。以前は章全体に分散していた SQL に関する情報をまとめ、お客様の体験に基づいて情報を拡張しました。
 - VAGen のランタイム環境変数とランタイム・プロパティのマイグレーションに関する情報。
- 付録 B に以下のセクションが追加されました。
 - 461 ページの『準備テンプレートおよびプロシージャ』
 - 463 ページの『ランタイム・テンプレート』
 - 466 ページの『ランタイム環境変数』
 - 468 ページの『vgj.properties』
- 付録 E は次のように変更されました。
 - お客様の体験と EGL 妥当性検査メッセージに対する機能拡張に基づいて、いくつかのメッセージに関する説明を拡張し、新しいメッセージを追加しました。
 - 537 ページの『メッセージの参照情報 - ネーム解決規則および名前の修飾規則』のセクションを追加しました。このセクションには、別のレコード、フォ

ーム、またはデータ・テーブルと同じ名前のフィールドがある場合に、VisualAge Generator と EGL のネーム解決の違いによって生成されるメッセージを解決するために役立つ情報が記載されています。

- 付録 F は、VisualAge Generator に必要な APAR のリストで更新されました。

EGL 6.0.1.1 ifix003 以降のマイグレーション・ツールの新機能

次に示す機能は、EGL 6.0.1.1 ifix003 以降に新しく追加されたか、改良されたものです。

- 両方のステージ 1 ツールで、commonParts.egl ファイル内の複数の生成可能パーツで参照されている関数およびデータ項目が正しく配置されます。さらに、Smalltalk のステージ 1 ツールが更新され、メッセージ・テーブル接頭部を指定するプログラムの関連付けとしてメッセージ・テーブルが追加されました。この問題は、Java のステージ 1 ツールでは発生していませんでした。
- 両方のステージ 1 ツールが完全に書き換えられ、大規模なマイグレーション・セットのマイグレーションが改良されました。ステージ 1 ツールを実行する際には、次の違いがあります。
 - ステージ 1 データベースを前もってキャプチャーした場合は、既存のデータベースでステージ 2 および 3 を実行できます。これにより、既存のデータベースに対して新しい EGL バージョン 7.1 の構文を使用できます。
 - 新しいステージ 1 ツールを実行するには、以下の手順に従う必要があります。
 - 修正パッケージ 15 を適用した DB2 バージョン 8.1、または修正パッケージ 8 を適用した DB2 バージョン 8.2 をインストールします。これらの 2 つの修正パッケージ・レベルは同等です。
 - DB2 のコマンド・プロンプト・ウィンドウで、setupDatabase.bat と setupTables.bat を実行します。正しいレベルの DB2 をインストールした後、2 つの .bat ファイルを実行する必要があります。
 - EGL V7.1 より前のバージョンからステージ 1 データベースを復元した後で、ステージ 1 を再度実行することに決めた場合は、setupDatabase.bat と setupTables.bat を再度実行する必要があります。古いデータベースを復元すると、データベース定義が以前のバージョンのマイグレーション・ツールで使用していた定義にリセットされるからです。
 - 以前に Java でステージ 1 マイグレーション・ツールを使用していた場合は、新しいバージョンのステージ 1 ツールをインストールした後で、以下の追加手順を実行する必要があります。
 1. Java の VAGen の「ワークベンチ」ウィンドウで、「プロジェクト」タブをクリックします。
 2. 「IBM VisualAge Generator EGL マイグレーション」プロジェクトにナビゲートする。
 3. マイグレーション・プロジェクトを展開し、com.ibm.vgj.mig パッケージを展開する。
 4. パッケージ内で、VAGenToEGLMigration クラスを右クリックし、以下のステップを実行する。

- a. ポップアップ・メニューで「プロパティ (Properties)」をクリックする。
 - b. 「クラスパス」タブをクリックします。
 - c. 「プロジェクト・パス」セクションの横の「編集」をクリックします。
 - d. 「クラスパス」ウィンドウで、「**IBM VisualAge Generator ランタイム (IBM VisualAge Generator Runtime)**」を必ず確認するようにします。
 - e. 「OK」をクリックして「クラスパス」ウィンドウを閉じます。
 - f. 「OK」を再度クリックして「プロパティ」ウィンドウを閉じます。
5. **PreferencesUI** クラスについて、ステップ 4 を繰り返します。
- 以前にホワイト・ペーパーを使用して Java でステージ 1 ツールをカスタマイズした場合は、以下の変更点に注意してください。
 - ファイルのロケーションを変更した場合は、組み込みのカスタマイズを使用してファイルのロケーションを変更する方法の詳細を 164 ページの『ステージ 1 マイグレーション・ツールのカスタマイズ』で参照してください。
 - プロジェクトまたはパッケージを統合した場合、「*How to Consolidate Projects and Packages during Stage 1 of the VisualAge Generator on Java to Enterprise Generation Language Migration Tool*」というタイトルのホワイト・ペーパーで説明しているように、変更を **IBM VisualAge Generator EGL マイグレーション・プロジェクト**、**com.ibm.vgj.mig** パッケージ、**Preferences** クラス、および **applyRenameRulesTo(String,String)** メソッドに限定していれば、カスタマイズ・コードが引き続き適用されます。
 - 以前にホワイト・ペーパーを使用して Smalltalk でステージ 1 ツールをカスタマイズした場合は、以下の変更点に注意してください。
 - ファイルのロケーションを変更した場合は、組み込みのカスタマイズを使用してファイルのロケーションを変更する方法の詳細を 192 ページの『ステージ 1 マイグレーション・ツールのカスタマイズ』で参照してください。
 - プロジェクトまたはパッケージを統合した場合、「*How to Consolidate Projects and Packages during Stage 1 of the VisualAge Generator on Smalltalk to Enterprise Generation Language Migration Tool*」というタイトルのホワイト・ペーパーで説明しているように、変更を **HptEGLMigrationBaseApp** アプリケーション、**VAGenToEGLMapper** クラス、**API** クラス・カテゴリー、**convertProjectNameToWSED** public メソッドおよび **convertApplicationToWSEDPackageName** public メソッドに限定していれば、カスタマイズ・コードが引き続き適用されます。
 - ステージ 2 での構文マイグレーション:
 - マイグレーション・ツールの提供する EGL ソースは、EGL V7.1 の EGL 言語の変更に応じて更新されています。

- Execution time statement build オプションを指定している SQL 関数のマイグレーションで、デフォルト SQL が使用されている場合、および **prepare** ステートメントでソフト・エラーが発生している場合のマイグレーションが改善されました。

EGL 7.1 以降の VAGen マイグレーション・ツールの新機能

EGL 7.1 以降、マイグレーション・ツールに対する変更はありません。

マイグレーション・ツールに関する既知の制限事項

ステージ 1

- ステージ 1 マイグレーションは、Linux または Vista 環境ではサポートされません。
- DB2 バージョン 9.x は、Java 環境のステージ 1 マイグレーションではサポートされません。

ステージ 2 および 3

- VAGen マイグレーション・ウィザードには次の制限事項があります。
 - 進行状況ウィンドウの「取り消し」ボタンは機能しません。タスク・マネージャーを使用しない限り、ステージ 2 または 3 のマイグレーション・ツールを開始後に取り消すことはできません。
- マイグレーション・データベースおよびアプリケーション・データベースを切り換える場合は、切り換えるたびに EGL 開発環境をシャットダウンする必要があります。
- Linux 環境では、ステージ 2 と 3 はサポートされません。

構文のマイグレーション

- マイグレーション・ツールは、SQL レコード構造の中で列名として使用されている SQL キーワードを正しく変換します。ただしマイグレーション・ツールは、レコードの SQL **defaultSelectCondition** 内、または関数の SQL 節内で、列名として使用されている SQL キーワードを正しく処理しません。この解決策として、300 ページの『特殊な処理を必要とする SQL 予約語』の説明に従って、SQL **defaultSelectCondition** または SQL 節を変更します。この項では、SQL キーワードのリストと、SQL **defaultSelectCondition** および SQL 入出力ステートメントに必要な構文の詳細を示しています。
- このプロパティの使用に関する制限について詳しくは、48 ページの『containerContextDependent プロパティ』を参照してください。

第 3 章 未確定状態の処理

対応する EGL ステートメントを作成するのに必要な VisualAge Generator からの一部の情報をマイグレーション・ツールが持っていない状態が多数あります。こうした状態を未確定状態と呼びます。これは、VisualAge Generator からの入力によって対応する EGL ステートメントが変化したり、異なる結果が得られたりするからです。このような未確定状態では、マイグレーション・ツールは VisualAge Generator で意図していたものと一致する EGL ステートメントにマイグレーションできないことがあります。未確定状態では、次に示すマイグレーション・プロセスのどれを行うかによって、作成される EGL ステートメントが異なる場合がよくあります。

- プログラムおよびその関連パーツをマイグレーションするかどうか。マイグレーションする場合は、それらのプログラムをマイグレーションする順序。
- パーツ間マイグレーションを依然として実行できるように、プログラムをマイグレーションしないが必要な関連パーツをマイグレーションするかどうか。
- マイグレーション・ツールがより良い選択を行うことができる情報に限定されるように、関連付けをマイグレーションしないでマイグレーションするかどうか。

プログラムおよびその関連パーツをマイグレーションする方法は、最大の情報が保証されるので、関連付けをマイグレーションする方法として推奨されます。この章に示す表では、次のパーツ型について、関連パーツと一緒にマイグレーションする場合と、関連パーツなしでマイグレーションする場合の違いを説明しています。

- データ項目
- レコード
- テーブル
- マップ・グループおよびマップ
- プログラム
- I/O ステートメントを含む関数
- その他のステートメント
- EZE ワード

これらの表では、こうした未確定状態に関連して起こりうる問題と、これらの問題に対して考えられる解決策も示しています。それぞれの状態に最適な解決策は 1 つではありません。例えば、同じパーツ名のパーツが 2 つある場合には、使用頻度が低い方の名前を変更すれば、コードの他の部分に及ぶ影響が最小限になります。

データ項目の未確定状態の処理

偶数の長さの PACK データ項目

VisualAge Generator: PACK データ項目の長さは、最大 18 までの桁数として指定されます。共用データ項目定義の中、および非共用データ項目のレコード定義の中では、偶数の長さが記録されます。一方、ほとんどのエディター、およびテストと生成に使用される長さは、次に大きな奇数の長さ (最大の場合は 18) です。SQL レ

コード・エディターのみが偶数の長さを表示します。SQL WHERE 文節、または Execution time statement build オプションを指定している SQL ステートメントの中で、偶数の長さの項目がホスト変数として使用されている場合、テストおよび生成機能は偶数の長さの一時変数を作成します。

EGL: DECIMAL プリミティブ型が、VAGen の PACK 型の置換表現です。
VisualAge Generator 互換モードでは、EGL のテストおよび生成機能は VisualAge Generator と同じサポートを提供します。偶数の精度をもつ 10 進項目の場合、テストおよび生成機能はすべてのレコードの精度を 1 ずつ増やし、SQL WHERE 文節または **prepare** ステートメント内では偶数の精度の一時変数を使用します。
VisualAge Generator 互換モードが指定されていない場合、EGL はデータ項目に対して指定された精度を使用します。

マイグレーションに必要な関連パーツ: 適用不可。

表 10. 偶数の長さの PACK データ項目

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マイグレーション・ツールは、VAGen マイグレーション設定の「項目または変数の evensql=y を受け入れない」に基づいてパック・データ項目をマイグレーションします。</p> <p>この設定が選択されていない場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none">項目が SQL 行レコードで使用されていたかどうかに関係なく、共用データ項目定義について VisualAge Generator で指定された偶数または奇数長を使用します。すべてのレコード定義の非共用項目については、VisualAge Generator に指定された偶数または奇数長を使用します。これは、項目が SQL WHERE 文節または prepare ステートメントでホスト変数として使用されることがあるからです。テーブル、関数仮パラメーター、関数戻り値、および関数ローカル・ストレージの非共用項目については、偶数桁の数値を判別する情報がこの場合には VisualAge Generator に記録されていないので、奇数長 (項目が最大長の場合は 18) を使用します。 <p>この設定が選択されている場合、マイグレーション・ツールは、変数のすべての項目について常に奇数長 (項目が最大長である場合は 18) を使用します。マイグレーション・ツールは evensql=y を指定しているあらゆるデータ項目に関して警告メッセージを発行します。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

表 10. 偶数の長さの PACK データ項目 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題 1: 奇数長に <code>evensql = y</code> をマイグレーションする設定を選択すると、問題が発生します。この場合、SQL 列定義と一致しないホスト変数長を使うため、ランタイム・パフォーマンスに影響が出る可能性があります。</p> <p>解決策 1: マイグレーション・ログ情報を参照し、<code>evensql = y</code> が指定されている <code>DataItem</code> パーツがないかどうかを調べます。SQL テーブルおよびビューの定義を検証し、それらの定義が <code>DataItem</code> パーツの定義と一致しているかどうかを判断します。また、<code>DataItem</code> パーツを型定義として指定する変数を使用している SQL の WHERE 文節と <code>prepare</code> ステートメントを確認します。</p> <p>起こりうる問題 2: マイグレーション中に <code>evensql=y</code> を受け入れる設定をクリアして、後で <code>VisualAge Generator</code> 互換モードを使用しないと判断した場合にも、問題が発生します。この場合には、<code>VisualAge Generator</code> 互換モードよりも有効数字が少なくなるので、オーバーフローが発生することがあります。</p> <p>解決策 2: EGL レコード内のすべての 10 進 <code>DataItem</code> パーツ定義とプリミティブ・フィールドを検討し、偶数長の項目がないかどうかを調べます。これらのいずれかの項目にオーバーフローが発生するかどうか評価します。</p>	<p>起こりうる問題: 『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があり、同じ解決策を適用できます。</p>

共用編集とメッセージ

VisualAge Generator: 共用 `DataItem` パーツ定義には、マップと UI レコード両方のデフォルト編集とメッセージを指定できます。

EGL: `DataItem` パーツ定義の編集とメッセージのプロパティは、1 セットだけ存在します。マイグレーション・ツールは、データ項目のマップと UI プロパティをマージします。

マイグレーションに必要な関連パーツ: 適用不可。

表 11. 共用編集とメッセージ

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マイグレーション・ツールは、マップと UI 編集を以下の方法でマージします。</p> <ul style="list-style-type: none"> 各 編集またはメッセージについて、マイグレーション・ツールは次のリスト内の最初に適用される基準に基づいて編集を変換します。 <ul style="list-style-type: none"> UI 編集が指定されていると、マイグレーション・ツールは UI 編集とその関連メッセージ情報を対応する EGL プロパティに変換します。 マップ編集が指定されている場合のみ、マイグレーション・ツールはマップ編集とその関連メッセージを対応する EGL プロパティに変換します。 関連 UI 編集なしで UI メッセージが指定されていると、マイグレーション・ツールは UI メッセージを対応する EGL プロパティに変換します。 関連マップ編集なしでマップ・メッセージが指定されていると、マイグレーション・ツールはマップ・メッセージを対応する EGL プロパティに変換します。 UI およびマップ編集とメッセージ情報が指定されていない場合、マイグレーション・ツールは対応する EGL プロパティを設定しません。通常の EGL のデフォルトが適用されます。 VisualAge Generator では、マップ編集に行末揃えおよび 16 進数編集のみが指定されるので、対応する EGL プロパティの設定に常に使用されます。 数値データ項目に対しては、マイグレーション・ツールにより以下の方法で記号編集のマイグレーションが行われます。 <ul style="list-style-type: none"> 記号編集が指定されていない場合、いずれかの UI 編集が指定されていれば、マイグレーション・ツールは EGL 記号プロパティを先頭に設定します。 UI 編集が指定されていない場合、マイグレーション・ツールでは EGL 記号プロパティを設定しません。 	<p>83 ページの『共用データ項目のマップ編集ルーチン』で後述する場合を除き、マイグレーション・ツールはデフォルトの編集とメッセージを、関連パーツがあってもなくても同じようにマイグレーションします。</p>

表 11. 共用編集とメッセージ (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題 1: 問題は、VisualAge Generator に矛盾するマップ編集と UI 編集が存在し、マップ・レコードと UI レコード間で編集が異なることを本当に意図している場合にのみ発生します。テキスト書式または印刷書式に項目が追加されるまで問題は発生しません。</p> <p>注: VAGen Web トランザクションを使用していない場合は、マップ編集のみが VisualAge Generator に存在し、問題は発生しません。</p> <p>考えられる解決策: VAGen マップ項目の編集とメッセージをリストするコメントを DataItem パーツ定義に追加するほかに、DataItem パーツ定義に対して実行できることはありません。項目をテキスト書式または印刷書式に追加すると、その特定の書式に対しては異なった指定が必要なプロパティを指定変更できます。</p> <p>起こりうる問題 2: EGL VGUI レコードで項目を使用する場合にも問題が発生することがあります。その項目には、VAGen マップ編集からマイグレーションされた追加の編集やメッセージが存在する場合があります。</p> <p>解決策: 編集を必ず検討し、VGUI レコード内の型定義を使用して定義されたフィールドがないかどうか調べてください。</p>	<p>『関連パーツを使用したマイグレーション』欄で説明した問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

共用データ項目のマップ編集ルーチン

VisualAge Generator: 共用データ項目定義で、テーブル、関数、EZEC10 または EZEC11 であるマップ編集ルーチンを指定できます。編集メッセージは、編集ルーチンが EZEC10、EZEC11、またはテーブルである場合のみ使用されます。

EGL: データ項目には、**validatorDataTable** と **validatorFunction** の両方を指定できます。データ項目にはさらに、**validatorDataTableMsgKey** と **validatorFunctionMsgKey** の両方を指定することもできます。

マイグレーションに必要な関連パーツ: テーブルまたは演算部のいずれか。

表 12. 共用データ項目のマッピング編集ルーチン

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>共用データ項目を初めてマイグレーションする場合、マイグレーション・ツールは次のリスト内の最初に適用される基準に基づいて、マッピング編集ルーチンを変換します。</p> <ul style="list-style-type: none"> • <code>editRoutineName</code> が <code>EZEC10</code> または <code>EZEC11</code> であれば、マイグレーション・ツールは <code>validatorFunction</code> プロパティを EGL の同等のシステム・ライブラリ関数に設定します。さらにマイグレーション・ツールは、<code>validatorFunctionMsgKey</code> を編集メッセージ (あれば) に設定します。 • <code>editRoutineName</code> が関数であれば、マイグレーション・ツールは <code>validatorFunction</code> プロパティを設定します。<code>validatorFunctionMsgKey</code> は VisualAge Generator では使用されないため、マイグレーション・ツールが省略します。マイグレーション・ツールはまた、<code>import</code> ステートメントがステージ 3 で作成されるように、データ項目パーツの関連としてこの関数を追加します。 • <code>editRoutineName</code> がテーブルであれば、マイグレーション・ツールは <code>validatorDataTable</code> プロパティを設定します。さらにマイグレーション・ツールは、<code>validatorDataTableMsgKey</code> を編集メッセージ (あれば) に設定します。マイグレーション・ツールはまた、<code>import</code> ステートメントがステージ 3 で作成されるように、データ項目パーツの関連としてこのテーブルを追加します。 • 編集ルーチンが指定されておらず、編集メッセージが指定されている 場合、マイグレーション・ツールは <code>validatorDataTableMsgKey</code> を編集メッセージに設定します。 <p>注: プログラムのコンテキストでマイグレーションする場合でも、共用項目がマップで使用されていない、またはマップでの編集が共用項目定義で指定されたものと異なる場合には、<code>editRoutineName</code> が使用できないことがあります。</p>	<p><code>editRoutineName</code> と同じ名前の関数またはテーブルが使用できない場合、マイグレーション・ツールは次のリスト内の最初に適用される基準に基づいて、マッピング編集ルーチンを変換します。</p> <ul style="list-style-type: none"> • <code>editRoutineName</code> が <code>EZEC10</code> または <code>EZEC11</code> であれば、マイグレーション・ツールは <code>validatorFunction</code> プロパティを EGL の同等のシステム・ライブラリ関数名に設定します。さらにマイグレーション・ツールは、<code>validatorFunctionMsgKey</code> を編集メッセージ (あれば) に設定します。 • 7 文字より長い <code>editRoutineName</code> は関数名であると考えられるので、マイグレーション・ツールは <code>validatorFunction</code> プロパティを設定します。<code>validatorFunctionMsgKey</code> は VisualAge Generator では使用されないため、マイグレーション・ツールが省略します。 • 編集メッセージが指定されている場合、マイグレーション・ツールは <code>validatorDataTable</code> と <code>validatorDataTableMsgKey</code> を設定します。 • そうでなければ、マイグレーション・ツールは <code>validatorFunction</code> プロパティを設定し、エラー・メッセージを出します。 <p>編集ルーチンが指定されておらず、編集メッセージが指定されている 場合、マイグレーション・ツールは <code>validatorDataTableMsgKey</code> を編集メッセージに設定します。</p>

表 12. 共用データ項目のマップ編集ルーチン (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: 関数と DataTable の名前が同じである (通常、異なるサブシステム内で) 場合に限り、問題が起こります。この場合、あるサブシステムが関数を、別のサブシステムが DataTable を使用します。テキスト書式に項目が追加されるまで問題は発生しません。</p> <p>考えられる解決策: 同名のパーツが 2 つ存在しないように、DataTable の名前を変更します。</p> <p>validatorFunction プロパティおよび validatorDataTable プロパティ両方を DataItem パーツに指定します。項目をテキスト書式に追加する場合は、その特定の書式に必要な、validatorFunction または validatorDataTable のどちらかのプロパティを削除します。</p> <p>欠点: 新しい DataTable 名を使用するようにプログラムと書式を変更する必要があります。</p>	<p>起こりうる問題 1: マイグレーション・ツールが予測を誤ると、問題が発生します。</p> <p>考えられる解決策: データ項目定義を訂正します。</p> <p>起こりうる問題 2: 関連パーツを使用したマイグレーション欄にリストされたのと同じ問題も発生することがあります。同じ解決策を使用できます。</p>

共用データ項目の充てん文字

VisualAge Generator: マップ編集のデフォルトの充てん文字は、文字、混合、または DBCS の場合は NULL で、数値の場合はブランク、16 進数の場合は 0 です。UI 編集のデフォルトの充てん文字は、文字、混合、DBCS、ユニコード、および数値の場合はブランクで、16 進数の場合は 0 です。Null は UI レコードでは無効な充てん文字です。マップ編集と UI 編集に別の充てん文字を指定できます。

EGL: DataItem パーツのデフォルト **fillCharacter** プロパティは 1 つだけ存在します。マイグレーション・ツールは、データ項目のマップと UI **fillCharacter** プロパティをマージします。

マイグレーションに必要な関連パーツ: 適用不可。

表 13. 共用データ項目の充てん文字

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>共用データ項目を初めてマイグレーションする場合、マイグレーション・ツールは次のリスト内の最初に適用される基準に基づいて、充てん文字を変換します。</p> <ul style="list-style-type: none"> UI 編集で充てん文字を指定すると、マイグレーション・ツールは文字を EGL fillCharacter プロパティにマイグレーションします。NULL 充てんは VisualAge Generator 内の UI レコードでは無効であるため、マイグレーション・ツールは N to N 変換を行います。 マップ編集で充てん文字を指定すると、マイグレーション・ツールは文字を EGL fillCharacter プロパティにマイグレーションします。マイグレーション・ツールは N を NULL 充てんに変換します。 UI 編集でもマップ編集でも充てん文字を指定しないと、マイグレーション・ツールは EGL fillCharacter プロパティを設定しません。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題: あるタイプの編集を使用して別のタイプのユーザー・インターフェース (書式または VGUI レコード) にマイグレーションされた DataItem パーツに、型定義を使用してフィールドを追加した場合に限り、問題が起こります。</p> <p>考えられる解決策: 書式または VGUI レコードに新規フィールドを追加する際には必ず、fillCharacter プロパティを検討してください。</p>	<p>『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

レコードの未確定状態の処理

再定義レコード

VisualAge Generator: 再定義レコード・タイプは、別のレコードに異なるデータ項目レイアウトを提供します。再定義レコードは、再定義するレコードの名前を指定します。例えば、RecordA は RecordB を再定義する再定義レコードであるとしします。VisualAge Generator は、プログラム内での RecordA の使用方法に基づいて、RecordA が実際に RecordB の再定義であるかどうかを判別します。RecordA が呼び出し先パラメーターとして使用されている場合、RecordA は RecordB の再定義として扱われません。

EGL: RecordA は基本レコードです。再定義情報はプログラム定義内でのみ提供され、RecordA の定義内では提供されません。

マイグレーションに必要な関連パーツ:

- 再定義レコードをマイグレーションする場合: 適用不可。
- プログラムをマイグレーションする場合: 再定義レコード (RecordA)。

表 14. 再定義レコード

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>再定義レコードをマイグレーションする際に、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> 再定義レコード (RecordA) を基本レコードにマイグレーションします。 RecordA が RecordB を再定義したことを示す VAGen 情報コメントを RecordA に組み込みます。 RecordA が RecordB を再定義したことを示す情報メッセージを出します。 	<p>再定義レコードをマイグレーションする際に、マイグレーション・ツールは『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>プログラムをマイグレーションする際に、RecordA が使用可能であれば、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> RecordA が VisualAge Generator 内で RecordB の再定義として扱われている場合、マイグレーション・ツールは RecordA の宣言に redefines プロパティを組み込みます。 RecordA が VisualAge Generator 内で RecordB の再定義として扱われていない場合、マイグレーション・ツールは RecordA の宣言に redefines プロパティを組み込みません。 	<p>プログラムをマイグレーションする際に、RecordA が使用不可ならば、マイグレーション・ツールは RecordA が再定義レコードであることを認識できません。マイグレーション・ツールは、RecordA の宣言に redefines プロパティを組み込みません。</p>
<p>新規使用の場合の考慮事項: RecordA と RecordB を新規プログラム内で使用する必要がある場合に限り、問題が起こります。RecordA が RecordB の再定義として扱われるようにしたい場合は、必ず RecordA に redefines プロパティを組み込んでください。</p>	<p>起こりうる問題: VAGen プログラムが RecordA を再定義として使用する場合には、問題が生じます。マイグレーションの直後は、RecordA の定義と import ステートメントが欠落しているので、プログラムは有効な EGL プログラムではありません。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。RecordA をマイグレーションしてプログラムに import ステートメントを追加すると、プログラムが有効な EGL プログラムに変換されます。ただし、データ域は 2 つ (RecordA 用に 1 つ、RecordB 用に 1 つ) 作成されます。EGL は、検証または生成の際にはこの変更を検出しません。このプログラムは、VisualAge Generator の場合と同じようには実行しません。</p> <p>解決策: 追加のレコードをマイグレーションする場合、またはプログラムに import ステートメントを追加する場合は、レコード定義にある VAGen 情報コメントを検討してください。RecordA が RecordB の再定義であることを示す VAGen 情報コメントがある場合は、RecordA の宣言に redefines プロパティを組み込んでプログラムを更新します。</p> <p>考慮事項: 『関連パーツを使用したマイグレーション』欄にリストしたものと同一、新規使用の場合の考慮事項が適用されます。</p>

レコード内のレベル 77 項目

VisualAge Generator: 作業用ストレージ・レコードにはレベル 77 項目が存在する可能性があります。

EGL: レコードにレベル 77 項目は存在しません。

マイグレーションに必要な関連パーツ:

- 作業用ストレージ・レコードをマイグレーションする場合: 適用不可。
- プログラムをマイグレーションする場合: 1 次作業用ストレージ・レコード。
- 関数をマイグレーションする場合: 作業用ストレージ・レコード。

表 15. レコード内のレベル 77 項目

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>レベル 77 項目を含む作業用ストレージ・レコードをマイグレーションする際に、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none">• レベル 77 項目を含む作業用ストレージ・レコードを 2 つの基本レコードに分割します (作業用ストレージ構造用に 1 つ、レベル 77 項目用に 1 つ)。新しいレベル 77 レコード名は、オリジナルの作業用ストレージ・レコード名にレベル 77 接尾部を付けたものです。レベル 77 接尾部は、VAGen マイグレーション構文を設定することによって指定できます。• オリジナルの作業用ストレージ・レコードと同じファイルに、新しいレベル 77 レコードを配置します。• レベル 77 レコードが作成されていることを示す情報メッセージを出します。	<p>レベル 77 項目を含む作業用ストレージ・レコードをマイグレーションする際に、マイグレーション・ツールは『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>プログラムをマイグレーションする際に、1 次作業用ストレージ・レコードが使用可能で、レベル 77 項目を含んでいる場合、マイグレーション・ツールは新しいレベル 77 レコードのレコード宣言をプログラムに追加します。</p>	<p>プログラムをマイグレーションする際に、1 次作業用ストレージ・レコードが使用不可ならば、マイグレーション・ツールは 1 次作業用ストレージ・レコードがレベル 77 項目を含んでいるかどうか判別できません。マイグレーション・ツールは、レベル 77 レコードのレコード宣言を組み込みません。</p>
<p>関数をマイグレーションする際に、作業用ストレージ・レコードが使用可能ならば、マイグレーション・ツールはその関数内のレベル 77 項目への修飾参照を変更して、新しいレベル 77 レコード名を使用するようにします。</p>	<p>関数をマイグレーションする際に、作業用ストレージ・レコードが使用不可ならば、マイグレーション・ツールは項目名の修飾を変更しません。</p>

表 15. レコード内のレベル 77 項目 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: 同名のレコードが 2 つ存在し (異なるサブシステム内にあると考えられる)、一方のレコード内では項目がレベル 77 項目であり、他方ではそうでない場合に限り、レベル 77 項目に関する問題が起こります。</p> <p>考えられる解決策: その項目を (新規) 共通レコードに移動し、すべての関数内でその項目の修飾を変更します。あるいは、関数内でその項目を修飾しません。</p> <p>新規使用の場合の考慮事項: 新規プログラムの inputRecord プロパティとしてオリジナルの作業用ストレージ・レコードを指定すると、問題が起こる可能性があります。新しいレベル 77 レコードの宣言をさらに追加する必要があるかどうか、必ず検討してください。</p>	<p>起こりうる問題 1: 1 次作業用ストレージ・レコードがレベル 77 を含んでいて、プログラムがレベル 77 を使用していた場合には、問題が起こります。プログラムの検証は、項目定義の欠落のために失敗します。</p> <p>解決策: プログラムにレベル 77 レコードを追加します。</p> <p>起こりうる問題 2: 修飾データ項目が実際には VAGen レベル 77 項目である場合、関数に問題が起こります。</p> <p>解決策: データ項目の正しい修飾子を指定するように関数を変更します。</p> <p>起こりうる問題 3: 『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p> <p>考慮事項: 『関連パーツを使用したマイグレーション』欄にリストしたものと同じ、新規使用の場合の考慮事項が適用されます。</p>

代替仕様レコード

VisualAge Generator: レコードは、別のレコードを代替仕様 (altspec) レコードとして指定できます。例えば、RecordA が altspec として RecordB を指定している場合、RecordB は RecordA の構造を提供します。SQL レコードの場合、RecordB は RecordA の SQL 表とキーのリストも提供します。RecordA がキー項目を指定している場合は、デフォルト選択条件を判別する際にそのキーが RecordB のキーとマージされます。DL/I セグメント・レコードの場合、RecordB 内のフィールド名は DL/I PSB 内のフィールド名でもあります。

EGL: レコード構造を取得するために、レコードに別のレコードを組み込むことができます。組み込まれるものはレコード構造のみです。SQL レコードはそれぞれ、SQL レコードが参照する SQL 表とキーの完全なセットを明示する必要があります。DL/I セグメント・レコードでは、RecordB が DL/I セグメントでもある場合、その RecordB は DL/I PSB 内のフィールド名を指定する **dliFieldName** プロパティを提供できます。また、RecordB が DL/I セグメントでない場合、RecordA は **embed** キーワードをオーバーライドするものとして **dliFieldName** プロパティを提供できます。

マイグレーションに必要な関連パーツ: RecordA が SQL レコードまたは DL/I セグメントである場合、altspec レコードとして指定されたレコード (RecordB) が必要です。

表 16. 代替仕様レコード

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>RecordA が RecordB の代替仕様を指定する SQL レコードであり、RecordB が使用可能であれば、マイグレーション・ツールは RecordA に対して次の処理を行います。</p> <ul style="list-style-type: none"> RecordB 内で指定されている表のリストから、表名のリストを作成します。 次の項目をマージして、キーのリストを作成します。 <ul style="list-style-type: none"> key=yes を指定した RecordB 内の項目 (存在する場合)。 キー項目として指定された RecordA 内の項目 (存在する場合)。 <p>キーの順序は、項目が RecordB の構造に出現する順序です。</p> <ul style="list-style-type: none"> RecordB をポイントする embed キーワードを組み込みます。 RecordA のデフォルト選択条件にある !itemColumnName 変数を、RecordB にある対応する SQL 列名にマイグレーションします。 	<p>RecordA が RecordB の代替仕様を指定する SQL レコードであり、RecordB が使用不可であれば、マイグレーション・ツールは RecordA に対して次の処理を行います。</p> <ul style="list-style-type: none"> tableNames プロパティを <code>###TABLES_NOT_FOUND###</code> に設定します。 keyItems プロパティを、<code>###KEYS_NOT_FOUND###</code> の後に RecordA 内でキー項目として指定されている項目 (あれば) を付けた内容に設定します。 RecordB をポイントする embed キーワードを組み込みます。 RecordA のデフォルト選択条件にある !itemColumnName 変数を、置換せずに !itemColumnName にマイグレーションします。 表とキーを判別できないことを示すエラー・メッセージを出します。 !itemColumnName 変数が存在する場合は、エラー・メッセージを出します。
<p>RecordA が RecordB の代替仕様を指定する DL/I セグメント・レコードであり、RecordB が使用可能であるが DL/I セグメント・レコードではない場合、マイグレーション・ツールは RecordA に関して次の処理を行います。</p> <ul style="list-style-type: none"> RecordB をポイントする embed キーワードを組み込みます。 RecordB 内の名前を変更しなければならない各フィールドごとに、オーバーライド・ステートメントを組み込みます。オーバーライド・ステートメントは、dliFieldName プロパティをオリジナルの VAGen フィールド名に設定して、EGL で DL/I フィールド名を使用できるようにします。 	<p>RecordA が RecordB の代替仕様を指定する DL/I セグメント・レコードであり、RecordB が使用可能ではない場合、マイグレーション・ツールは RecordA に関して次の処理を行います。</p> <ul style="list-style-type: none"> RecordB をポイントする embed キーワードを組み込みます。 dliFieldName 情報を保存するためにオーバーライド・ステートメントが必要かどうかをマイグレーション・ツールが決定できない、というエラー・メッセージを発行します。

表 16. 代替仕様レコード (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題 1: SQL については、RecordB の定義が異なる場合に限り (通常は、別々のサブシステム内で) 問題が生じます。</p> <p>解決策 1: それぞれのサブシステムで RecordA を個別に定義されるように、RecordA の定義を複製します。</p> <p>起こりうる問題 2: DL/I についても、RecordB の定義が異なる場合に限り (通常は、別々のサブシステム内で) 問題が生じます。</p> <p>解決策 2: 解決策は解決策 1 と同じです。</p>	<p>問題 1: SQL レコード RecordA の EGL 検証の結果、「問題」ビューにメッセージが表示されます。</p> <p>解決策 1: RecordA を編集し、RecordB に基づいて適切な表とキーの情報を組み込みます。また、デフォルト選択条件にある !itemColumnName 変数すべてを、RecordB からの対応する SQL 列名に置き換えます。</p> <p>起こりうる問題 2: 予約語であるため名前を変更しなければならないフィールドが RecordB にある場合、またはそれらのフィールドが # 記号または @ 記号で始まっている場合、DL/I セグメント RecordA に関する問題が発生します。この場合、このフィールドがデフォルトの SSA で使用されており、変更後のフィールド名が 8 文字を超えていたり DL/I では無効な文字 (下線など) を含んでいたりする場合、EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策 2: RecordA を編集し、DL/I フィールド名を指定するためのオーバーライド・ステートメントを組み込みます。</p> <p>起こりうる問題 3: 名前変更されたフィールドがデフォルトの SSA 内で使用されるが、無効な DL/I 名である場合にも、DL/I セグメント RecordA について問題が発生します。この場合、プログラム実行時にランタイム・エラーが発生します。</p> <p>解決策 3: RecordA を編集し、DL/I フィールド名を指定するオーバーライド・ステートメントを組み込みます。</p>

同じレコード名をもつ異なる定義

VisualAge Generator: レコードには、現在ワークスペース内にあるプロジェクトとパッケージに基づいた共用データ項目が含まれています。これにより、さまざまなサブシステムやプログラムが、同じレコード名に対して異なる定義を使用できます。

EGL: レコード定義用の **containerContextDependent** プロパティを提供しています。このプロパティを YES に設定すると、プログラムのパーツのネーム・スペースに基づいて、型定義に使用する DataItem パーツを指定することができます。

マイグレーションに必要な関連パーツ: 適用不可。このレコード・プロパティを設定するには、サブシステムすべての VAGen パーツ構造を完全に理解している必要があります。

表 17. 同じレコード名をもつ異なる定義

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マイグレーション・ツールは、レポート定義で containerContextDependent プロパティを YES に設定しません。この機能が必要な場合は、この機能を必要とするそれぞれのレコードごとにこのプロパティを設定する必要があります。</p> <p>注: このプロパティの使用に関する制限については、48 ページの『containerContextDependent プロパティ』を参照してください。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

予約語と UI レコード名

VisualAge Generator: VisualAge Generator に予約語はありません。VisualAge Generator では、UI レコード名または UI レコード内のフィールドの先頭文字として # 記号または @ 記号が使えます。

EGL: EGL には予約語があります。さらに EGL の場合は、レコード名の先頭文字として # 記号または @ 記号を使用することはできません。UI レコード名は予約語にすることはできず、また名前の先頭文字を # 記号や @ 記号にすることはできません。UI レコード内のフィールド名は予約語にすることはできず、また名前の先頭文字を # 記号や @ 記号にすることはできません。マイグレーション・ツールの拡張予約語リストにその UI レコード名またはフィールド名がある場合も、ネーム解決競合が起こる可能性があります。

マイグレーションに必要な関連パーツ: 適用不可。

表 18. 予約語と UI レコード名

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>UI レコードをマイグレーションするとき、そのレコード名がマイグレーション・ツールの拡張予約語リストにあるか、# または @ 記号で始まる場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> レコード名の冒頭に名前変更接頭部を組み込んで UI レコード名を変更します。これは、他の VAGen レコードに関してマイグレーション・ツールが行う名前変更処理と同じです。名前変更接頭部は、VAGen マイグレーション構文を設定することによって指定できます。 alias プロパティを組み込み、UI レコードのオリジナルの VAGen 名に設定します。 名前変更済みの UI レコードとマイグレーションのステージ 3 で一致するように、UI レコードの .egl ファイル名を変更します。 警告メッセージを出します。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

表 18. 予約語と UI レコード名 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>UI レコード内のフィールドをマイグレーションするとき、そのフィールド名がマイグレーション・ツールの拡張予約語リストにあるか、# または @ 記号で始まる場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> フィールド名の冒頭に名前変更接頭部を組み込んでフィールドを変更します。これは、他の VAGen レコード内のフィールドに関してマイグレーション・ツールが行う名前変更処理と同じです。 フィールドに alias プロパティを組み込み、そのフィールドのオリジナルの VAGen 名に設定します。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>マイグレーション・ツールは、名前を変更するためには、他のレコードと同様に UI レコードを扱います。ツールは、UI レコードに対するすべての参照に、EGL パーツ名を使用します。この対象には、次の場所での参照が含まれます。</p> <ul style="list-style-type: none"> プログラム・パーツ: <ul style="list-style-type: none"> 先頭 UI レコード レコード宣言 UI レコード・パーツ - Link パラメーター内の先頭 UI レコード 演算部 - ステートメントにおけるレコードの使用 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題: なし。 alias プロパティは VAGen と同じ JSP 名を設定します。</p>	<p>起こりうる問題: なし。 alias プロパティは VAGen と同じ JSP 名を設定します。</p>

テーブルの未確定状態の処理

予約語とテーブル名

VisualAge Generator: VisualAge Generator に予約語はありません。VAGen テーブル名では # 記号または @ 記号は無効です。

EGL: EGL には予約語があります。さらに EGL の場合は、パーツ名の先頭文字として # 記号または @ 記号を使用することはできません。DataTable 名が予約語であってはなりません。マイグレーション・ツールの拡張予約語リストにその DataTable 名がある場合も、ネーム解決競合が起こる可能性があります。

マイグレーションに必要な関連パーツ: 適用不可。

表 19. 予約語とテーブル名

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マイグレーション・ツールは、テーブルの名前を自動的に変更しません。そのテーブル名がマイグレーション・ツールの拡張予約語リストにある場合、マイグレーションのステージ 1 で使用されるマイグレーション・ツールがエラー・メッセージを出します。テーブル名を変更しない場合は、マイグレーションのステージ 2 で使用されるマイグレーション・ツールもエラー・メッセージを出します。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題: DataTable 名がマイグレーション・ツールの拡張予約語リスト上にある場合に限り、問題が起こります。EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策: マイグレーションの前に VisualAge Generator のテーブルの名前を変更するか、マイグレーションの後に EGL の DataTable の名前を変更します。</p> <p>VisualAge Generator 内で名前を変更する場合は、プログラム、マップ、関数、UI レコード、およびデータ項目定義の中で、そのテーブルへのすべての参照を必ず変更してください。EGL で名前を変更する場合は、DataTable の名前とその名前に対するすべての参照を変更する必要があります。この対象には、次の場所での参照が含まれます。</p> <ul style="list-style-type: none"> プログラムの use 宣言ステートメント プログラムおよび関数のロジック・ステートメント データ項目 validatorDataTable プロパティ 書式フィールド validatorDataTable プロパティ VGUI レコードの validatorDataTable プロパティ <p>生成される DataTable の名前としてオリジナルのテーブル名を保持する必要がある場合は、alias プロパティをオリジナルの DataTable 名に設定します。</p> <p>alias プロパティを指定しない場合は、CICS プログラム定義など、EGL 以外からの DataTable 名に対するすべての参照を必ず変更してください。</p>	<p>『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

マップ・グループとマップの未確定状態の処理

予約語と FormGroup 名

VisualAge Generator: VisualAge Generator に予約語はありません。VAGen マップ・グループ名に # 記号または @ 記号は使えません。

EGL: EGL には予約語があります。さらに EGL の場合は、パーツ名の先頭文字として # 記号または @ 記号を使用することはできません。FormGroup 名は予約語で

あってはなりません。マイグレーション・ツールの拡張予約語リストにその FormGroup 名がある場合も、ネーム解決競合が起こる可能性があります。

マイグレーションに必要な関連パーツ: 適用不可。

表 20. 予約語と FormGroup 名

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
マイグレーション・ツールは FormGroup の名前を自動的に変更しません。マップ・グループ名がマイグレーション・ツールの拡張予約語リストにある場合、マイグレーションのステージ 1 で使用されるマイグレーション・ツールがエラー・メッセージを出します。マップ・グループ名を変更しない場合は、マイグレーションのステージ 2 で使用されるマイグレーション・ツールもエラー・メッセージを出します。	マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。
<p>起こりうる問題: FormGroup 名がマイグレーション・ツールの拡張予約語リストにある場合に限り、問題が起こります。EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策: マイグレーションの前に VisualAge Generator のマップ・グループの名前を変更するか、マイグレーションの後に EGL の FormGroup の名前を変更します。VisualAge Generator 内でマップ・グループの名前を変更する場合は、そのマップ・グループに属するマップすべての名前を必ず変更してください。また、すべてのプログラム定義にあるマップ・グループの参照をすべて変更してください。EGL で FormGroup の名前を変更する場合は、FormGroup の名前と、その名前に対するすべての参照 (プログラム use 宣言ステートメント内での参照など) を変更する必要があります。生成した FormGroup の名前としてオリジナルのマップ・グループ名を保持する必要がある場合は、オリジナルのマップ・グループ (FormGroup) 名に alias プロパティを設定します。alias プロパティを指定しない場合は、CICS プログラム定義など、EGL 以外による FormGroup 名の参照すべてを必ず変更してください。</p>	『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。

マップ・グループと FormGroup の要件

VisualAge Generator: マップ・グループは、浮動域仕様が存在する場合のみ必要です。

EGL: FormGroup は、フォームを格納するために常に必要です。

マイグレーションに必要な関連パーツ: マップ・グループと、マップ・グループ内のすべてのマップ。

表 21. マップ・グループと FormGroup の要件

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マップ・グループが存在しない場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • 同じマップ・グループ名を持つすべてのマップに対して、FormGroup を作成します。 • 同じ FormGroup 用のすべての書式を、同じ EGL ファイルに書き込みます。 • 単一ファイル・マイグレーションによるマイグレーションを行っていない場合は、FormGroup 定義の中で書式をネストします。 • 単一ファイル・モードでマイグレーションを行っている場合は、書式をネストするために FormGroup の編集が必要であることを示すエラー・メッセージが出されます。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。ただし、マップ・グループとそのグループ内のすべてのマップが同じマイグレーション・セットに入っていないければ、問題が生じる可能性があります。</p>
<p>起こりうる問題: なし。マップ・グループのマップはすべて、同じマイグレーション・セットに含まれている必要があります。マイグレーション・セットは生成される内容の表現なので、マイグレーション・セットにはマップ・グループ内のマップをすべて組み込む必要があります。</p> <p>単一ファイル・モードでマイグレーションを行っている場合は、マップ・グループ内のマップすべてを必ず同じ外部ソース形式ファイルに組み込んでください。</p>	<p>起こりうる問題: 同じマップ・グループ名のマップがすべて同じマイグレーション・セット (単一ファイル・モードのマイグレーションの場合は、外部ソース形式ファイル) に含まれていない場合は、FormGroup にすべての書式が組み込まれません。</p> <p>考えられる解決策 1: マイグレーション・セットが、同じマップ・グループ名をもつすべてのマップを含んでいることを確認します。</p> <p>考えられる解決策 2: 欠落している書式を EGL ファイルに追加し、FormGroup 定義の中でこれらのフォームをネストします。</p>

浮動域と開始位置

VisualAge Generator: VisualAge Generator では、装置サイズが同じであるさまざまな装置タイプに対して、異なる浮動域サイズと開始位置を指定することができます。

EGL: EGL FormGroup と印刷書式は、装置サイズのみを指定します。EGL テキスト・フォームは、装置サイズとフォーム・サイズの両方を指定します。EGL の場合、1 つの装置サイズに対して指定できるマージン仕様は 1 セットだけです。

マイグレーションに必要な関連パーツ: 適用不可。

表 22. 浮動域と開始位置

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> 複数の装置が同じ装置サイズのもので、異なる浮動域サイズまたは開始位置を使用している場合は、エラー・メッセージが出されます。 それぞれの VAGen 浮動域仕様ごとに、同等な EGL 装置サイズとマージン仕様を組み込みます。複数の VAGen 装置が変換によって同じ EGL 装置のサイズ仕様およびマージン仕様を持つようになった場合、マイグレーション・ツールは EGL に関して 1 つのエントリのみを組み込みます。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題: 同じ装置サイズをもつ複数の装置が、異なる浮動域サイズまたは開始位置を VisualAge Generator 内で指定している場合に限り、問題が起こります。EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。</p> <p>考えられる解決策: エラー・メッセージを検討します。FormGroup 定義を編集して、この装置サイズに必要な、浮動域サイズと開始位置を 1 つ指定します。</p>	<p>『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

マップ名とヘルプ・マップ名

VisualAge Generator: マップ名は、マップ・グループとマップ名からなる 2 部構成の名前です。プログラムのメインのマップ・グループとヘルプ・マップ・グループは、両方とも同名のマップを含むことができます。例えば、プログラム X のメインのマップ・グループ GROUPA とヘルプ・マップ・グループ GROUPH は、それぞれ MAP1 という名前のマップを含むことができます。マップ名は 8 文字に制限されます。マップ名はプログラム名と同じでも構いません。VAGen マップ・グループ名の中では # 記号と @ 記号は無効ですが、マップ名のマップ名部分では # 記号も @ 記号も使用できます。

EGL: 書式名は FormGroup 名を含みません。代わりに、FormGroup パーツ内でテキスト書式と印刷書式が定義されます。また、EGL では、メイン FormGroup とヘルプ FormGroup の書式名がすべて固有である (1 つのプログラムの 2 つの FormGroup で書式名が重複していない) ことが必要です。EGL は、書式名がプログラム名と同じであることを許容しません。さらに、EGL では、フォーム名が予約語であること、またフォーム名の先頭文字として # 記号と @ 記号を使用することは許されません。EGL は、定義時に 8 文字を超える書式名を許します。生成時には、別名が指定されていれば、別名が書式名として使用されます。COBOL 生成では、alias のフォーム名は 8 文字に制限されます。生成したコードのメイン FormGroup とヘルプ FormGroup では、名前の重複が許可されます。

マイグレーションに必要な関連パーツ: マップ・グループをマイグレーションするには、プログラムとそのマップ・グループ、ヘルプ・マップ・グループ、および両マップ・グループ内のマップすべてが必要です。

表 23. マップ名とヘルプ・マップ名

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>メイン・マップ・グループまたはヘルプ・マップ・グループのいずれかを最初にマイグレーションするプログラムに基づいて、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> マイグレーション・ツールの拡張予約語リストや、名前のマップ名部分の先頭文字として # または @ 記号が使用されていることに起因するマップ名の変更を実行します。プログラムのメイン・マップ・グループ、およびヘルプ・マップ・グループにあるマップの名前が、必要に応じて変更されます。 プログラムのヘルプ・マップ・グループにあるすべての マップの名前を検査して、メイン・マップ・グループと重複する名前がないかどうかを調べます。 プログラム名と、そのプログラムのメインのマップ・グループとヘルプ・マップ・グループ内のすべてのマップの名前を比較します。 <p>ヘルプ・マップ・グループにあるマップの名前が、メインのマップ・グループにあるどのマップとも同じでなければ、マイグレーション・ツールはヘルプ・マップ名を変更しません。</p> <p>ヘルプ・マップ・グループにあるマップの名前が、プログラムのメインのマップ・グループにあるいずれかのマップと同じである場合は、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> ヘルプ・マップが定数のみを含む場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> ヘルプ・マップの名前を、<code>helpMapName</code> にお客様指定の接尾部を付けた名前に変更します。 オリジナルのヘルプ・マップ名に alias プロパティを組み込みます。 すべてのマップの helpForm プロパティを変更して、新しいヘルプ・マップ名を指定します。 ヘルプ・マップが変数のみを含む場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> エラー・メッセージを出します。 マップの名前を変更しません。 <p>これは、そのプログラムのメイン・マップ・グループとしてそのヘルプ・マップ・グループを指定する別の何らかのプログラムが、そのマップを使用している可能性があるからです。</p> <p>ヘルプ・マップ・グループ内のマップの内容が定数フィールドのみであり、マップ名がプログラム名と同じである場合、マイグレーション・ツールはヘルプ・マップの名前を変更します。行われる処理は、ヘルプ・マップ・グループとメインのマップ・グループ内でマップ名が競合する場合に、名前変更を実行できるときの処理と同じです。</p> <p>ヘルプ・マップ・グループにあるマップが変数フィールドを含み、そのマップの名前がプログラム名と同じである場合、またはメインのマップ・グループにあるマップの名前がプログラムと同じである場合は、マイグレーション・ツールはマップの名前を変更しません。行われる処理は、ヘルプ・マップ・グループとメインのマップ・グループ内でマップ名が競合する場合に、名前変更を実行できないときの処理と同じです。</p>	<p>マップ・グループの名前を変更する際に、プログラムが使用できなければ、マイグレーション・ツールは 2 つのマップ・グループが関連しているかどうかを判別できず、マップ・グループがヘルプ・マップ・グループとして指定されているかどうかも判別できません。マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> マイグレーション・ツールの拡張予約語リストや、# または @ 記号に起因するマップ名の変更を実行します。 そのほかには、ヘルプ・マップの名前変更に関する検査を行いません。

表 23. マップ名とヘルプ・マップ名 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題 1: FormGroup があるプログラムではメイン FormGroup として使用され、別のプログラムではヘルプ FormGroup として使用されている場合に、問題が起こる可能性があります。</p> <p>考えられる解決策: ヘルプ FormGroup を 2 つの FormGroup (1 つはヘルプ書式のみを含み、もう 1 つは変数フィールドのある書式を含む) に分割します。ヘルプ書式のみを含む FormGroup を、オリジナル・プログラムのヘルプ FormGroup として指定します。変数フィールドのある書式を含む FormGroup をメイン FormGroup として指定し、ヘルプ書式のみを含む FormGroup を 2 つ目のプログラムのヘルプ FormGroup として指定します。</p> <p>起こりうる問題 2: ヘルプ FormGroup 内のマップが変数フィールドを含んでおり、メイン FormGroup 内のマップと同じ名前である場合に、問題が起こります。</p> <p>考えられる解決策: 問題 1 の考えられる解決策と同じです。</p> <p>起こりうる問題 3: 同じヘルプ FormGroup が複数のプログラムによって共有されている場合に、問題が起こる可能性があります。この場合、マイグレーション・ツールは、さまざまなプログラムに対して、名前変更を必要とするヘルプ書式すべての名前を変更しないことがあります。</p> <p>考えられる解決策: ヘルプ FormGroup の中で、ヘルプ・マップ接尾部を名前に追加して、必要なすべての書式の名前を変更します。alias プロパティを組み込んで、生成時に使用するオリジナルのヘルプ・マップ名を指定します。すべての FormGroup 内の対応するすべてのテキスト書式を変更して、新しいヘルプ書式名を指定します。</p>	<p>起こりうる問題: FormGroup がプログラム内で使用されており、メイン FormGroup とヘルプ FormGroup 内の書式名の間に競合がある場合に限り、問題が起こります。</p> <p>考えられる解決策: 『関連パーツを使用したマイグレーション』に示したのと同じ解決策が適用されます。</p>

数値変数フィールド

VisualAge Generator: マップの数値フィールドの長さは 1 つです。すべての数字、小数点、符号、通貨記号、および数字分離記号が収まるだけの長さにする必要があります。ただし、実行時にフィールドの長さが不足している場合、VisualAge Generator は通貨記号と数字分離記号を省略します。VisualAge Generator は、数値が正の場合にも符号を省略します。使用可能なスペースに収めるために、VisualAge Generator は必要に応じて高位の桁を除去します。

EGL: 書式の変数フィールドは、桁数や小数部などの型定義と、データが書式に占めるスペースを指定する **fieldLen** プロパティをいずれも指定します。実行時にフィールド長が不足していて、すべての数字と書式制御文字を収容できない場合、EGL はランタイム・メッセージを出します。

マイグレーションに必要な関連パーツ: 適用不可。

表 24. 数値変数フィールド

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マップの数値フィールドをマイグレーションする際に、マイグレーション・ツールは長さとして fieldLen を以下の方法で設定します。</p> <ul style="list-style-type: none"> マイグレーション・ツールは、常に、fieldLen を VisualAge Generator の変数フィールドに指定されているものと同じ長さに設定します。 マイグレーション・ツールは、型定義の長さとし小数部を以下の方法で設定します。 <ul style="list-style-type: none"> 変数フィールドが小数部を指定していない場合、マイグレーション・ツールは型定義の長さを fieldLen に設定します。 変数フィールドが小数部を指定している場合、マイグレーション・ツールは、小数点を入れるために、型定義の長さを fieldLen から 1 を減算した値に設定します。この技法により、実行時に発生するオーバーフローの問題が回避されます。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題: 書式のフィールド長が不足していて、実行時に数字、小数点、符号、通貨記号、および数字分離記号がすべて収まらない場合、EGL はランタイム・エラー・メッセージを出します。</p> <p>解決策: 実行時に発生する可能性がある最大の数値、および指定した書式制御文字すべてが収まるように、十分な大きさの fieldLen を指定して書式定義を変更します。</p>	<p>『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

マップ変数フィールドと編集ルーチン

VisualAge Generator: マップ変数には編集ルーチンを指定でき、このルーチンはテーブル、関数、EZEC10、または EZEC11 です。編集メッセージは、編集ルーチンが EZEC10、EZEC11、またはテーブルである場合のみ使用されます。

EGL: 書式フィールドには、**validatorDataTable** と **validatorFunction** の両方を指定できます。書式フィールドにはさらに、**validatorDataTableMsgKey** と **validatorFunctionMsgKey** の両方を指定できます。

マイグレーションに必要な関連パーツ: テーブルまたは演算部のどちらか。

表 25. 変数マップ・フィールドと編集ルーチン

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マップを初めてマイグレーションする場合、マイグレーション・ツールは次のリスト内の最初に適用される基準に基づいて、編集ルーチンを変換します。</p> <ul style="list-style-type: none"> • <code>editRoutineName</code> が EZEC10 または EZEC11 であれば、マイグレーション・ツールは <code>validatorFunction</code> プロパティを EGL の同等のシステム・ライブラリー関数に設定します。さらにマイグレーション・ツールは、<code>validatorFunctionMsgKey</code> を編集メッセージ (あれば) に設定します。 • <code>editRoutineName</code> が関数であれば、マイグレーション・ツールは <code>validatorFunction</code> プロパティを設定します。<code>validatorFunctionMsgKey</code> は VisualAge Generator では使用されないため、マイグレーション・ツールが省略します。 • <code>editRoutineName</code> がテーブルであれば、マイグレーション・ツールは <code>validatorDataTable</code> プロパティを設定します。さらにマイグレーション・ツールは、<code>validatorDataTableMsgKey</code> を編集メッセージ (あれば) に設定します。 	<p><code>editRoutineName</code> と同じ名前の関数またはテーブルが使用できない場合、マイグレーション・ツールは次のリスト内の最初に適用される基準に基づいて、編集ルーチンを変換します。</p> <ul style="list-style-type: none"> • <code>editRoutineName</code> が EZEC10 または EZEC11 であれば、マイグレーション・ツールは <code>validatorFunction</code> プロパティを EGL の同等のシステム・ライブラリー関数名に設定します。さらにマイグレーション・ツールは、<code>validatorFunctionMsgKey</code> を編集メッセージ (あれば) に設定します。 • 7 文字より長い <code>editRoutineName</code> は関数名であると考えられるので、マイグレーション・ツールは <code>validatorFunction</code> プロパティを設定します。<code>validatorFunctionMsgKey</code> は VisualAge Generator では使用されないため、マイグレーション・ツールが省略します。 • 編集メッセージが指定されている場合、マイグレーション・ツールは <code>validatorDataTable</code> と <code>validatorDataTableMsgKey</code> を設定します。 • 編集メッセージが指定されていない場合、マイグレーション・ツールは <code>validatorFunction</code> プロパティを設定し、エラー・メッセージを出します。
<p>起こりうる問題: 関数と <code>DataTable</code> の名前が (通常、異なるサブシステム内で) 同じであり、2 つのプログラムが同じ <code>FormGroup</code> を (通常、同じサブシステム内で) 共用しており、かつ一方のプログラムが関数の使用を予期し、他方のプログラムが <code>DataTable</code> の使用を予期している場合に限り、問題が起こります。</p> <p>考えられる解決策: <code>FormGroup</code> を共用するプログラムを検討します。この状態が生じる場合は、<code>validatorDataTable</code> を使用する別個の <code>FormGroup</code> を作成します。</p> <p>欠点: 保守する <code>FormGroup</code> が 2 つになります。同一の書式を共通のファイルに移動し、それぞれの <code>FormGroup</code> 内で共通書式をポイントする <code>use formName</code> ステートメントを指定することにより、この欠点を最小限にすることができます。</p>	<p>起こりうる問題: マイグレーション・ツールが予測を誤った場合に限り、問題が起こります。マイグレーション・ツールが関数を指定したときに、この書式を使用するいずれかのプログラムが <code>DataTable</code> を予期している可能性があります。</p> <p>考えられる解決策: エラー・メッセージのあるマップの使用方法を検討します。</p>

マップ・フィールドと数値ハードウェア属性

VisualAge Generator: VisualAge Generator は、文字定数フィールド、文字変数フィールド、および数値変数フィールドに対して、数値ハードウェア属性をサポートします。数値ハードウェア属性により、ユーザーが変数フィールドに数値以外のデータを入力することを防止できます。

EGL: EGL は、文字変数フィールドの **isDecimalDigit** プロパティのみをサポートします。数値フィールドには、有効な数字と、符号や小数点などの書式制御文字のみがフィールドに入力されるように保証するソフト編集機能が備わっています。

マイグレーションに必要な関連パーツ: 適用不可。

表 26. マップ・フィールドと数値ハードウェア属性

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> 数値ハードウェア属性を指定したマップ上の文字変数に関して、ツールは isDecimalDigit プロパティを YES に設定します。 マップにある文字定数に対しては、ツールは常に isDecimalDigit プロパティを省略します。 マップにある数値変数フィールドに対しては、ツールは常に isDecimalDigit プロパティを省略します。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>潜在的な問題: 実行時の動作に若干の変更があります。ユーザーは、数値以外のデータを数値フィールドに入力できるようになりました。しかし、これを行うと、EGL がランタイム・エラー・メッセージを発行します。また、ユーザーは、isDecimalDigit プロパティを YES に設定した文字フィールドに、記号や小数点を入力することはできません。</p> <p>考えられる解決策: VAGen 生成のコードから EGL 生成のコードに変更するときこの相違点が予想されることを、ユーザーに知らせることを検討してください。文字フィールドが、記号または小数点を含む必要がある場合は、基本タイプを数値に変更し、isDecimalDigit プロパティを削除してください。</p>	<p>『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

マップ配列と属性

VisualAge Generator: VisualAge Generator では、1 つの配列の要素に対して、異なる属性を使用できます (以下の属性を含む)。

- 入力必須
- 入力時充てん必須
- 数値ハードウェア属性
- ライト・ペン検出

ただし、これらの属性は、一般的に配列のすべての要素で同じ値に設定されます。

EGL: EGL では、配列項目の以下のプロパティのみをオーバーライドできます。

- フィールド表示プロパティ:
 - color
 - highlight
 - intensity

- protect
- modified
- outline
- cursor,
- position
- value

マイグレーションに必要な関連パーツ: 適用不可。

表 27. マップ配列と属性フィールド

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マイグレーション・ツールは、配列の先頭要素 (配列指標 1) に対して次のプロパティを使用して、EGL と同等のプロパティを設定します。</p> <ul style="list-style-type: none"> • 入力必須 • 入力時充電必須 • 数値ハードウェア属性 • ライト・ペン検出 <p>EGL は、配列の先頭要素のプロパティを、配列のすべての要素に使用します。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題: 配列の要素に異なる属性を使用していた場合に限り、問題が起きます。</p> <p>考えられる解決策: 配列の先頭要素のプロパティを最も制約の小さい値に変更し、配列の各要素が必要な基準を満たしているかどうか検査するロジックを validatorFunction プロパティで指定された関数に追加します。また、実行時の書式の外観に違いが生じる場合は、ユーザーにそのことを知らせてください。</p>	<p>『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

名前なしマップ変数フィールド

VisualAge Generator: VisualAge Generator では、マップで名前なし変数フィールドを使用できます。生成時に、名前なし変数フィールドは定数に変換されます。プログラムと関数は、名前なし変数フィールドを参照することはできません。

EGL: EGL は、名前なし変数フィールドが書式に存在することを許容しません。

マイグレーションに必要な関連パーツ: 適用不可。

表 28. 名前なしマップ変数フィールド

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マップにある名前なし変数フィールドに対して、マイグレーション・ツールは次のいずれかの値が指定されているかどうかを検査します。</p> <ul style="list-style-type: none"> • 初期値 • Protect = yes • Cursor = yes • 「アウトラインなし」以外のアウトライン • 「強調表示なし」以外の強調表示 <p>リストされた値のいずれかが指定されている場合、マイグレーション・ツールは対応する EGL プロパティを使用して定数フィールドを作成し、警告メッセージを出します。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>名前なし変数フィールドにこれらのプロパティが指定されていない場合 (またはデフォルト値のみが指定されている場合)、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • 書式に定数フィールドを作成しません。 • 警告メッセージを出します。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題: なし。VisualAge Generator ではこのフィールドは参照できません。</p>	<p>起こりうる問題: なし。</p>

無保護マップ定数

VisualAge Generator: VisualAge Generator は、マップ上で無保護定数の使用をサポートします。テストおよび生成時に、無保護定数は、保護が自動スキップに設定されている場合と同様に処理されます。

EGL: EGL は、書式上で無保護定数の使用をサポートしません。テキスト書式の定数の場合、EGL は、**protect** プロパティを **protect** または **skipProtect** のいずれかに設定することをサポートします。印刷書式の場合、EGL は **protect** プロパティをサポートしません。

マイグレーションに必要な関連パーツ: 適用不可。

表 29. 無保護マップ定数

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>書式をマイグレーションする際に、マイグレーション・ツールは、無保護定数フィールドに対して次の処理を行います。</p> <ul style="list-style-type: none"> 書式がテキスト書式である場合、マイグレーション・ツールは EGL の protect プロパティを skipProtect に設定して、エラー・メッセージを出します。 書式が印刷書式である場合、マイグレーション・ツールは protect プロパティを省略し、メッセージを出しません。protect プロパティは、EGL の印刷書式では使用されません。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
起こりうる問題: なし。	起こりうる問題: なし。

row=0, column=0 にあるフィールド

VisualAge Generator: VisualAge Generator バージョン 4.5 は、システム共通プロダクトまたは VisualAge Generator の旧リリースで row=0, column=0 に配置されたフィールドを許容します。ただし、VisualAge Generator バージョン 4.5 にはこの位置にフィールドを作成する手段はありません。 row=0, column=0 に配置されたフィールドの属性設定情報は設定できません。

EGL: EGL は、row=0, column=0 に配置されたフィールドをサポートしません。すべてのフィールドに属性バイトが含まれている必要があります。

マイグレーションに必要な関連パーツ: 適用不可。

表 30. row=0, column=0 にあるフィールド

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>書式をマイグレーションする際に、フィールドが row=0, column=0 に配置されている場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> フィールドが定数フィールドであり、値の先頭文字がブランクの場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> 値から先頭文字を除去し、フィールド長を 1 だけ減らします。 position プロパティを [1,1] に設定します。 フィールドにデフォルト表示プロパティを組み込みます。 警告メッセージを出します。 フィールドが定数フィールドであり値の先頭文字がブランクではない場合、またはフィールドが変数フィールドである場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> 値もフィールド長も変更しません。 position プロパティを [0,0] に設定します。 フィールドにデフォルト表示プロパティを組み込みます。 エラー・メッセージを出します。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題 1: フィールドが変更不可で position=[0,0] にある場合は、EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策 1: 書式を修正して、フィールドの位置を変更します。フィールドの属性バイトが入る場所を確保するために、他のフィールドを移動したり、定数の位置を変更したりする必要が生じることがあります。また、デフォルト表示プロパティを検討して、正しい色、強調表示などが使用されていることを確認してください。</p> <p>起こりうる問題 2: 定数フィールドが position=[1,1] に変更された場合に、デフォルト表示プロパティが原因で実行時の外観に相違点が生じる可能性があります。</p> <p>解決策 2: マイグレーションの警告メッセージを検討し、マイグレーション・ツールがフィールドの位置を調整した箇所の書式を必ずテストしてください。</p>	<p>起こりうる問題: 『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

プログラムの未確定状態の処理

プログラム名と予約語

VisualAge Generator: VisualAge Generator に予約語はありません。VAGen プログラム名で # 記号と @ 記号は使えません。

EGL: EGL には予約語があります。さらに EGL の場合は、パーツ名の先頭文字として # 記号または @ 記号を使用することはできません。プログラム名が予約語であってはなりません。マイグレーション・ツールの拡張予約語リストにそのプログラム名がある場合も、ネーム解決競合が起こる可能性があります。

マイグレーションに必要な関連パーツ: 適用不可。

表 31. プログラム名と予約語

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
マイグレーション・ツールはプログラムの名前を自動的に変更しません。そのプログラム名がマイグレーション・ツールの拡張予約語リストにある場合、マイグレーションのステージ 1 で使用されるマイグレーション・ツールがエラー・メッセージを出します。プログラム名を変更しない場合は、マイグレーションのステージ 2 で使用されるマイグレーション・ツールもエラー・メッセージを出します。	マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。
起こりうる問題: プログラム名がマイグレーション・ツールの拡張予約語リストにある場合に限り、問題が起こります。EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。 解決策: プログラムの名前を変更します。この作業は、VisualAge Generator またはマイグレーション後の EGL のどちらでも行うこともできます。EGL でプログラム名を変更する場合は、プログラム名と、その名前に対するすべての参照 (call 、 transfer 、 show の各ステートメントでの参照や、リンクージ・オプション・パーツでの参照など) を変更する必要があります。また、このプログラムに対応するバインド制御パーツ、またはリンク・エディット・パーツの名前も変更します。生成したプログラムの名前としてオリジナルのプログラム名を保持する必要がある場合は、オリジナルのプログラム名に alias プロパティを設定します。 alias プロパティを指定しない場合は、CICS プログラム定義およびトランザクション定義など、EGL 以外からのプログラム名に対するすべての参照を必ず変更してください。	『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。

プログラム内の暗黙データ項目

VisualAge Generator: VisualAge Generator では、暗黙データ項目 (レコード、マップ、テーブル、呼び出し先パラメーター・リスト、関数仮パラメーター・リスト、または関数ローカル・ストレージ内で明示的に定義されていない項目) を使用できます。ただし、暗黙的な項目の使用は、一般的に好ましくありません。

EGL: EGL は、暗黙データ項目の使用を許容しません。

マイグレーションに必要な関連パーツ: 適用不可。

表 32. プログラム内の暗黙データ項目

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
マイグレーション・ツールは、暗黙項目の定義を自動的に作成しません。プログラムが暗黙項目を許容している場合、マイグレーションのステージ 2 で使用されるマイグレーション・ツールが警告メッセージを出します。	マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。
<p>起こりうる問題: プログラムが実際に暗黙項目を使用する場合に限り、問題が起こります。暗黙項目を許容するプログラムがないかどうか、「TO DO」リスト・ログを検討してください。プログラムが、実際に暗黙的な項目として使用される場合、EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策: VisualAge Generator または EGL のどちらでも、プログラムに暗黙項目の定義を追加できます。VAGen による検証を行うと、暗黙項目に必要な定義が表示されます。マイグレーションのステージ 1 の実行前に暗黙項目を作成する場合に役立つホワイト・ペーパーについては、19 ページの『参照』を参照してください。</p>	『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。

関連プログラム・パーツ

VisualAge Generator: VisualAge Java の場合は複数のプロジェクトとパッケージ、VisualAge Smalltalk の場合は複数の構成マップとアプリケーションに、プログラムの関連パーツが存在する可能性があります。

EGL: 複数のプロジェクト、フォルダー、パッケージ、およびファイルに、プログラムの関連パーツが存在する可能性があります。

マイグレーションに必要な関連パーツ: プログラムの場合: すべての関連パーツ。

注: **import** ステートメントについて詳しくは、45 ページの『EGL のビルド・パスと import ステートメント』を参照してください。

表 33. 関連プログラム・パーツ

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> • EGL ファイルの配置先となるパッケージを指定する、package ステートメントを組み込みます。 • 現行ファイル内のパーツが必要とするが、現行ファイルのパッケージには含まれていない関連パーツを含むパッケージの import ステートメントを、EGL ファイルに組み込みます。この import ステートメントは、ステージ 1 から 3 までを使用してマイグレーションを行った場合のみ組み込まれます。 • このプログラムに対して、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> – 1 次作業用ストレージ・レコードの宣言をプログラムに組み込みます。 VAGen の 1 次作業用ストレージ・レコードにレベル 77 項目が存在する場合、ツールは新しいレベル 77 項目レコードの宣言も組み込みます。 – VAGen のテーブルおよび追加レコードのリストにあるすべてのレコードの宣言を組み込み、該当する場合は、VAGen の再定義レコードに対する redefines プロパティを組み込みます。 – 入出力レコードすべての宣言を組み込みます。 – MQ API 呼び出しのパラメーターとして使用されたレコード (VisualAge Generator 内で MQ メッセージ・レコードの属性として指定されたレコード) の宣言を組み込みます。 – 先頭 UI レコードとして使用される UI レコードの宣言を CONVERSE ステートメントまたは XFER ステートメント内に組み込みます。 – プログラムが I/O オプションで参照する DL/I セグメント・レコードの宣言が I/O オブジェクトへの階層パスにあるため、レコードに直接宣言を組み込みます。 – VAGen のテーブルおよび追加レコードのリストにあるすべてのテーブルの use 宣言を組み込みます。 – テーブルがプログラムのステートメントで明示的に参照される場合、メッセージ・テーブルに use 宣言ステートメントを組み込みます。 – マップ・ステートメントのある XFER で、プログラムが CONVERSE 表示、またはクローズ I/O オプションで参照するプログラムのマップ・グループ内の各マップに、呼び出し先パラメーターまたはプログラムの最初のマップとして use 宣言を組み込みます。 – プログラムのヘルプ・マップ・グループに use 宣言ステートメントを組み込みます。 – VAGen プログラムが PSB を指定している場合、プログラムに PSB の変数宣言を組み込みます。 	<p>マイグレーション・ツールは、使用可能なプログラム関連パーツをすべて使用します。</p>

表 33. 関連プログラム・パーツ (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: なし。</p>	<p>起こりうる問題 1: 欠落しているパーツがある場合に限り、問題が起こります。マイグレーション・ツールは、欠落しているパーツを検出すると、欠落しているパーツを示す警告メッセージを出します。マイグレーション・ツールは、欠落しているパーツに関する推測は行いません。このため、マイグレーションしたプログラムに、次の例のようなさまざまな問題が生じる可能性があります。</p> <ul style="list-style-type: none"> • import ステートメントの欠落。 • レベル 77 レコード宣言の欠落。 • VAGen 再定義レコードの redefines プロパティの欠落。 • 入出力レコード宣言の欠落。 • MQ API 呼び出しのパラメーターとして使用されたレコードの宣言の欠落。 • UI レコード宣言の欠落。 • SSA 内で参照されるセグメントの DL/I セグメント・レコード宣言の欠落。 • メッセージ・テーブルの use 宣言ステートメントの欠落 (テーブルがプログラムのステートメントで明示的に参照される場合)。 • マップ・グループ内のマップの use 宣言ステートメントの欠落。 <p>redefines プロパティが欠落している場合を除いて、「問題」ビューに表示されるエラーが、問題の識別に役立ちます。</p> <p>注: マイグレーション・ツールは、欠落しているパーツをすべて検出するとは限りません。</p> <p>考えられる解決策 1A: マイグレーション・セットを変更して、VisualAge Generator 内でプログラムの検証に必要なパーツをすべて組み込みます。プログラムの関連パーツがすべて一緒にマイグレーションされるように、新しいマイグレーション・セットを使用してプログラムのマイグレーションを再度行います。</p> <p>考えられる解決策 1B: EGL 内で欠落しているパーツを見つけて、EGL プログラムを修正します。</p> <p>起こりうる問題 2: レベル 77 項目が欠落している場合は、87 ページの『レコード内のレベル 77 項目』を参照してください。</p> <p>起こりうる問題 3: 再定義レコードが欠落している場合は、86 ページの『再定義レコード』を参照してください。</p>

呼び出しパラメーター・リストに EZEDLPCB が含まれるプログラム

VisualAge Generator: VisualAge Generator は、EZEDLPCB[n] を使って、プログラムが PCB をパラメーターとして受け取ることを示します。n は数値リテラルでなければなりません。n の値は 0 (I/O PCB の場合) であるか、プログラムの PSB パーツ内で定義されている PCB のいずれかに対応する数値でなければなりません。

EGL: EGL は xxxx_PCBRecord という型定義を持つ変数名を使って、プログラムが PCB をパラメーターとして受け取ることを示します。xxxx は PCB の型に応じて IO、ALT、DB、または GSAM になります。EGL にはまた、PCB 変数名をプログラムの PSB レコード内の対応する位置にマッピングするための **pcbParms** プロパティも必要です。

マイグレーションに必要な関連パーツ: PSB パーツ

表 34. 呼び出しパラメーター・リストに EZEDLPCB が含まれるプログラム

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>プログラムをマイグレーションする際に、そのプログラムがパラメーターとして EZEDLPCB[n] を指定し、かつ PSB パーツが使用可能であれば、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> パラメーターとして変数 PCB n を組み込み、次のように PSB 内の対応する PCB に基づいて型定義を指定します。 <ul style="list-style-type: none"> マイグレーション・ツールは、EZEDLPCB[0] の型定義を IO_PCBRecord に設定します。 n が PSB パーツ内の PCB に対応する場合、マイグレーション・ツールはその PSB 内の PCB 型を基に型定義を設定します。 n が PSB パーツ内の PCB 数より大きい場合、マイグレーション・ツールはメッセージを発行し、型定義を EZE_UNKNOWN_PCB_TYPE に設定します。 pcbParms プロパティ内の対応する場所に、すべての pcbn 変数をリストします。I/O PCB は pcb0 です。これは、パラメーターとして指定された場合、pcbParms プロパティの冒頭にリストされます。残りの pcbn 変数は、pcbParms リスト内の n+1 の位置にリストされます。 	<p>プログラムをマイグレーションする際に、そのプログラムがパラメーターとして EZEDLPCB[n] を指定し、かつ PSB パーツが使用不可であれば、マイグレーション・ツールはメッセージを発行し、次の処理を行います。</p> <ul style="list-style-type: none"> パラメーターとして変数 PCB n を組み込み、次のように PCB 番号に基づいて型定義を指定します。 <ul style="list-style-type: none"> マイグレーション・ツールは、EZEDLPCB[0] の型定義を IO_PCBRecord に設定します。 マイグレーション・ツールはメッセージを発行し、その他すべての pcbn 変数の型定義を EZE_UNKNOWN_PCB_TYPE に設定します。 メッセージを発行し、pcbParms プロパティを EZE_UNKNOWN_PCB_MAPPING に設定します。
<p>起こりうる問題: PSB パーツに EZEDLPCB[n] パラメーターの n の最高値より少ない数の PCB しかない場合にのみ、問題が発生します。</p> <p>解決策: このプログラムは、VisualAge Generator では無効です。プログラム・ロジックを検討して、プログラムのパラメーター・リストまたは PSB レコードのどちらを変更するのかを判断します。</p>	<p>起こりうる問題 1: 正しい PCB 型定義および pcbParms プロパティを指定する必要があります。</p> <p>解決策: プログラムの PSB レコードを見つけます。プログラムを編集し、PCB 型定義を修正します。また、pcbParms プロパティの pcbn 変数を正しくマッピングします。</p>

マイグレーションに必要な中間変数

VisualAge Generator: 一部の VAGen ステートメントは、EGL で同等なサポートを実現するために中間変数を必要とします。

EGL: EGL は、VAGen マイグレーションに必要な情報を提供するシステム・ライブラリー関数を備えています。このサポートは、VisualAge Generator 互換モードでのみ使用可能です。

マイグレーションに必要な関連パーツ: 適用不可。

表 35. マイグレーションに必要な中間変数

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>プログラムをマイグレーションする際に、マイグレーション・ツールは常に以下の宣言を組み込みます。</p> <ul style="list-style-type: none"> • <code>custPrefixEZEREPLY</code> • <code>custPrefixEZE_ITEMLEN</code> • <code>custPrefixEZE_WAIT_TIME</code> <p>マイグレーション設定の「古い EZESYS 値を初期化しない」が選択されていないと、マイグレーション・ツールは次の処理も行います。</p> <ul style="list-style-type: none"> • <code>custPrefixEZESYS</code> の宣言を組み込みます。 • <code>custPrefixEZESYS</code> の値を以前の <code>VAGen EZESYS</code> 値に設定する初期化ステートメントを組み込みます。 <p><code>custPrefix</code> は、マイグレーション・ツールの拡張予約語リストと競合するパーツ名の変更に使用されるものと同じ接頭部です。<code>custPrefix</code> に値を設定するには、<code>VAGen</code> マイグレーション設定を使用します。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>4 つの変数を使用して、以下のコードをマイグレーションします。</p> <ul style="list-style-type: none"> • (REPLY オプションが指定されていない場合は、<code>VAGen</code> サービス・ルーチン。この場合は、<code>handleSysLibraryErrors</code> の現行値を保管してから復元する必要があります。 • 直接相当するものが <code>EGL</code> にない <code>TEST nnn</code>、<code>+nnn</code>、または <code>-nnn</code> ステートメント。ユーザーが入力したデータの長さの判別には、<code>EGL</code> システム・ライブラリー関数が使用されます。 • <code>EZEWAIT</code> 関数。この場合、マイグレーション・ツールは時間を秒数に変換するためのロジックを追加します。 • <code>IF</code>、<code>WHILE</code>、および <code>TEST</code> を除く、以前の <code>VAGen</code> 値を必要とするステートメント内での <code>EZESYS</code> の参照。 	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

表 35. マイグレーションに必要な中間変数 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: マイグレーション中に、VAGen マイグレーション設定「以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)」を選択し、ステートメントに IF、WHILE、または TEST ではなく EZESYS を使用する場合に限り問題が起こります。この状態では、マイグレーション済みツールはステートメントで <code>custPrefixEZESYS</code> を使用しますが、プログラムには <code>custPrefixEZESYS</code> の宣言および初期化ステートメントがありません。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。</p> <p>考えられる解決策 1 EGL ロジックを変更して、<code>sysVar.systemType</code> の新規の値を使用します。</p> <p>考えられる解決策 2: <code>custPrefixEZESYS</code> の宣言および初期化ステートメントを、EZESYS の以前の VAGen 値の使用が必要なプログラムに追加します。</p>	<p>『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

関数 (入出力ステートメントを含む) の未確定状態の処理

マップの DISPLAY 入出力オプション

VisualAge Generator: DISPLAY は、表示マップおよびプリンター・マップの両方に使用されます。

EGL: 2 つの別個のステートメントを使用します。

- **display form** は、テキスト書式に使用されます。
- **print form** は、印刷書式に使用されます。

VisualAge Generator 互換モードでは、書式が印刷書式である場合に **display form** が受け入れられます。

マイグレーションに必要な関連パーツ: 装置タイプを判別するためにマップが必要です。マイグレーション・ツールは、使用可能ないずれかのマップ・グループにある、このマップ名をもつ最初のマップを使用します。プログラムのコンテキストでマイグレーションを行う場合、マイグレーション・ツールはメインのマップ・グループでのみプログラムを検索します。

表 36. マップの display 入出力オプション

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、この名前でのマップが使用可能であれば、マイグレーション・ツールは次の EGL コードに変換します。</p> <ul style="list-style-type: none"> • display textForm (マップが表示マップの場合) • print printForm (マップがプリンター・マップの場合) 	<p>この名前のマップが使用不可の場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • display form に変換します • マップ・タイプを判別できなかったことを示す警告メッセージを出します。

表 36. マップの *display* 入出力オプション (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題 1: 最初にマイグレーションされたプログラムが印刷書式を使用していたため、マイグレーション・ツールは print ステートメントにマイグレーションされました。別のプログラムが、同じ関数をテキスト書式に対して使用していると、問題が起こります。</p> <p>解決策 1: VisualAge Generator 互換モードを使用します。関数を編集して、print ステートメントを display ステートメントに変更します。</p> <p>起こりうる問題 2: VisualAge Generator 互換モードの使用を中止する場合に、テキスト書式を指定したプログラムと印刷書式を指定したプログラムがこの関数を使用していると、問題が発生します。</p> <p>考えられる解決策 2A: ある特定のターゲット環境が常に表示マップを使用し、その他の環境が常に印刷マップを使用する場合は、以下の例のように EGL 関数を変更することができます。</p> <pre>if (sysVar.systemType is zoscics) DISPLAY_FUNCTION(); else PRINT_FUNCTION(); end</pre> <p>ここで、DISPLAY_FUNCTION および PRINT_FUNCTION は、それぞれ display ステートメントおよび print ステートメントを使用します。</p> <p>考えられる解決策 2B: この関数が、以下に示すコードのように display ステートメントにマイグレーションされた場合を想定します。</p> <pre>before-logic display TextForm; after-logic</pre> <p>この場合は、以下に示すコードのように、関数を使用するようステートメントを変更することを検討します。</p> <pre>before-logic-function(); display TextForm; after-logic-function();</pre> <p>before-logic と after-logic を別個の関数に配置すると、共通関数のロジックの大部分を保持できます。こうすると、変更後の display 関数のコピーを作成して、print ステートメントを使用するように変更しても、引き続き共通の before-logic-function および after-logic-function を使用することができます。</p> <p>欠点: 元の DISPLAY 関数を使用する関数に影響を与える可能性があります。</p>	<p>起こりうる問題: 『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策を適用できます。</p>

入出力エラー・ルーチン

VisualAge Generator: ファイルまたはデータベースの入出力を行う関数は、入出力エラー・ルーチンを指定できます。入出力エラー・ルーチンは、main 関数または非 main 関数のどちらであってもよく、構文は同じです。VisualAge Generator は、テストまたは生成時に、入出力エラー・ルーチンがプログラムの main 関数または非 main 関数のどちらであるかを判別します。main 関数が入出力エラー・ルーチンとして使用されている場合、VisualAge Generator は関数スタックをスタックの先頭にポップし、スタックに (入出力エラー・ルーチンの) main 関数のみが存在する状態でスタックを再び開始してから、main 関数を呼び出します。非 main 関数が入出力エラー・ルーチンとして使用されている場合、VisualAge Generator は非 main 関数を現行関数スタックに追加してから、関数を呼び出します。

EGL: try ブロックおよび onException ステートメントがエラー処理に使用されます。onException ステートメントの構文は、以下のオプションをサポートします。

- **exit stack functionName;** を使用した、main 関数への再転送
- **nonmainfunctionName();** を使用した、非 main 関数の呼び出し。
- **mainfunctionName();** を使用した、main 関数の呼び出し。VisualAge Generator はこの書式をサポートしません。EGL は、main 関数を現行関数スタックに追加してから、main 関数を呼び出します。

マイグレーションに必要な関連パーツ: main 関数のリストを指定したプログラム。

表 37. ファイルおよびデータベースの入出力エラー・ルーチン

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、使用可能なプログラムがあれば、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none">• プログラムの main 関数を指定する入出力エラー・ルーチンを以下の書式に変更します。 <pre>try I/O-Statement; onException exit stack functionName; end</pre> <ul style="list-style-type: none">• 非 main 関数を指定する入出力エラー・ルーチンを以下の書式に変更します。 <pre>try I/O-Statement; onException functionName(); end</pre>	<p>使用可能なプログラムが存在しない場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none">• 入出力エラー・ルーチン内で指定されている関数が非 main 関数であると仮定して、その関数を以下の書式に変更します。 <pre>try I/O-Statement; onException functionName(); end</pre> <ul style="list-style-type: none">• 警告メッセージは出しません。警告メッセージは大量に発行される上に、重要視されない傾向があり、他の重大なエラー・メッセージが見つけにくくなる可能性があります。

表 37. ファイルおよびデータベースの入出力エラー・ルーチン (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: この関数が使用されるプログラム内で、入出力エラー・ルーチンが main 関数または非 main 関数のどちらとして使用されるかがオリジナル・プログラムと異なる場合に、問題が起こります。</p> <p>注: EGL 検証の結果、「問題」ビューにエラー・メッセージは表示されません。生成時にエラーは検出されません。ただし、このプログラムは、VisualAge Generator の場合と同じようには実行されません。VisualAge Generator ではスタックをポップしますが、EGL ではそうする代わりに main 関数をスタックに追加します。</p> <p>考えられる解決策: この状態が発生した場合は、main 関数への転送を行うための適切な構文を使用して、この入出力関数の新しいバージョンを作成します。</p> <p>欠点: この技法は、入出力関数を呼び出す他の関数に影響を与える可能性があります。</p>	<p>起こりうる問題: 入出力エラー・ルーチンが main 関数であるプログラム内でこの関数が使用される場合に、問題が起こります。</p> <p>注: EGL 検証の結果、「問題」ビューにエラー・メッセージは表示されません。生成時にエラーは検出されません。ただし、このプログラムは VisualAge Generator の場合と同じようには実行されません。VisualAge Generator ではスタックをポップしますが、EGL ではそうする代わりに main 関数をスタックに追加します。</p> <p>考えられる解決策: 『関連パーツを使用したマイグレーション』にリストしたものと同一解決策が適用されます。</p>

SQL 入出力ステートメント

VisualAge Generator: SQL 入出力の場合、レコード定義と Execution time statement build オプションの使用に基づいて、テストおよび生成機能が必要に応じて単一の入出力オプションを複数の SQL ステートメントに拡張します。テストおよび生成機能は、SQL レコード定義にある入出力ステートメントに対応する tables 文節を常に作成します。

EGL: SQL ステートメントは、EGL プログラム内で明示的に指定されている必要があります。SQL ステートメントを変更する場合は、INTO 文節を除くすべての SQL 文節が必要です。Execution time statement build は、**prepare** ステートメントとそれに続く **open**、**get**、または **execute** ステートメントで置き換えられます。

マイグレーションに必要な関連パーツ: SQL レコードと、代替仕様レコードとして指定されたレコード (存在する場合)。

表 38. SQL 入出力ステートメント

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、SQL レコードとその代替仕様レコードが使用可能であれば、マイグレーション・ツールはレコード定義、関数内の SQL ステートメント、および Execution time statement build オプションの使用に基づき、対応する EGL ステートメントを作成します。関数内の SQL ステートメントが変更された場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • INTO 文節を含むすべての文節を使用して、EGL SQL ステートメントを作成します。 • SQL レコード内、またはその代替仕様レコード (該当する場合) 内の表名から、必要な tables 文節を作成します。 • 入出力オブジェクトのレコード定義、または入出力オブジェクトの代替仕様レコードのレコード定義 (該当する場合) に基づいて、この SQL 入出力ステートメントに必要な、その他の欠落している文節を作成します。 • !itemColumnName を、項目名から対応する SQL 列名に変換します。 • 特殊な処理を必要とする SQL 予約語について、SQL ステートメントの検討を行いません。予約語のリスト、およびこれらの予約語のいずれかを表名または列名として使用している場合に SQL ステートメントに対して必要な変更については、300 ページの『特殊な処理を必要とする SQL 予約語』を参照してください。 <p>注:</p> <ul style="list-style-type: none"> • SQL 文節の欠落に関連した問題について詳しくは、119 ページの『SQL 入出力と必須 SQL 文節の欠落』を参照してください。 • !itemColumnName の使用に関連した問題について詳しくは、125 ページの『SQL 入出力と !itemColumnName』を参照してください。 	<p>SQL レコードまたはその代替仕様レコードが使用不可であれば、マイグレーション・ツールは SQL ステートメントの変更情報と Execution time statement build 情報のみを使用して、EGL SQL ステートメントを作成します。マイグレーション・ツールは使用可能なレコード定義を持たないため、以下の処理を行います。</p> <ul style="list-style-type: none"> • INTO 文節を含むすべての文節を使用して、EGL SQL ステートメントを作成します。 • 表名として EZE_UNKNOWN_SQLTABLE を使用し、tables 文節の表ラベルとして T1 を使用します。 • 欠落している SQL 文節に対して、EZE_UNKNOWN_SQL_clauseName を使用します。ここで、clauseName は欠落している SQL 文節の外部ソース形式のキーワードです (例: SELECT または VALUES)。 • 列名変数に対して !itemColumnName を使用します。 • 関数の検討が必要であることを示すエラー・メッセージを出します。 • 特殊な処理を必要とする SQL 予約語について、SQL ステートメントの検討を行いません。予約語のリスト、およびこれらの予約語のいずれかを表名または列名として使用している場合に SQL ステートメントに対して必要な変更については、300 ページの『特殊な処理を必要とする SQL 予約語』を参照してください。 <p>注:</p> <ul style="list-style-type: none"> • SQL 文節の欠落に関連した問題について詳しくは、119 ページの『SQL 入出力と必須 SQL 文節の欠落』を参照してください。 • !itemColumnName の使用に関連した問題について詳しくは、125 ページの『SQL 入出力と !itemColumnName』を参照してください。

表 38. SQL 入出力ステートメント (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題 1: 同名のレコードが 2 つあり、これらのレコードの SQL 表名または表ラベルが異なる場合に限り、問題が起こります。この問題は、複数の異なるサブシステムを使用する場合、またはテスト用と実動用に異なる表を使用して生成を行う場合に起こる可能性があります。</p> <p>考えられる解決策 1A: テストから実動までの間に表名の修飾を変更したことが問題の原因である場合は、非修飾の表名を使用するように変更し、バインド時に修飾情報を指定します。</p> <p>考えられる解決策 1B: 異なるサブシステム内で異なる表名を使用していることが問題の原因である場合は、レコードのコピーを作成し、コピーの名前を変更します。その後、入出力関数のコピーを作成して、新しいレコード名を使用します。新しい入出力関数の table 文節を適切に訂正します。</p> <p>欠点: この入出力関数を使用する関数に影響が波及する可能性があります。</p> <p>考えられる解決策 1C: 異なるサブシステムで異なる表名を使用していることが問題の原因である場合は、tableNameVariables プロパティを使用するようにレコードを変更し、さらに入出力関数を呼び出す前に表名変数を設定するように、このレコードの入出力を行うすべての関数を (可能であれば、各プログラムの EGL main 関数で) 変更します。あるいは、VisualAge Generator 内で表名ホスト変数を変更して、プログラム、レコード、および関数を再度マイグレーションします。</p> <p>欠点: この処置により、静的 SQL が動的 SQL に変更されるので、パフォーマンスへの影響が生じる可能性があります。</p> <p>起こりうる問題 2: SQL 表名または列名が、特殊な処理を必要とする SQL 予約語のいずれかである場合は、問題が起こります。マイグレーション・ツールは、これらの SQL 予約語を二重引用符で囲みません。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策 2A: 関数を編集して、SQL 表名または列名を二重引用符で囲みます。SQL 予約語のリストと、必要な構文の例については、300 ページの『特殊な処理を必要とする SQL 予約語』を参照してください。</p>	<p>起こりうる問題 1: 変更された SQL ステートメントまたは Execution time statement build を使用する SQL ステートメントがある場合は、問題が発生します。レコードが欠落しているかどうか、および SQL ステートメントから具体的にどの SQL 文節が欠落しているかに応じて、「問題」ビューにエラーが示される場合があります。</p> <p>解決策: 欠落している SQL 文節または表名に関連したメッセージがあるかどうか、マイグレーション・ログを検討してください。あるいは、EZE_UNKNOWN_SQL が発生しているかどうかワークスペースを検索します。レコード定義に基づいて、適切な tables 文節を判別します。EGL を使用して SQL 文節を再作成する方法については、119 ページの『SQL 入出力と必須 SQL 文節の欠落』を参照してください。!itemColumnName 変数の訂正については、125 ページの『SQL 入出力と !itemColumnName』を参照してください。</p> <p>起こりうるその他の問題: 『関連パーツを使用したマイグレーション』に示した問題と同じことが起こる可能性があり、同じ解決策が適用されます。</p>

SQL 入出力と必須 SQL 文節の欠落

VisualAge Generator: SQL 文節を変更した場合、VisualAge Generator バージョン 4.5 はすべての SQL 文節を保管します。一方、システム共通プロダクトと VisualAge Generator の一部の旧バージョンは、変更した文節のみを保管します。旧バージョンの関数を VisualAge Generator バージョン 4.5 内で変更しなかった場合、必要な SQL 文節の一部が欠落する可能性があります。さらに、VisualAge Generator バージョン 4.5 では、デフォルトの SQL は Execution time statement build オプションを指定して使用できます。生成時には、SQL PREPARE ステートメントの後に OPEN が続き、次に (INQUIRY または UPDATE の場合) FETCH が続きます。VisualAge Generator は、SQL PREPARE ステートメントに必要な情報を自動的に作成します。以下の表では、マイグレーション・ツールがさまざまな入出力オプションに対して作成できる文節 (tables 文節を含む) を示します。

表 39. マイグレーション・ツールが作成できる SQL 文節

入出力 オプション	Execution time statement build を 指定した/指定しない SQL 文節の 欠落	Execution time statement build を 指定したデフォルトの SQL
ADD	INSERTCOLNAME、VALUES	適用不可
CLOSE	適用不可	適用不可
DELETE	適用不可	適用不可
INQUIRY	SELECT、INTO	SELECT、INTO、WHERE
REPLACE	適用不可	適用不可
SCAN	適用不可	適用不可
SETINQ	SELECT、INTO	SELECT、INTO、WHERE、 ORDERBY
SETUPD	SELECT、INTO、FORUPDATEOF	SELECT、INTO、WHERE、 FORUPDATEOF
SQLEXEC	適用不可	レコードおよび model=UPDATE を使用する場合 UPDATE、SET、WHERE レコードおよび model=DELETE を使用する場合 DELETE、WHERE
UPDATE	SELECT、INTO、FORUPDATEOF	SELECT、INTO、WHERE、 FORUPDATEOF

EGL: いずれかの SQL 文節を変更する場合は、SQL ステートメントの SQL 文節をすべて指定する必要があります。さらに EGL では、**prepare** ステートメントは、SQL PREPARE が必要とする SQL ステートメント全体を指定する必要があります。

マイグレーションに必要な関連パーツ: SQL レコードと、代替仕様レコードとして指定されたレコード (存在する場合)。

表 40. SQL 入出力と SQL 文節の欠落

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
SQL レコードとその代替仕様レコードが使用可能であり、いずれかの SQL 文節が存在し、ただし一部の文節が欠落している場合、マイグレーション・ツールはこの表の以降の行で説明するとおりに欠落している文節を作成します。さらにデフォルトの SQL が Execution time statement build オプションを指定して使用される場合、マイグレーション・ツールはその入出力オプションのすべての関連文節を作成します。この関数の最初のマイグレーションを基準として、マイグレーション・ツールは SQL レコードとその代替仕様レコード (存在する場合) を使用して、欠落している文節を作成します。	SQL レコードまたはその代替仕様レコードが使用できず、いずれかの SQL 文節が存在し、ただし一部の文節が欠落している場合、マイグレーション・ツールはこの表の以降の行で説明するとおりに欠落している文節を作成します。さらにデフォルトの SQL が Execution time statement build オプションを指定して使用される場合、マイグレーション・ツールはその入出力オプションのすべての関連文節を作成します。この関数の最初のマイグレーションを基準として、SQL レコードまたはその代替仕様レコードが使用できなければ、マイグレーション・ツールは意図的に無効な EGL 構文を作成します。
tables 文節の欠落: マイグレーション・ツールは、レコードにある SQL 表と表ラベルすべてをリストして、tables 文節を作成します。マイグレーション・ツールは、SQL 表名と表名ホスト変数の両方を、VAGen レコード定義内での出現順と同じ順序で組み込みます。	tables 文節の欠落: マイグレーション・ツールは、SQL 表名を EZE_UNKNOWN_SQLTABLE に設定して、表ラベルを T1 に設定し、エラー・メッセージを出します。
SELECT 文節の欠落: マイグレーション・ツールは、レコード内での項目の出現順と同じ順序でレコードにある SQL 列名をすべてリストして、SELECT 文節を作成します。	SELECT 文節の欠落: マイグレーション・ツールは、SELECT 文節の SQL 列名を EZE_UNKNOWN_SQL_SELECT に設定して、エラー・メッセージを出します。
INTO 文節の欠落: マイグレーション・ツールは、レコード内での項目の出現順と同じ順序でレコードにある項目名をすべてリストして、INTO 文節を作成します。	INTO 文節の欠落: マイグレーション・ツールは、INTO 文節の項目名を EZE_UNKNOWN_SQL_INT0 に設定して、エラー・メッセージを出します。
INSERTCOLNAME 文節の欠落: マイグレーション・ツールは、レコード内での項目の出現順と同じ順序でレコードにある SQL 列名をリストして、VAGen ADD 関数の挿入される列名のリストを作成します。マイグレーション・ツールは、読み取り専用として指定されている項目の SQL 列名を省略します。	INSERTCOLNAME 文節の欠落: マイグレーション・ツールは、リストの SQL 列名を EZE_UNKNOWN_SQL_INSERTCOLNAME に設定して、エラー・メッセージを出します。
VALUES 文節の欠落: マイグレーション・ツールは、レコード内での項目の出現順と同じ順序でレコードにある項目名をリストして、VAGen ADD 関数の VALUES 文節を作成します。マイグレーション・ツールは、読み取り専用として指定されている項目の項目名を省略します。	VALUES 文節の欠落: マイグレーション・ツールは、VALUES 文節の項目名を EZE_UNKNOWN_SQL_VALUES に設定して、エラー・メッセージを出します。
FOR UPDATE OF 文節の欠落: マイグレーション・ツールは、レコード内での項目の出現順と同じ順序でレコードにある SQL 列名をリストして、FOR UPDATE OF 文節を作成します。マイグレーション・ツールは、EGL keyItems プロパティに含まれている項目、または読み取り専用として指定されている項目の SQL 列名を省略します。	FORUPDATEOF 文節の欠落: マイグレーション・ツールは、FOR UPDATE OF 文節の SQL 列名を EZE_UNKNOWN_SQL_FORUPDATEOF に設定して、エラー・メッセージを出します。

表 40. SQL 入出力と SQL 文節の欠落 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>SET 文節の欠落: 変更された SQL の場合は、tables 文節以外にただ 1 つの文節 (SET 文節) のみが存在します。REPLACE 入出力オプションの SET 文節が欠落している場合、この入出力オプションはデフォルトの SQL を使用します。この場合、マイグレーション・ツールは SET 文節を作成しません。</p> <p>Execution time statement build がデフォルトの SQL を使用して指定されている場合、あるレコードが指定されて、モデル・オプションが UPDATE に設定されると、マイグレーション・ツールは SQL 列名をリストして各列をレコードの対応する項目の値に設定することにより、SET 文節を作成します。マイグレーション・ツールは、レコード内での項目の出現順と同じ順序で SQL 列名と項目名のペアをリストします。マイグレーション・ツールは、EGL keyItems プロパティに含まれている項目、または読み取り専用として指定されている項目の SQL 列名と項目名のペアを省略します。</p>	<p>SET 文節の欠落: 変更された SQL の場合、マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p> <p>Execution time statement build がデフォルトの SQL を使用して指定されている場合、あるレコードが指定されて、モデル・オプションが UPDATE に設定されると、マイグレーション・ツールは SET 文節を EZE_UNKNOWN_SQL_SET に設定してエラー・メッセージを出します。</p>

表 40. SQL 入出力と SQL 文節の欠落 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>WHERE 文節の欠落: 変更された SQL の場合、WHERE 文節は必要ありません。この場合、マイグレーション・ツールは WHERE 文節を作成しません。</p> <p>Execution time statement build がデフォルトの SQL を使用して指定されている場合、マイグレーション・ツールは入出力オプションに基づいて、以下の方法で WHERE 文節を作成します。</p> <ul style="list-style-type: none"> SETINQ または SETUPD の場合、マイグレーション・ツールは次の方法で WHERE 文節を作成します。 <ul style="list-style-type: none"> デフォルトの選択状態があり、キー項目がないか複数のキー項目がある場合、マイグレーション・ツールはそのデフォルトの選択状態から WHERE 文節を作成します。 デフォルトの選択状態はないが、1 つのキー項目がある場合、マイグレーション・ツールはそのキー項目を使用して WHERE 文節を作成します。 デフォルトの選択状態と 1 つのキー項目がある場合、マイグレーション・ツールはそのデフォルトの選択状態とキー項目の両方を使用して WHERE 文節を作成します。 それ以外の場合はすべて、マイグレーション・ツールは WHERE 文節を省略します。 モデル・オプションが UPDATE または DELETE に設定されている INQUIRY、UPDATE、または SQLEXEC の場合、マイグレーション・ツールは次の方法で WHERE 文節を作成します。 <ul style="list-style-type: none"> デフォルトの選択状態があり、キー項目がない場合、マイグレーション・ツールはそのデフォルトの選択状態から WHERE 文節を作成します。 デフォルトの選択状態がなく、1 つ以上のキー項目がある場合、マイグレーション・ツールはすべてのキー項目を使用して WHERE 文節を作成します。 デフォルトの選択状態があり、1 つ以上のキー項目がある場合、マイグレーション・ツールはそのデフォルトの選択状態およびそれらすべてのキー項目を使用して WHERE 文節を作成します。 それ以外の場合はすべて、マイグレーション・ツールは WHERE 文節を省略します。 	<p>WHERE 文節の欠落: 変更された SQL の場合、マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p> <p>Execution time statement build がデフォルトの SQL を使用して指定され、入出力オプションが SETINQ、SETUPD、INQUIRY、UPDATE、または SQLEXEC で、モデル・オプションが UPDATE または DELETE に設定されている場合、マイグレーション・ツールは WHERE 文節を EZE_UNKNOWN_SQL_WHERE に設定してエラー・メッセージを出します。</p>

表 40. SQL 入出力と SQL 文節の欠落 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>ORDER BY 文節の欠落: 変更された SQL の場合、ORDER BY 文節は必要ありません。この場合、マイグレーション・ツールは ORDER BY 文節を作成しません。</p> <p>Execution time statement build がデフォルトの SQL を使用して指定されている場合、マイグレーション・ツールは入出力オプションに基づいて、以下の方法で ORDER BY 文節を作成します。</p> <ul style="list-style-type: none"> • SETINQ の場合、マイグレーション・ツールは EGL keyItems プロパティの各項目に対応するフィールド位置をリストして ASC オプションを組み込むことにより ORDER BY 文節を作成します。レコードの最初のフィールドは、位置 1 と呼ばれます。 • INQUIRY、UPDATE、SETUPD、または SQLEXEC の場合、マイグレーション・ツールは ORDER BY 文節を作成しません。 	<p>ORDER BY 文節の欠落: 変更された SQL の場合、マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p> <p>Execution time statement build がデフォルトの SQL を使用して指定されている場合、入出力オプションが SETINQ に設定されていると、マイグレーション・ツールは SET 文節を EZE_UNKNOWN_SQL_ORDERBY に設定してエラー・メッセージを出します。</p>
<p>起こりうる問題: 同名のレコードが 2 つあり (通常は、別のサブシステム内に)、これらのレコードの項目名または SQL 列名が異なる場合に限り、問題が起こります。</p> <p>考えられる解決策: 別のサブシステム内で使用するための関数のコピーを作成し、正しい項目名と SQL 列名を使用するように新規関数を変更します。</p> <p>欠点: この入出力関数を使用する関数に影響が波及する可能性があります。</p>	<p>起こりうる問題 1: 変更された SQL ステートメントまたは Execution time statement build を使用する SQL ステートメントがある場合は、問題が発生します。</p> <p>解決策 1A: SQL 文節の欠落に関連したメッセージがあるかどうか、エラー・メッセージのリストを検討してください。SQL 入出力関数を変更して、欠落している文節を組み込みます。欠落している文節を作成するために必要な情報は、『関連パーツを使用したマイグレーション』欄の対応する行にあります。</p> <p>解決策 1B: VisualAge Generator 内で関数を編集し、SQL エディターを使用して、行末へのブランクの追加など細かい変更を行います。SQL 文節を保管し、関数を再度マイグレーションします。マイグレーション・ツールが EGL 入出力ステートメントに SQL 表の情報を含めることができるように、レコード定義を必ず組み込んでください。</p> <p>起こりうる問題 2: 『関連パーツを使用したマイグレーション』に示した問題と同じことが起こる可能性があり、同じ解決策が適用されます。</p>

SQL 入出力と Execution time statement build

VisualAge Generator: SQL INQUIRY、UPDATE、SETINQ、または SETUPD 関数で Execution time statement build オプションが指定されている場合、VisualAge Generator は SQL PREPARE ステートメントに続いて OPEN、次に (INQUIRY または UPDATE の場合) FETCH を作成します。SQL PREPARE ステートメントでソフト・エラーが発生した場合、SQL OPEN または GET を使用して処理が継続されます。さらに SQLEXEC 関数で Execution time statement build オプションが指定されている場合、VisualAge Generator は SQL EXECUTE IMMEDIATE ステートメントを作成します。

EGL: EGL **prepare** ステートメントは、標準の入出力プロセスに従います。ソフト・エラーが発生する場合、制御は **onException** ブロック (あれば) に移ります。これは、SQL PREPARE ステートメントでソフト・エラーが発生する場合、処理が対応する SQL OPEN、GET、または EXECUTE ステートメントを使用して自動的に続行されないことを意味します。さらに、EGL は SQL EXECUTE IMMEDIATE ステートメントをサポートしません。EGL **prepare** ステートメントの後に **execute** ステートメントを続けることが、このステートメントに最も近い処理になります。

マイグレーションに必要な関連パーツ: 適用不可。

表 41. SQL 入出力と *!itemColumnName*

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>Execution time statement build オプションおよび入出力エラー・ルーチンを使用する SQL INQUIRY、UPDATE、SETINQ、または SETUPD 関数の場合、マイグレーション・ツールは入出力オプションを以下のコードに変換して、EGL prepare ステートメントのソフト・エラーに対処します。</p> <pre>try prepareStatement; end if (recordName not HardIOError) try openOrGetStatement; onException IOErrorRoutineEquivalent; end else IOErrorRoutineEquivalent; end</pre> <p>関数に入出力エラー・ルーチンが含まれていない場合、マイグレーション・ツールは入出力オプションを以下のコードに変換して、警告メッセージを出します。</p> <pre>prepareStatement; openOrGetStatement;</pre>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>Execution time statement build オプションを使用する SQLEXEC 関数の場合、マイグレーション・ツールは入出力オプションを EGL prepare ステートメントとそれに続く EGL execute ステートメントに変換します。マイグレーション・ツールは警告メッセージを出します。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

表 41. SQL 入出力と !itemColumnName (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題 1: INQUIRY、UPDATE、SETINQ、または SETUPD 関数で Execution time statement build オプションを指定しているが入出力エラー・ルーチンを指定していない場合は、問題が発生します。マイグレーション・ツールは警告メッセージを出します。</p> <p>考えられる解決策 1: EGL prepare ステートメントでのハード・エラー後に処理を継続する必要がある場合は、関数を編集して前に示したロジックを追加します。</p> <p>起こりうる問題 2: SQLEXEC 関数で Execution time statement build オプションを指定している場合、問題が発生する可能性があります。これは、EGL における prepare ステートメントとそれに続く execute ステートメントのパフォーマンスが、VisualAge Generator における SQL EXECUTE IMMEDIATE ステートメントのパフォーマンスと同じでない可能性があるためです。マイグレーション・ツールは警告メッセージを出します。</p> <p>考えられる解決策 2: なし。プログラムを生成してテストし、パフォーマンスが許容可能であることを確認してください。</p>	<p>起こりうる問題: 『関連パーツを使用したマイグレーション』欄にリストした問題と同じ問題が発生する可能性があります、同じ解決策を適用できます。</p>

SQL 入出力と !itemColumnName

VisualAge Generator: SQL 入出力の場合、VisualAge Generator は SQL ステートメントの一部の文節で !itemColumnName の使用を許容します。テストおよび生成時に、SQL 行レコード内の項目名に対応する SQL 列名が決定されます。

EGL: !itemColumnName の使用はサポートされません。

マイグレーションに必要な関連パーツ: SQL レコードと、代替仕様レコードとして指定されたレコード (存在する場合)。

表 42. SQL 入出力と !itemColumnName

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、SQL レコードとその代替仕様レコードが使用可能ならば、マイグレーション・ツールは SQL レコードまたはその代替仕様レコード (存在する場合) に基づいて、!itemColumnNames を対応する SQL 列名に変換します。</p>	<p>SQL レコードまたはその代替仕様レコードが使用不可の場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • 列名変数に対して !itemColumnNames を使用します。 • 関数の検討が必要であることを示すエラー・メッセージを出します。

表 42. SQL 入出力と !itemColumnName (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: 同名のレコードが 2 つあり (通常は、別のサブシステム内に)、これらのレコードの !itemColumnName に対応する SQL 列名が異なる場合に限り、問題が起こります。</p> <p>考えられる解決策: 別のサブシステム内で使用するための関数のコピーを作成し、正しい SQL 列名を使用するように新規関数を変更します。</p> <p>欠点: この入出力関数を使用する関数に影響が波及する可能性があります。</p>	<p>起こりうる問題 1: 変更された SQL ステートメント、または Execution time statement build オプションを使用する SQL ステートメントがある場合は、問題が発生します。</p> <p>解決策: !itemColumnNames に関連したメッセージがあるかどうか、エラー・メッセージのリストを検討してください。SQL 入出力関数を変更して、SQL 行レコードに基づいた正しい列名を組み込みます。</p> <p>起こりうる問題 2: 『関連パーツを使用したマイグレーション』に示した問題と同じことが起こる可能性があり、同じ解決策が適用されます。</p>

複数の UPDATE 関数または SETUPD 関数を使用する SQL 入出力

VisualAge Generator: SQL 入出力の場合、プログラム内に複数の UPDATE 関数または SETUPD 関数があれば、それぞれの SQL REPLACE 関数に、対応する UPDATE 関数または SETUPD 関数の名前を指定する必要があります。これは、非 SQL 入出力の場合は必要ありません。非 SQL 入出力の場合は、SETUPD はサポートされません。

EGL: SQL 入出力に対して、複数の **get forUpdate** または **open forUpdate** ステートメントがある場合、それぞれの SQL **replace** ステートメントで、それに対応する **get** または **open** ステートメントの名前を指定する必要があります。 **get** および **open** ステートメントは、それぞれ resultSetID を指定します。 **replace** ステートメントは、対応する **get** または **open** ステートメントの resultSetID を指定します。 resultSetID は、非 SQL 入出力の場合は適用されません。

マイグレーションに必要な関連パーツ: 入出力オブジェクトであるレコード。

表 43. 複数の UPDATE 関数または SETUPD 関数を使用する SQL 入出力

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、レコードが使用可能ならば、マイグレーション・ツールはレコード・タイプに基づいて対応する EGL ステートメントを作成します。</p> <p>SQL の場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> UPDATE 関数または SETUPD 関数をマイグレーションする場合は、常に resultSetID を組み込みます。resultSetID は、関数名とお客様指定の接尾部を使用して作成されます。 対応する UPDATE または SETUPD の関数名を指定していた REPLACE 関数をマイグレーションする場合は、resultSetID を組み込みます。resultSetID は、対応する UPDATE または SETUPD の関数名と、お客様指定の接尾部を使用して作成されます。 <p>非 SQL の場合、マイグレーション・ツールは、UPDATE 関数または REPLACE 関数をマイグレーションする際に resultSetID を常に省略します。非 SQL 入出力の場合は、SETUPD 関数は存在しません。</p>	<p>UPDATE 関数をマイグレーションする際にレコードが使用不可の場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> この関数が SQL 入出力用であるかどうかの判別を試みます。そのため、関数が SQL 文節を持っているかどうか、または Execution time statement build、single row select、Declare cursor with hold などの SQL 固有の情報を持っているかどうかを検査します。 マイグレーション・ツールは、この UPDATE 関数が SQL レコード用であることが判別できる場合、get ステートメントに resultSetID を組み込みます。 そうでなければ、マイグレーション・ツールは resultSetID を組み込みません。マイグレーション・ツールは警告メッセージを出します。 <p>SETUPD 関数をマイグレーションする場合、SETUPD は SQL の場合のみ有効なので、マイグレーション・ツールは resultSetID を常に組み込みます。</p> <p>REPLACE 関数をマイグレーションする場合は、関数が対応する UPDATE または SETUPD の関数名を指定していれば、マイグレーション・ツールは resultSetID を組み込みます。</p>
<p>起こりうる問題: なし。</p>	<p>起こりうる問題: 未変更の UPDATE 関数が実際に SQL レコードを参照しており、複数の get または open forUpdate ステートメントが存在するプログラム内で使用されている場合にのみ、問題が発生します。この場合、それぞれの replace ステートメントに resultSetID が組み込まれますが、VAGen UPDATE 関数に対してマイグレーションされた get ステートメントには resultSetID が組み込まれません。プログラムの生成は失敗します。</p> <p>解決策: 関数を変更して、get ステートメントに resultSetID を組み込みます。</p>

DL/I I/O および比較値項目

VisualAge Generator: DL/I I/O で比較値項目が修飾されていない場合、VisualAge Generator は、現在のセグメント検索索引数 (SSA) に指定されるセグメントに対応するレコードを優先します。そのレコードに比較値項目が見つからない場合、ネーム解決ルールは時間の経過とともに変化しています。一般的に、VisualAge Generator は次に作業用ストレージ・レコード内を、続いて入出力オブジェクトなど他の DL/I セグメント・レコード内を検索します。関数のローカル・ストレージまたはパラメーター・リストにある項目は無視されます。関数のローカル・ストレージまたはパラメーター・リストにあるレコードが対象になりますが、プログラム内で一意的な名前を持つ項目のみが考慮されます。

EGL: DL/I I/O の場合、比較値項目が修飾されていないと、EGL は標準の EGL 修飾ルールに従います。EGL は最初に関数のローカル・ストレージ内またはパラメーター・リスト内の項目を検索し、次に関数のローカル・ストレージ内、パラメーター・リスト内、入出力オブジェクト内のレコードのフィールドを検索し、最後にプログラム内のすべての変数を検索します。

マイグレーションに必要な関連パーツ: DL/I セグメント・レコードと、代替仕様レコードとして指定されたレコード (存在する場合)。

表 44. DL/I I/O および比較値項目

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、比較値項目が修飾されていない場合には、マイグレーション・ツールは現在の SSA に関連する DL/I セグメント・レコードを検索します。DL/I セグメント・レコードとその代替仕様レコードが使用可能な場合、マイグレーション・ツールは次の方法で比較値項目のレコードを検査します。</p> <ul style="list-style-type: none"> 比較値項目が DL/I セグメント・レコード内にある場合、マイグレーション・ツールはその DL/I セグメント・レコード名を使って比較値項目を修飾します。 項目がレコード内にない場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> レコード名として EZE_UNKNOWN_QUALIFIER を使います。 マイグレーション・ツールが比較値項目の修飾を確定できないというメッセージを発行します。 	<p>DL/I セグメント・レコードまたはその代替仕様レコードが使用不可の場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> レコード名として EZE_UNKNOWN_QUALIFIER を使います。 マイグレーション・ツールが比較値項目の修飾を確定できないというメッセージを発行します。
<p>ありうる問題: 比較値項目が修飾されていなくて、かつ関連する DL/I セグメント・レコードまたはその代替仕様レコード内にない場合にのみ、問題が発生します。</p> <p>考えられる解決策: プログラム・ロジックを検討して、使用すべき適切な修飾を決定します。さらに、プログラムを生成した最後の時刻から、生成済み COBOL のソース・コードを検討できます。VisualAge Generator では、ある時点で比較値項目の修飾ルールが変化しています。このため、最後のプログラム生成以降に VisualAge Generator の現行リリースが変更されていないことが確信できない場合、現行のリリースを使用してプログラムを再生成しないでください。</p>	<p>起こりうる問題 1: 修飾されていない比較値項目がある場合、問題が起こります。</p> <p>解決策: DL/I I/O 関数を変更して、比較値項目を正しく修飾するようにします。まず、修飾ステートメントに関連する DL/I セグメント・レコードを必ずチェックしてください。</p> <p>起こりうる問題 2: 『関連パーツを使用したマイグレーション』に示した問題と同じことが起こる可能性があり、同じ解決策が適用されます。</p>

その他のステートメントの未確定状態の処理

ステートメントの暗黙データ項目

VisualAge Generator: VisualAge Generator では、暗黙データ項目 (レコード、マップ、テーブル、呼び出し先パラメーター・リスト、関数仮パラメーター・リスト、または関数ローカル・ストレージ内で明示的に定義されていない項目) を使用できます。ただし、暗黙データ項目の使用は、一般にあまり好ましくありません。

EGL: EGL は暗黙項目を許容しません。

マイグレーションに必要な関連パーツ: 適用不可。

表 45. ステートメントの暗黙データ項目

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
107 ページの『プログラム内の暗黙データ項目』を参照	107 ページの『プログラム内の暗黙データ項目』を参照

ステートメントのレベル 77 項目

VisualAge Generator: 作業用ストレージ・レコードのみがレベル 77 項目を含むことができます。プログラムは、1 次作業用ストレージ・レコード内のレベル 77 項目のみを参照できます。

EGL: レベル 77 項目は許容されません。

マイグレーションに必要な関連パーツ: 関数をマイグレーションする場合には、1 次作業用ストレージ・レコードが必要です。

表 46. ステートメントのレベル 77 項目

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
87 ページの『レコード内のレベル 77 項目』を参照	87 ページの『レコード内のレベル 77 項目』を参照

ステートメントのテーブル参照

VisualAge Generator: 関数がテーブルを参照している場合、そのテーブルはその関数の関連パーツであるとは見なされません。

EGL: 関数が DataTable を参照している場合、この関数を含むファイルには、DataTable を含むパッケージに対する **import** ステートメントを組み込む必要があります。

マイグレーションに必要な関連パーツ: テーブル。

表 47. ステートメントのテーブル参照

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
テーブルが使用可能な場合、マイグレーション・ツールはステージ 2 マイグレーション中、関数の関連付けとしてテーブルを追加します。次に、ステージ 3 マイグレーションでは、対応する import ステートメントを、関数を含むファイルに追加します。	テーブルが使用可能でない場合、マイグレーション・ツールは関数の関連付けとしてテーブルを追加しません。ステージ 3 の間に import ステートメントが追加されません。
起こりうる問題: なし。	<p>起こりうる問題: テーブルと同じパッケージにパーツをインポートする必要があるため、関数、または関数と同じファイル内の他のパーツに対する import ステートメントが存在しない場合は、問題が発生します。</p> <p>解決策: 関数を含むファイルに import ステートメントを追加します。</p>

単一行テーブルをソースとして含む MOVEA

VisualAge Generator: 内容が単一行であるテーブルを MOVEA ステートメントのソースとして使用すると、そのソースはスカラーとして扱われ、ターゲット配列はこのスカラー・ソースによって完全に初期化されます。これは VisualAge Generator の資料と矛盾しています。この資料では、テーブルは常に配列として扱う必要があり、これによりターゲット配列の最初の要素のみが初期化される、と説明されています。

EGL: for 修飾子付きの **move** は、常にある配列から別の配列への移動用と見なされます。内容が単一行であるテーブルが、**for** 修飾子付きの **move** のソースとして使用されている場合は、ターゲット配列の最初の要素のみが初期化されます。

マイグレーションに必要な関連パーツ: テーブル。

表 48. 単一行テーブルをソースとして含む MOVEA

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
テーブルをマイグレーションする場合、そのテーブルの内容に単一行しか含まれていない場合、マイグレーション・ツールは警告メッセージを発行します。	テーブルをマイグレーションするときに、マイグレーション・ツールは『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。
ステートメントをマイグレーションする場合、MOVEA ステートメントのソースが修飾されており、その修飾子を使用可能になっていれば、マイグレーション・ツールは、その修飾子が単一行のみの内容を含むテーブルであるかどうかを判別します。そうである場合、マイグレーション・ツールはエラー・メッセージを発行します。	ステートメントをマイグレーションする場合、MOVEA ステートメントのソースが修飾されており、その修飾子名の文字数が 7 文字以下であり、その修飾子を使用不可であれば、マイグレーション・ツールは警告メッセージを発行します。
MOVEA のソースが修飾されていないか、修飾子がテーブルではない場合は、マイグレーション・ツールはメッセージを発行しません。	
起こりうる問題 1: 内容が単一行であるテーブルには問題が発生します。マイグレーション・ツールはエラー・メッセージを出します。	起こりうる問題: 『関連パーツを使用したマイグレーション』欄にリストした問題と同じ問題が発生する可能性があります。同じ解決策を適用できます。
考えられる解決策 1: ターゲット配列全体を初期化する必要がある場合は、プログラム・ロジックを変更し、ループを使用して単一行のソース・テーブルからターゲット配列の要素を初期化します。	
起こりうる問題 2: ソースが修飾されておらず、このソースが、単一行の内容しか持たないテーブル内の 1 つのフィールドに解決する場合も問題が発生します。この場合、プログラムは VisualAge Generator の場合と同じようには実行しません。	
考えられる解決策 2: 解決策は起こりうる問題 1 の場合と同じです。	

代入ステートメント

VisualAge Generator: 代入ステートメントは、レコードとマップに対して使用でき、「move corresponding」を行います。MOVE ステートメントは、項目に対して使用できます。

EGL: 代入ステートメントは、データ項目に対して使用するか、レコードのバイト単位での移動にしか使用できません。代入ステートメントを書式に対して使用することはできません。レコードと書式の「move corresponding」には、**move byName** ステートメントが必要です。**withV60Compat** 修飾子を指定して **move** ステートメントを使用すると、ソースまたはターゲットがレコードまたは書式である場合には **move corresponding** を実行することができ、ソースまたはターゲットがフィールドである場合には代入ステートメントを実行することができます。

マイグレーションに必要な関連パーツ: 適用不可。

表 49. 代入ステートメント

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>可能なかぎり多くの共通コードを保持するために、代入ステートメントまたは move ステートメントのソースとターゲットがいずれも修飾されておらず、添え字なしの名前である場合に、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none">関数パラメーター・リスト、ローカル・ストレージ、および入出力オブジェクトを検査して、代入ステートメントまたは MOVE ステートメントのソースまたはターゲットが、項目、レコード、またはマップのどれであるか判別を試みます。マイグレーション・ツールが判別できる場合には、以下のようにしてマイグレーションが行われます。<ul style="list-style-type: none">ソースまたはターゲットが項目ならば、代入ステートメントへ。ソースまたはターゲットがレコードまたはマップの場合は、move byName ステートメントへ。マイグレーション・ツールは、パーツ型を判別できない場合、代入ステートメントと MOVE ステートメントを withV60Compat 修飾子が指定された move ステートメントにマイグレーションします。	<p>この処理は、『関連パーツを使用したマイグレーション』欄の説明と同じように行われます。</p>
<p>起こりうる問題: なし。EGL のデバッグおよび生成によって、withV60Compat 修飾子を持つ move ステートメントが VAGen MOVE ステートメントに変換されます。移動の実際のソースとターゲットに応じて、項目間の move または move byName (move corresponding) が実行されます。どのプログラムも、関数を変更せずに使用できます。</p>	<p>起こりうる問題: なし。『関連パーツを使用したマイグレーション』欄の説明と同じ状況が当てはまります。</p>

FIND ステートメント

VisualAge Generator: FIND ステートメントの検索列はオプションです。デフォルトは、VAGen テーブルの先頭列です。

EGL: FIND ステートメントは、if ステートメントで置き換えられます。検索列は必須です。

マイグレーションに必要な関連パーツ: VAGen テーブル。

表 50. FIND ステートメント

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
この関数の最初のマイグレーションを基準として、検索列が明示的に指定されておらず、テーブルが使用可能ならば、マイグレーション・ツールはテーブルを展開して、テーブルの先頭列から検索列の名前を取得します。	検索列が明示的に指定されておらず、テーブルが使用できない場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none">検索列を次のように設定します。 <code>EZE_UNKNOWN_SEARCH_COLUMN</code>正しい列名を使用するように関数を変更する必要があることを示すエラー・メッセージを出します。
<p>起こりうる問題: 問題が発生するのは、おそらく異なるサブシステム内にある 2 つの DataTable の DataTable 名が同じで、検索列名が異なる場合のみです。</p> <p>解決策: 第 2 のサブシステムに対して、DataTable の最初の列のサブストラクチャーとしてフィールドを追加します。この新規フィールドの名前は、第 1 のサブシステムの検索列と同じであることが必要です。この技法により、第 2 のサブシステム内でコードを変更せずに、共通関数を共用できます。</p>	<p>起こりうる問題 1: 検索列名を指定する必要があります。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策: 関数を編集し、DataTable に正しい列名を指定します。</p> <p>起こりうる問題 2: 『関連パーツを使用したマイグレーション』に示した問題と同じことが起こる可能性があり、同じ解決策が適用されます。</p>

RETR ステートメント

VisualAge Generator: RETR ステートメントの検索列と戻す列はオプションです。検索列のデフォルトは、VAGen テーブルの先頭列です。戻す列のデフォルトは 2 列目です。

EGL: RETR ステートメントは、if ステートメントで置き換えられます。検索列と戻す列は必須です。

マイグレーションに必要な関連パーツ: VAGen テーブル。

表 51. RETR ステートメント

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
この関数の最初のマイグレーションを基準として、検索列または戻す列が明示的に指定されておらず、テーブルが使用可能であれば、マイグレーション・ツールはテーブルを展開して、以下の情報を取得します。 <ul style="list-style-type: none">検索列の名前をテーブルの先頭列から。戻す列の名前をテーブルの 2 列目から。	検索列または戻す列が明示的に指定されておらず、テーブルが使用できない場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none">検索列を次のように設定します。 <code>EZE_UNKNOWN_SEARCH_COLUMN</code>戻す列を次のように設定します。 <code>EZE_UNKNOWN_RETURN_COLUMN</code>正しい列名を使用するように関数を変更する必要があることを示すエラー・メッセージを出します。

表 51. *RETR* ステートメント (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: 問題が発生するのは、おそらくは異なるサブシステムにある 2 つの <i>DataTable</i> の <i>DataTable</i> 名が同じで、検索列または戻す列の名前が異なる場合のみです。</p> <p>解決策: 第 2 のサブシステムに対して、<i>DataTable</i> の最初の列のサブストラクチャーとしてフィールドを追加します。この新規フィールドの名前は、第 1 のサブシステムの検索列と同じである必要があります。<i>DataTable</i> の 2 列目を、第 1 のサブシステム内の戻す列の名前でサブストラクチャー化します。この技法により、第 2 のサブシステム内でコードを変更せずに、共通関数を共用できます。</p>	<p>起こりうる問題 1: 検索列と戻す列の名前を指定する必要があります。EGL 検証の結果、欠落している列ごとに、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策: 関数を編集し、<i>DataTable</i> に正しい列名を指定します。</p> <p>起こりうる問題 2: 『関連パーツを使用したマイグレーション』に示した問題と同じことが起こる可能性があり、同じ解決策が適用されます。</p>

SET map PAGE ステートメント

VisualAge Generator: *SET map PAGE* は、表示マップと印刷マップの両方に使用されます。

EGL: 2 つの別個のステートメントを使用します。マップ名は指定しません。

- テキスト (表示) 書式の場合は `converseLib.clearScreen()`
- 印刷書式の場合は `converseLib.pageEject()`

マイグレーションに必要な関連パーツ: 装置タイプを判別するためにマップが必要です。マイグレーション・ツールは、使用可能ないずれかのマップ・グループにある、このマップ名をもつ最初のマップを使用します。プログラムのコンテキストでマイグレーションを行う場合、マイグレーション・ツールはメインのマップ・グループでのみプログラムを検索します。

表 52. *SET map PAGE* ステートメント

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、マップが使用可能であれば、マイグレーション・ツールは <i>SET map PAGE</i> を以下の関数の 1 つに変換します。</p> <ul style="list-style-type: none"> • テキスト書式の場合は <code>converseLib.clearScreen()</code> • 印刷書式の場合は <code>converseLib.pageEject()</code> <p>またマイグレーション・ツールは、オリジナルのマップ名を示すコメントを組み込みます。</p>	<p>マップが使用不可の場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • <i>SET map PAGE</i> を <code>converseLib.EZE_SETPAGE()</code> に変換します。 • オリジナルのマップ名を示すコメントを組み込みます。 • マップのタイプを識別できないことを示すエラー・メッセージを出します。

表 52. SET map PAGE ステートメント (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: 関数のマイグレーション時に決定されたものと異なるマップ・タイプを使用するプログラムは、実行時には異なる振る舞いをする可能性があります。これは、clearScreen() はテキスト書式にのみ適用され、pageEject() は印刷書式にのみ適用されるからです。EGL 検証の結果、「問題」ビューにエラー・メッセージは表示されません。プログラムの生成は失敗しません。</p> <p>考えられる解決策: 特定のターゲット環境で印刷を行い、その他の環境では常に表示マップを使用する場合は、EGL 関数を以下の例のように変更します。</p> <pre>if (sysVar.systemType is ZOSBATCH) converseLib.pageEject(); else converseLib.clearScreen(); end</pre> <p>ご使用のシステムの具体的な詳細に応じて、トランザクション・コード、ユーザー IDなどを基に同様なロジックを使用できます。</p>	<p>起こりうる問題 1: ステートメントを含む関数がプログラムで使用される場合、EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。関数がプログラムで使用されていない場合、「問題」ビューにメッセージは表示されません。</p> <p>解決策: 関数を編集し、EZE_SETPAGE() をマップ・タイプに応じて converseLib.clearScreen() または converseLib.pageEject() のいずれかに変更します。</p> <p>起こりうる問題 2: 『関連パーツを使用したマイグレーション』に示した問題と同じことが起こる可能性があり、同じ解決策が適用されます。</p>

SET mapItem 属性

VisualAge Generator: VisualAge Generator は、プリンター・マップ上の変数と定数に対して、保護、強調表示、色などの属性を許容します。

EGL: 下線を例外として、EGL は印刷書式の属性をサポートしません。

マイグレーションに必要な関連パーツ: 適用不可。

表 53. SET mapItem 属性

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>プリンター・マップのマイグレーション時に、マイグレーション・ツールは EGL がサポートしない印刷書式の属性を省略します。</p> <p>関数をマイグレーションする場合、マイグレーション・ツールは、マップが表示マップであるかプリンター・マップであるかに関係なく、SET ステートメントをマイグレーションします。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

表 53. SET mapItem 属性 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: テキスト書式の場合、問題はありません。印刷書式に対して、色、強調表示、保護などの属性を設定するロジックが関数に組み込まれている場合に限り、問題が起こります。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策: 関数が印刷書式のみに対して使用されている場合は、関数を変更して set ステートメントを除去します。関数がテキスト書式と印刷書式の両方に使用されている場合は、印刷書式に対して使用するための関数のコピーを作成します。新規関数を変更して set ステートメントを除去し、この新規関数を任意の印刷書式に使用します。</p> <p>欠点: set ステートメントを含む関数を使用する関数に、影響が及ぶ可能性があります。</p>	<p>起こりうる問題: 『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策が適用されます。</p>

IN がリテラルかスカラーかの検査

VisualAge Generator: VisualAge Generator は、IF ステートメントまたは WHILE ステートメントのデータ項目 IN がリテラルかスカラーかの検査をサポートします。この場合、VisualAge Generator は EZETST の値を設定し、等価かどうかの比較を行います。

EGL: EGL は、IN のデータ項目がリテラルかスカラーかの検査をサポートしません。

マイグレーションに必要な関連パーツ: 適用不可。

表 54. IN がリテラルかスカラーかの検査

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>リテラル内のデータ項目を検査する IF ステートメントまたは WHILE ステートメントの場合、マイグレーション・ツールは VAGen の動作と一致させるために以下の処理を行います。</p> <ul style="list-style-type: none"> • sysVar.arrayIndex を 0 に初期化するステートメントを追加します。 • if ステートメントまたは while ステートメントを同値比較に変更します (例: <code>if a == "b"</code>)。 • if または while の直後に、sysVar.arrayIndex を 1 に設定するステートメントを追加します。 <p>データ項目が別のデータ項目内にあるかどうかを検査する IF ステートメントまたは WHILE ステートメントに対して、マイグレーション・ツールは 2 番目のデータ項目が配列かスカラーかの判別を試みません。マイグレーション・ツールは、EGL の in 比較にマイグレーションします。(例: <code>if a in b</code>)。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

表 54. IN がリテラルかスカラーかの検査 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: 比較の対象がリテラルである場合、問題はありません。2 番目のデータ項目が実際にはスカラーである場合に限り、問題が起こります。この場合、EGL 検証の結果、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策: if または while ステートメントの前では sysVar.arrayIndex を 0 に初期設定し、if または while ステートメントの直後では sysVar.arrayIndex を 1 に設定するように関数を変更します。また、if または while ステートメントを、in ではなく == を使用する比較に変更します。</p>	<p>起こりうる問題: 『関連パーツを使用したマイグレーション』欄にリストした問題と同じことが起こる可能性があります。同じ解決策が適用されます。</p>

SQL 項目とマップ項目が NULL かどうかの検査

VisualAge Generator: IF、WHILE、および TEST は、SQL 項目またはマップ項目が NULL かどうかの検査をサポートします。

EGL: SQL 項目は **null** かどうかを検査できます。書式フィールドが **blanks** であるかどうかを検査できます。

マイグレーションに必要な関連パーツ: レコードまたはマップ。項目が修飾されていない場合は、プログラムと関連パーツすべてが必要です。

表 55. SQL 項目とマップ項目が NULL かどうかの検査

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、項目が修飾されていれば、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> 修飾子を検査して、レコードまたはマップのどちらであるかを判別します。 修飾子が SQL レコードである場合は、null の有無の検査に変換します。 修飾子がマップである場合は、blanks の有無の検査に変換します。 	<p>マイグレーション・ツールは、以下の方法で項目のタイプの判別を試みます。</p> <ul style="list-style-type: none"> 項目が修飾されているが、修飾子を使用できない場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> 修飾子が関数の入出力オブジェクトでもあるかどうかを検査します。そうである場合、CONVERSE オプションと DISPLAY 入出力オプションがあれば、入出力オブジェクトがマップであることが保証されます。CLOSE 入出力オプションは、レコードまたはマップのどちらにも有効です。その他の入出力オプションがあれば、入出力オブジェクトはレコードであることが保証されます。 関数パラメーター・リストとローカル・ストレージも検査します。修飾子が検出された場合、修飾子はレコードです。 マイグレーション・ツールは、項目が SQL レコードまたはマップのいずれにあるか判別できる場合は、次の値のどちらか 1 つにマイグレーションします。 <ul style="list-style-type: none"> SQL レコードの場合は null マップ項目の場合は blanks マイグレーション・ツールは、項目が SQL レコードまたはマップのいずれにあるのか判別できない場合、次の処理を行います。 <ul style="list-style-type: none"> EZE_NULL に変換します。 このステートメントの検討が必要であることを示すエラー・メッセージを出します。
<p>この関数の最初のマイグレーションを基準として、項目が修飾されていなければ、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> 関数パラメーター・リストを検査して、項目が SQLITEM パラメーターまたは MAPITEM パラメーターのどちらとしてこのリストに指定されているかを調べます。該当する場合、ツールはその結果を基礎としてマイグレーションを行います。 プログラムとその関連パーツが使用可能である場合、マイグレーション・ツールは VAGen 修飾規則を使用して、その項目を含むレコードまたはマップを判別し、その結果を基にマイグレーションを行います。 	<p>項目が修飾されていない場合、マイグレーション・ツールは関数パラメーター・リストを検査して、項目がそのリストで SQLITEM または MAPITEM のどちらとして指定されているかを調べます。</p> <p>マイグレーション・ツールは、項目が SQL レコードまたはマップのいずれにあるか判別できる場合は、次の値のどちらか 1 つにマイグレーションします。</p> <ul style="list-style-type: none"> SQL レコードの場合は null マップ項目の場合は blanks <p>マイグレーション・ツールは、項目が SQL レコードまたはマップのいずれにあるのか判別できない場合、次の処理を行います。</p> <ul style="list-style-type: none"> EZE_NULL に変換します。 このステートメントの検討が必要であることを示すエラー・メッセージを出します。

表 55. SQL 項目とマップ項目が NULL かどうかの検査 (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
起こりうる問題: なし。	<p>起こりうる問題 1: マイグレーション・ツールは EZE_NULL を使用している場合は、問題が起こります。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。</p> <p>解決策: 関数を編集し、EZE_NULL を SQL 項目の場合は null に変更し、書式変数フィールドの場合は blanks に変更します。</p>

入出力エラー値 UNQ および DUP

VisualAge Generator: UNQ と DUP は、非 SQL の場合は常にソフト・エラーであり、SQL の場合はハード・エラーです。SQL の場合、UNQ と DUP は -803 SQL コードに基づいて常に設定されます。関数に対して入出力エラー・ルーチンが指定されている場合、エラー・ルーチンは以下の環境での制御を獲得します。

- すべてのソフト・エラー
- すべてのハード・エラー (EZEFEFEC = 1 の場合)
- DL/I I/O の場合、すべてのハード・エラー (EZEDLERR または EZEFEFEC = 1 のとき)

EGL: duplicate は常にソフト・エラーであり、入出力が成功したことを示します。**unique** は常にハード・エラーであり、入出力が失敗したことを示します。**duplicate** は、SQL の場合はサポートされません。**try** ブロックおよび **onException** ステートメントがエラー処理に使用されます。**onException** ステートメントが I/O ステートメントに対して指定されている場合は、**onException** ステートメントは、以下のどの環境でも制御を獲得します。

- すべてのソフト・エラー
- **vgVar.handleHardIOErrors** が 1 に設定されている場合のすべてのハード・エラー
- DL/I I/O の場合は、**dliVar.handleHardDLIErrors** または **vgVar.handleHardIOErrors** が 1 に設定されている場合のすべてのハード・エラー

マイグレーションに必要な関連パーツ: ステートメント内で使用されているレコード。

表 56. 入出力エラー値 UNQ および DUP

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>この関数の最初のマイグレーションを基準として、レコードが使用可能であれば、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> レコードが非 SQL の場合、マイグレーション・ツールは DUP を duplicate に、UNQ を unique に変更します。 レコードが SQL の場合、マイグレーション・ツールは DUP と UNQ を両方とも unique に変更します。 	<p>レコードが使用不可の場合、マイグレーション・ツールは以下の方法でレコードのタイプの判別を試みます。</p> <ul style="list-style-type: none"> ステートメントが関数の入出力オブジェクトと同じレコードを指定している場合、マイグレーション・ツールは、その関数が SQL 文節も持っているかどうか、または Execution time statement build、single row select、Declare cursor with hold、あるいは UPDATE/SETUPD 関数などの SQL 固有の情報を持っているかどうかを検査します。これらを持っている場合、マイグレーション・ツールは、そのレコードが SQL であると想定して、DUP と UNQ を unique に変換します。 以下に示すような、その他の状態では、マイグレーション・ツールはレコード・タイプを判別できません。 <ul style="list-style-type: none"> レコードが関数の入出力オブジェクトとして使用されていて、関数に SQL 固有の情報がない場合。 レコードが関数の入出力オブジェクトとして使用されていない場合。 <p>前述の状態や、マイグレーション・ツールがレコード・タイプを判別できないその他の状態では、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> UNQ を unique に変換します。 DUP を EZE_DUPLICATE に変換し、エラー・メッセージを発行します。

表 56. 入出力エラー値 UNQ および DUP (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題: 問題が発生するのは、同じレコード名に対して、SQL 用に 1 つ、非 SQL 用に 1 つの異なる定義が、おそらく異なるサブシステムに存在する場合のみです。関数のマイグレーション時に非 SQL レコードが使用可能であれば、その関数を SQL レコードと一緒に使用して、duplicate の有無を検査すると、EGL 検証がエラー・メッセージを表示します。◀関数のマイグレーション時に SQL レコードが使用可能であれば、重複検査によって得られる追加情報を非 SQL レコードに使用することはできません。</p> <p>考えられる解決策: 関数をコピーして、オリジナルの関数を SQL に対して使用し、新規関数を非 SQL に対して使用します。</p> <p>欠点: UNQ または DUP の検査を行っていたオリジナルの関数を使用する関数に、影響が波及する可能性があります。</p> <p>SQL の場合に起こりうる問題: なし。DUP と UNQ は常に同じ方法で設定されており、unique は引き続きハード・エラーになります。</p> <p>非 SQL の場合に起こりうる問題 1: プログラムに対して vgVar,handleHardIOErrors (EZEFECE) を 1 に設定しなかった場合は、問題が起こります。この場合、unique はハード・エラーになるので、onException ステートメントは制御を獲得せず、プログラムは終了します。</p> <p>解決策: プログラムが vgVar,handleHardIOErrors を 1 に設定することを確認します。</p> <p>非 SQL の場合に起こりうる問題 2: hardIOError を明示的にテストしている場合にも、問題が起こります。この場合、unique はハード・エラーになるので、従来の VisualAge Generator では hardIOError のテスト結果が true にならなくても、EGL ではテスト結果が true になる場合があります。検証時と生成時に、エラーは検出されません。ただし、このプログラムは、VisualAge Generator の場合と同じようには実行されない可能性があります。</p> <p>考えられる解決策: プログラム・ロジックの中で入出力エラー値のテストを再配列する必要がある場合があります。</p>	<p>起こりうる問題 1: EZE_DUPLICATE は EGL では無効です。</p> <p>解決策: 関数を編集し、EZE_DUPLICATE をレコード・タイプに応じて duplicate または unique に変更します。</p> <p>起こりうるその他の問題: 『関連パーツを使用したマイグレーション』に示した問題と同じことが起こる可能性があり、同じ解決策が適用されます。</p>

入出力エラー値 LOK

VisualAge Generator: LOK は、OS/400 の場合は常にソフト・エラーです。関数に対して入出力エラー・ルーチンが指定されている場合、エラー・ルーチンは以下の環境での制御を獲得します。

- すべてのソフト・エラー
- すべてのハード・エラー (EZEFEFEC = 1 の場合)

EGL: LOK は **deadlock** に置き換えられますが、これはハード・エラーになります。try ブロックおよび **onException** ステートメントがエラー処理に使用されます。**onException** ステートメントが入出力ステートメントに対して指定されている場合は、以下の環境で **onException** ステートメントが制御を獲得します。

- すべてのソフト・エラー
- **vgVar.handleHardIOErrors** が 1 に設定されている場合の、すべてのハード・エラー

マイグレーションに必要な関連パーツ: 適用不可。

表 57. 入出力エラー値 LOK

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
マイグレーション・ツールは、常に LOK を deadlock に変更します。	マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。
<p>起こりうる問題 1: プログラムに対して vgVar.handleHardIOErrors (EZEFEFEC) を 1 に設定しなかった 場合は、問題が起こります。この場合、deadlock はハード・エラーになるので、onException ステートメントは制御を獲得せず、プログラムは終了します。</p> <p>解決策: プログラムが vgVar.handleHardIOErrors を 1 に設定することを確認します。</p> <p>起こりうる問題 2: hardIOError を明示的にテストしている場合にも、問題が起こります。この場合、deadlock はハード・エラーになるので、従来の VisualAge Generator では hardIOError のテスト結果が true にならなくても、EGL ではテスト結果が true になることがあります。検証時と生成時に、エラーは検出されません。ただし、このプログラムは、VisualAge Generator の場合と同じようには実行されない可能性があります。</p> <p>考えられる解決策: プログラム・ロジックの中で入出力エラー値のテストを再配列する必要がある場合があります。</p>	『関連パーツを使用したマイグレーション』欄で説明した問題と同じことが起こる可能性があります。同じ解決策を使用できます。

EZE ワードの未確定状態の処理

一部の EZE ワードを置換するためには、マイグレーション・ツールが余分の項目変数を宣言する必要があります。マイグレーション・ツールは、これらの新しい項目変数をプログラムに追加します。これにより、プログラム内の任意の関数がこの変数を使用できるようになります。

EZELTERM

VisualAge Generator: EZELTERM は Web トランザクション・プログラム内の会話 ID であり、他のすべてのプログラム・タイプでは端末 ID です。

EGL: `sysVar.conversationID` は、VGWebTransaction プログラム用の会話 ID です。`sysVar.terminalID` は、その他すべてのプログラム・タイプの端末 ID です。

`sysVar.conversationID` と `sysVar.terminalID` は同義語として扱われるので、いずれを使用してもプログラム・タイプに応じて正しい情報が提供されます。

マイグレーションに必要な関連パーツ: プログラム。

表 58. EZELTERM

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
この関数の最初のマイグレーションを基準として、プログラムが使用可能であれば、マイグレーション・ツールは次の方法でプログラム・タイプに基づいて EZELTERM を変換します。 <ul style="list-style-type: none">プログラムが Web トランザクション・プログラムである場合、マイグレーション・ツールは <code>sysVar.conversationID</code> を使用します。プログラムが Web トランザクション・プログラムでない場合、マイグレーション・ツールは <code>sysVar.terminalID</code> を使用します。	プログラムが使用不可であれば、マイグレーション・ツールは常に EZELTERM を <code>sysVar.terminalID</code> に変換します。
起こりうる問題: なし。 <code>sysVar.conversationID</code> と <code>sysVar.terminalID</code> は同義語として扱われます。	起こりうる問題: なし。 <code>sysVar.conversationID</code> と <code>sysVar.terminalID</code> は同義語として扱われます。

EZESYS

VisualAge Generator: EZESYS は、VisualAge Generator によって指定されるリテラル値を含む IF、WHILE、および TEST の各ステートメント内で一般に使用されます。ただし、EZESYS はその他のステートメント内でも許容されます。

EGL: EGL システム変数 `sysVar.systemType` の値は、VisualAge Generator での値とは異なります。IF、WHILE、および TEST 以外のステートメント内で EZESYS が使用されている場合、マイグレーション・ツールはプログラムが予期している値を判別できないので、オリジナルの VAGen 値を使用する必要があります。EGL システム・ライブラリー関数 `vgLib.getVGSystemType` は、以前の VAGen 値を提供します。

マイグレーションに必要な関連パーツ: 適用不可。

表 59. EZESYS

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>プログラムをマイグレーションするとき、マイグレーション設定の「以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)」が選択されていないとマイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • <code>custPrefixEZESYS</code> の宣言を組み込みます。 • <code>custPrefixEZESYS</code> の値を以前の VAGen EZESYS 値に設定する初期化ステートメントを組み込みます。 <p>この設定が選択されている場合、マイグレーション・ツールは宣言または初期化ステートメントを組み込みません。</p> <p><code>custPrefix</code> は、予約語と競合するパーツ名の変更に使用されるものと同じ接頭部です。<code>custPrefix</code> の値を設定するには、VAGen マイグレーション設定を使用します。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>この関数の最初のマイグレーションを基準として、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • EZESYS が IF、WHILE、または TEST のいずれかのステートメント内で使用されている場合、マイグレーション・ツールは EZESYS を次のように変換します。 <p><code>sysVar.systemType</code></p> <p>マイグレーション・ツールは、EZESYS 値を等価な EGL 値に変換します。EZESYS 値に等価な EGL 値がない場合、マイグレーション・ツールはその値を「現状のまま」マイグレーションします。例えばマイグレーション・ツールは、MVS BATCH を EGL の同等の ZOS BATCH に変換します。マイグレーション・ツールは、OS2 と NTCICS を VisualAge Generator の場合と同じ値にマイグレーションします。変換される値について詳しくは、396 ページの表 125 を参照してください。</p> <ul style="list-style-type: none"> • EZESYS がその他のステートメント内で使用されている場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> – 以前の VAGen EZESYS 値が結果として使用されることを示す警告メッセージを出します。 – 次の値を使用して、 <code>custPrefixEZESYS</code> <p>ステートメント内の EZESYS を置き換えます。</p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

表 59. EZESYS (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>起こりうる問題 1: マイグレーション設定「以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)」を選択し、IF、WHILE、または TEST 以外のステートメントで EZESYS を使用する場合に問題が起こります。この状態では、マイグレーション済みツールはステートメントで <code>custPrefixEZESYS</code> を使用しますが、プログラムには <code>custPrefixEZESYS</code> の宣言および初期化ステートメントがありません。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。</p> <p>考えられる解決策 1A: EGL ロジックを、<code>sysVar.systemType</code> の新規の値を使用するように変更します。</p> <p>考えられる解決策 1B: <code>custPrefixEZESYS</code> の宣言および初期化ステートメントを、EZESYS の以前の VAGen 値の使用が必要なプログラムに追加します。</p> <p>起こりうる問題 2: if および while 以外のステートメントで新規の EGL 値を使用すると、問題が発生します。</p> <p>考えられる解決策 2: 関数を修正して、<code>custPrefixEZESYS</code></p> <p>の代わりに <code>sysVar.systemType</code> を使用するようにロジックを変更します。比較に使用するすべての DataTable、ファイル、またはデータベース内で、以前の VAGen 値を必ず新規の EGL 値に変更してください。</p>	<p>『関連パーツを使用したマイグレーション』欄で説明した問題と同じことが起こる可能性があります。同じ解決策を使用できます。</p>

EZEWAIT

VisualAge Generator: EZEWAIT は、待機秒数を 100 分の 1 秒単位で指定します。

EGL: `sysLib.wait`。EZEWAIT の置換表現で、待機時間を秒数単位で指定します。

マイグレーションに必要な関連パーツ: 適用不可。

表 60. EZEWAIT

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>プログラムをマイグレーションする際に、マイグレーション・ツールは常に次の宣言を組み込みます。</p> <p><code>custPrefixEZE_WAIT_TIME.</code></p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>

表 60. EZEWAIT (続き)

関連パーツを使用したマイグレーション	関連パーツを使用しないマイグレーション
<p>関数をマイグレーションする際に、EZEWAIT が使用されていれば、マイグレーション・ツールは待機時間を秒数単位で計算し、結果を以下に格納するロジックを組み込みます。</p> <p><code>custPrefixEZE_WAIT_TIME.</code></p>	<p>マイグレーション・ツールは、『関連パーツを使用したマイグレーション』欄の説明と同じ処理を行います。</p>
<p>起こりうる問題: なし。ただし、関数を新規プログラム内で使用する場合は、次の宣言をプログラムに必ず組み込んでください。</p> <p><code>custPrefixEZE_WAIT_TIME</code></p>	<p>『関連パーツを使用したマイグレーション』欄で説明した問題と同じことが起こる可能性があります。</p>

第 2 部 VisualAge Generator 4.5 on Java から EGL へのマイグレーション

第 4 章 ステージ 1 - Java からの抽出

ソース・コードを VisualAge Generator から抽出する前に、VisualAge for Java 上で稼働するステージ 1 マイグレーション・ツールをインストールする必要があります。また、VisualAge Generator 4.5 (VAGen 4.5) から EGL にマイグレーションするデータの保管に使用される、DB2 マイグレーション・データベースも作成する必要があります。

VisualAge for Java へのステージ 1 マイグレーション・ツールのインストール

VisualAge Generator から EGL へのステージ 1 マイグレーション・ツールは、VAGenMigJava.exe という名前の自己解凍ファイルとして提供されます。このファイルをインストールするには、以下の手順を実行します。

1. フィックスパック 5 を適用した VisualAge Generator 4.5 にアップグレードする。お客様の状況に応じてさらに VisualAge Generator APAR が必要になる可能性がありますので、545 ページの『付録 F. VisualAge Generator に必要な APAR』を参照してください。
2. 使用しているシステム上で、VisualAge for Java がインストールされている場所を判別する。
3. VisualAge for Java をシャットダウンする。
4. 自己解凍ファイル VAGenMigJava.exe を実行する。このファイルは、以下のディレクトリーにあります。

`\installationDirectory\bin`

注: 現在ご使用の製品をインストールする前に、前のバージョンの開発者製品をインストールし、保持していた場合、対象のインストール・ディレクトリーは、以前のインストール時に使用されたディレクトリーである可能性があります。

5. GUI プロンプトが表示されたら、VisualAge for Java がインストールされているドライブとディレクトリーにナビゲートする (例えば、`c:\Program Files\IBM\VisualAge for Java`)。その後で、「**Unzip**」をクリックする。

自己解凍型の実行可能ファイルが実行されると、次のファイルが VisualAge for Java インストール・ディレクトリーに抽出されます。

- `%ide%\vgmigration\checkStage1.bat`
- `%ide%\vgmigration\checkStage1.sql`
- `%ide%\vgmigration\createdatabase.sql`
- `%ide%\vgmigration\createindex.sql`
- `%ide%\vgmigration\createtables.sql`
- `%ide%\vgmigration\deletemigsets.bat`
- `%ide%\vgmigration\MigPreferences.xml`
- `%ide%\vgmigration\runStats.bat`

- %ide%\vgmigration\SetupDatabase.bat
- %ide%\vgmigration\SetupIndex.bat
- %ide%\vgmigration\SetupTables.bat
- %ide%\vgmigration\VGMigReserved.txt
- %ide%\features\com-ibm-vgj-mig\

この最後のディレクトリーは、Java 上でのステージ 1 マイグレーション・ツール用のフィーチャーを格納しています。また、Java 上でのステージ 1 マイグレーション・ツールに使用される .xml ファイル、および対応する .dtd ファイルも格納しています。

マイグレーション・フィーチャーの追加

ステージ 1 マイグレーション・ツールを使用するには、以下の手順に従って、**IBM VisualAge Generator EGL** マイグレーション・フィーチャーを追加する必要があります。

1. VisualAge Generator on Java を開始する。
2. 以下の手順を実行して、「**IBM VisualAge Generator EGL** マイグレーション」フィーチャーを追加する。
 - a. 「ワークベンチ」ウィンドウで、F2 を押す。
 - b. 左側の列から「フィーチャー」を選択し、右側の列にある「フィーチャーの追加」を選択する。「OK」をクリックします。
 - c. 「**IBM VisualAge Generator EGL** マイグレーション - バージョン番号 (**IBM VisualAge Generator EGL Migration - versionNumber**)」を選択する。「OK」をクリックします。マイグレーション・フィーチャーがロードされます。
 - d. ワークベンチの「プロジェクト」タブをクリックする。「**IBM VisualAge Generator EGL** マイグレーション」プロジェクトがワークスペースに表示されます。

注: リモート・リポジトリを使用していて、マイグレーション・フィーチャーの追加中に問題が発生した場合は、以下の手法のいずれかを試してください。

- 以下のディレクトリー

`VJavaInstallDirectory\ide\features\com-ibm-vgj-mig`

を、リモート・リポジトリを使用するマシン上のフィーチャー・ディレクトリーにコピーする。次に、前に説明したとおりに「**フィーチャーの追加**」ステップを試みる。複数の開発者がステージ 1 マイグレーション・ツールをロードする必要がある場合は、この手法をお勧めします。

- V4.5 FP5 の **IBM VisualAge Generator** ユーティリティーがワークスペースにロードされていることを確認する。次に、以下の手順を実行する。
 1. 「ワークベンチ」ウィンドウで、「ファイル」->「インポート」をクリックする。
 2. 「リポジトリ」をクリックしてから、「次へ」をクリックする。
 3. 「ローカル・リポジトリ」をクリックし、以下のディレクトリーの .dat ファイルを指示する。

VAJavaInstallDirectory\ide\features\com-ibm-vgj-mig

4. 「プロジェクト」をクリックして、「詳細」をクリックする。この .dat ファイルにあるプロジェクトバージョンのみを選択してから、「OK」をクリックする。「最新のプロジェクト・エディションをワークスペースに追加 (Add most recent project edition to workspace)」チェック・ボックスを選択する。「終了」をクリックします。

「IBM VisualAge Generator EGL マイグレーション」プロジェクトをワークスペースに追加する必要があります。

マイグレーション・データベースの作成

マイグレーション・データベースの作成については、547 ページの『DB2 マイグレーション・データベースの作成』を参照してください。VisualAge Java インストール・ディレクトリーの %ide%\vgmigration サブディレクトリーにある、SetupDatabase.bat ファイルと SetupTables.bat ファイルを使用する必要があります。

ステージ 1 設定の実行

ステージ 1 マイグレーション・ツールを VisualAge for Java にインストールするときに、インストール・プロセスによってサンプル設定ファイル MigPreferences.xml がディレクトリー VisualAge-Java-installation-directory\%ide%\vgmigration に作成されます。設定を変更する前に、バックアップのために MigPreferences.xml ファイルのコピーを作成しておく必要があります。MigPreferences.xml ファイルを VisualAge for Java インストール・ディレクトリーの外側のディレクトリーにコピーして、そのコピーに対して変更を加えることもできます。これによって、新規バージョンのマイグレーション・ツールをインストールした場合に変更内容を誤って上書きしてしまうことを防止できます。

テキスト・エディター、またはステージ 1 マイグレーション・ツールに付属の GUI エディターを使用して、MigPreferences.xml ファイルを編集できます。GUI エディターを使用するには、以下の手順を実行します。

1. VisualAge Generator for Java を開始する。
2. 「ワークベンチ」ウィンドウの「プロジェクト」タブをクリックする。
3. 「IBM VisualAge Generator EGL マイグレーション」プロジェクトにナビゲートする。マイグレーション・プロジェクトを展開し、com.ibm.vgj.mig パッケージを展開する。
4. このパッケージ内で、「PreferencesUI」クラスを選択する。
5. 「PreferencesUI クラスを右クリックしてから、「プロパティ」をクリックする。
6. 「プログラム」タブをクリックする。
7. 「プログラム」ページの「コマンド行引数」フィールドに以下のように指定して、編集する MigPreferences.xml ファイルを指示する。

-p filename

ここで、filename は MigPreferences.xml ファイルのドライブ、ディレクトリー、およびファイル名です。

8. 「OK」をクリックして、プロパティを保存する。

9. 「**PreferencesUI**」クラスを右クリックしてから、「実行」->「メインを実行」をクリックする。(または、ツールバーの走る人のアイコンをクリックすることもできます。) ステージ 1 GUI 設定エディターが開き、プログラムのプロパティ内で指定したファイルがロードされます。

注:

- 現在、Project List Parts (PLPs) を使用していない場合は、169 ページの『マイグレーション・プランと上位 PLP プロジェクト』を参照してください。
- ドライブとディレクトリーを必要とする設定の場合は、次に示す 2 つの方法のどちらかで情報を指定できます。

- 絶対パス。例: `d:\tempMig\MySystem\`

- 相対パス。この場合、パスは作業ディレクトリーに対する相対パスです。例えば、`..\tempMig\MySystem` は次のパスに相当します。

```
VisualAge-Java-installation-directory\ide\project_resources  
  \IBM VisualAge Generator EGL Migration\tempMig\MySystem.
```

- ログ・ファイル、デバッグ・ファイル、およびレポート・ファイル用のドライブとディレクトリーを指定しない場合、ファイルは次の作業ディレクトリーに書き込まれます。

```
VisualAge-Java-installation-directory\ide\project_resources  
  \IBM VisualAge Generator EGL Migration
```

以降では、変更できる設定について、GUI 内でその設定が表示されるページ別に説明します。

- 「プランの作成」ページ
- 「マッピング」ページ
- 「名前変更」ページ
- 「実行」ページ

「プランの作成」ページ

「プランの作成 (Build Plans)」ページでは、ステージ 1 マイグレーション・ツールがマイグレーション・プラン・ファイルの読み取りまたは書き込みを行う場所、およびリポジトリからマイグレーションするプロジェクトとバージョンを指定します。

マイグレーション仕様

「マイグレーション仕様」セクションでは、ステージ 1 ツールがリポジトリ・フィルターに基づいて作成した 1 つまたは複数のマイグレーション・プラン・ファイルを、マイグレーション・ツールが書き込む場所を指定します。また、マイグレーション・プラン・ファイルが既に作成済みである場合、「マイグレーション仕様」はマイグレーション・ツールがマイグレーション・プラン・ファイルを読み取る場所を指定します。

注:

- マイグレーション・プラン・ファイルのファイル拡張子は、マイグレーション・データベースのロードに使用される前は `.pln` で、正常に処理された後は `.done` になります。

- **VAGenToEGLMigration** クラス (実際のステージ 1 マイグレーション・ツール) に対する `-o` (オーバーライド) オプションの設定については、168 ページの『ステージ 1 ツールの実行』を参照してください。

プラン・ディレクトリー

ステージ 1 マイグレーション・ツールがマイグレーション・プラン・ファイルを配置するターゲット・ディレクトリー、またはステージ 1 ツールが既存のマイグレーション・プラン・ファイルを検索するターゲット・ディレクトリー。

プラン・ファイル名

マイグレーション・データベースをロードするために作成または使用する、マイグレーション・プラン・ファイルのファイル名 (オプション)。ステージ 1 マイグレーション・ツールの実行時には、**VAGenToEGLMigration** クラスに対して指定した `-o` (オーバーライド) オプションと組み合わされて、このファイル名が以下のようにして使用されます。

- **VAGenToEGLMigration** クラスのプロパティーに `-o` オプションを組み込んだ場合、ステージ 1 マイグレーション・ツールは、マイグレーション仕様で指定したファイル名に基づいて以下の新規プラン・ファイルを作成します。
 - 「プラン・ファイル名」を指定しない場合、マイグレーション・ツールは指定した「プラン・ディレクトリー」にあるすべての `.pln` ファイルを削除してから、新規プラン・ファイルを作成します。マイグレーション・ツールは、それぞれのマイグレーション・セットごとにプラン・ファイルを 1 つずつ作成します。この場合、マイグレーション・プラン・ファイル名の形式は `migrationSetName_version.pln` です。
 - 「プラン・ファイル名」を指定した場合、マイグレーション・ツールは指定の「プラン・ディレクトリー」から指定の `.pln` ファイルのみを削除してから、指定の「プラン・ファイル名」を使用して新規 `.pln` ファイルを作成します。この場合は、単一のプラン・ファイルにマイグレーション・セットがすべてリストされます。

リポジトリ・フィルターと上位 PLP プロジェクトに基づいて、マイグレーション・プラン・ファイルをステージ 1 マイグレーション・ツールに作成させたい場合は、`-o` オプションを使用します。PLP プロジェクト作成のために支援が必要な場合は、170 ページの『上位 PLP プロジェクトの作成』を参照してください。

- **VAGenToEGLMigration** クラスのプロパティーから `-o` オプションを省略した場合、ステージ 1 マイグレーション・ツールはマイグレーション・プラン・ファイルを新規に作成しません。代わりに、ステージ 1 マイグレーション・ツールは、「マイグレーション仕様」で指定した「プラン・ディレクトリー」および「プラン・ファイル名」に基づいて、以下のようにして既存のプラン・ファイルを使用します。

- 「プラン・ファイル名」を指定していない場合は、指定した「プラン・ディレクトリー」内のすべての .pln ファイルを使用してマイグレーション・ツールが実行されます。
- 「プラン・ファイル名」を指定した場合は、指定した「プラン・ディレクトリー」内のその .pln ファイルのみを使用してマイグレーション・ツールが実行されます。

マイグレーション・プラン・ファイルがすでに作成済みで、これらのファイルを使用してステージ 1 マイグレーション・ツールを実行し、マイグレーション・データベースをロードする場合は、-o オプションを省略します。ユーザー独自のマイグレーション・プラン・ファイルの作成について詳しくは、172 ページの『マイグレーション・プラン・ファイルの手動作成』を参照してください。

リポジトリ・フィルター

このセクションを使用すると、ご使用の Java リポジトリでステージ 1 マイグレーション・ツールによって処理されるプロジェクトおよびバージョンを制御することができます。プロジェクトとバージョンを制限することにより、ステージ 1 マイグレーション・ツールのパフォーマンスを大幅に向上できます。複数のフィルターを指定できます。ステージ 1 マイグレーション・ツールは、以下のようにして「プロジェクト」フィルターおよび「バージョン深さ」または「バージョン名」フィルターを使用します。

- マイグレーション・ツールは、リポジトリ内の各 VAGen プロジェクトを「プロジェクト」フィルターと突き合わせます。
 - プロジェクト名が「プロジェクト」フィルターのいずれにも一致しない場合、そのプロジェクトは以後の処理対象になりません。
 - プロジェクト名が「プロジェクト」フィルターの少なくとも 1 つと一致する場合、そのプロジェクトのバージョンは以下のようにして処理されます。
 - 「バージョン深さ」フィルターを選択している場合、プロジェクトの最新のいくつかのバージョン（「バージョン深さ」フィルターで指定した数まで）が以後の処理対象になります。デフォルトの「バージョン深さ」フィルターは 1 です。
 - 「バージョン名」フィルターが選択されている場合は、プロジェクトの各バージョン名と「バージョン名」フィルターのリストとの突き合わせが行われます。バージョン名が「バージョン名」フィルターのいずれかに一致する場合、そのバージョンは以後の処理対象になります。

注: 「バージョン深さ」と「バージョン名」は相互に排他的です。デフォルトでは、「バージョン名」フィルターが MigPreferences.xml ファイルに組み込まれています。「バージョン深さ」フィルターを使用する場合は、「バージョン深さ」ラジオ・ボタンをクリックして、マイグレーションするバージョンの数を指定します。

- プロジェクト名とバージョン名により以後の処理対象のプロジェクト・バージョンが指定された場合、ステージ 1 マイグレーション・ツールは以下のようにしてプロジェクト・バージョンを処理します。
 - プロジェクト・バージョンが上位 PLP プロジェクトならば、ステージ 1 マイグレーション・ツールはそのプロジェクト・バージョンをマイグレーション・セットの作成の基礎として使用します。バージョン名がバージョン・フィルターとマッチングすることを前提とすると、上位 PLP プロジェクトのそれぞれのバージョンごとに、異なるマイグレーション・セットが作成されます。
 - プロジェクト・バージョンが上位 PLP プロジェクトでなければ、プロジェクト・バージョンは以後の処理対象になりません。そのプロジェクト・バージョンは他のマイグレーション・セットにも組み込まれたままになる可能性があります、このプロジェクト・バージョン用にマイグレーション・セットが特別に作成されることはありません。

以下のようにして「リポジトリ・フィルター」情報を指定します。

- 「**プロジェクト**」フィルター。 マイグレーション・ツールは、リポジトリ内のプロジェクト名と指定した「**プロジェクト**」フィルター とを突き合わせます。「**プロジェクト**」フィルターは複数個指定することができます。フィルターを追加または除去するには、「**追加**」または「**除去**」ボタンを使用します。フィルターを更新するには、テーブル内で上書き入力します。フィルターに大/小文字の区別はありません。以下のようにして、ワイルドカードを使用することができます。
 - プロジェクト・フィルター `*xyz*` は、リポジトリ内にある、ストリング「xyz」を名前のどこかに含むプロジェクト名すべてとマッチングします。
 - プロジェクト・フィルター `xyz*` は、リポジトリ内にある、「xyz」から始まるプロジェクト名すべてとマッチングします。
 - プロジェクト・フィルター `*xyz` は、リポジトリ内にある、「xyz」で終わるプロジェクト名すべてとマッチングします。
- **バージョン深さ** フィルター。「**バージョン深さ**」フィルターを選択している場合にプロジェクト名が「**プロジェクト**」フィルターのいずれかと一致すると、ステージ 1 マイグレーション・ツールは「**バージョン深さ**」として指定した数のバージョンを処理します。デフォルトは 1 で、この場合ステージ 1 マイグレーション・ツールはプロジェクトの最新バージョンのみを処理します。
- **バージョン名** フィルター。「**バージョン名**」フィルターを選択している場合にプロジェクト名が「**プロジェクト**」フィルターのいずれかと一致すると、ステージ 1 マイグレーション・ツールは「**バージョン名**」フィルターを使用して、マイグレーション対象になるプロジェクト・バージョン(存在する場合)を判別します。「**バージョン名**」フィルターは複数個指定することができます。フィルターを追加または除去するには、「**追加**」または「**除去**」ボタンを使用します。フィルターを更新するには、テーブル内で上書き入力します。フィルターに大/小文字の区別はありません。以下のようにして、ワイルドカードを使用することができます。

- バージョン名フィルター `*xyz*` は、ストリング「xyz」をバージョン名のどこかに含むプロジェクト・バージョン名すべてとマッチングします。
- バージョン名フィルター `xyz*` は、「xyz」から始まるプロジェクト・バージョン名すべてとマッチングします。
- バージョン名フィルター `*xyz` は、「xyz」で終わるプロジェクト・バージョン名とマッチングします。

「マッピング」ページ

「マッピング」ページを使用すると、EGL ファイル内でのパーツの配置、マイグレーション時に作成されるいくつかの EGL プロジェクト、パッケージ、およびファイルの名前を制御することができます。

ファイル名

このセクションを使用すると、マイグレーション時に作成される 2 つの EGL ファイルの名前を制御することができます。

共通パーツ

マイグレーション・セットの有効範囲内で、複数の固有な生成可能パーツに共通するパーツを格納する EGL ファイルの名前を指定することができます。ファイル名は、拡張子またはパスを付けずに指定します。マイグレーション・ツールは、マイグレーション・セットにある複数の生成可能パーツによって使用されて (関連付けられて) いるパーツを含んでいるか、または共通プロジェクトまたはパッケージとして指定された VAGen プロジェクトまたはパッケージ内のパーツを含んでいる、それぞれの EGL パッケージ内に、1 つの共通パーツ・ファイルを作成します。パーツがプログラムとともに配置されるか、または共通パーツ・ファイル内に配置されるかの判別について詳しくは、50 ページの『EGL ファイル内でのパーツの配置』を参照してください。

未使用パーツ

マイグレーション・セットの有効範囲内で、使用されていないパーツを格納する EGL ファイルの名前を指定することができます。ファイル名は、拡張子またはパスを付けずに指定します。対応する VAGen プロジェクトまたはパッケージが共通のプロジェクトまたはパッケージとして指定されていないことを条件として、マイグレーション・セットにある生成可能パーツによって使用されて (関連付けられて) いないパーツを含む、それぞれの EGL パッケージ内で未使用パーツ・ファイルがマイグレーション・ツールによって作成されます。

スパン・マップ

このセクションでは、1 つのマップ・グループに複数のプロジェクトまたはパッケージのマップが含まれている場合に使用される接尾部を指定することができます。

プロジェクト接尾部

新規 EGL プロジェクト名を作成するための接尾部を指定することができます。ステージ 1 マイグレーション・ツールは、この接尾部

をマイグレーション・セット名に連結して新規 EGL プロジェクト名を作成します。マイグレーション・ツールは、マップ・グループとそのマップがマイグレーション・セット内で複数の VAGen プロジェクトに存在する場合に限り、この新規 EGL プロジェクトを作成します。新規プロジェクト名は、*migrationSetName_ProjectSuffix* です。マイグレーション・ツールは、「名前変更規則」がすべて適用された後、マイグレーション・セット名に接尾部を連結します。

パッケージ接尾部

EGL プロジェクト内で新規 EGL パッケージ名を作成するための接尾部を指定することができます。ステージ 1 マイグレーション・ツールは、この接尾部をプロジェクト名に連結して新規 EGL パッケージ名を作成します。マイグレーション・ツールは、マップ・グループとそのマップがプロジェクト内で複数の VAGen パッケージに存在する場合に限り、この新規 EGL パッケージを作成します。新規パッケージ名は、*projectName.PackageSuffix* です。マイグレーション・ツールは、「名前変更規則」がすべて適用された後、プロジェクト名に接尾部を連結します。

共通 ID

このセクションでは、共通 (共用) パーツを含む VAGen プロジェクトとパッケージを判別するためにマイグレーション・ツールが使用できる、ワイルドカードを含むストリングのリストを指定することができます。

プロジェクト

「プロジェクト」リストには、共通パーツを含むプロジェクトを識別するストリングのリストを指定できます。マイグレーション・ツールは、このストリングのリストをマイグレーション・セット内の各プロジェクト名と突き合わせて、プロジェクトに共通パーツが含まれているかどうかを判別します。ストリングがプロジェクト名にマッチングする場合、そのプロジェクト内のパーツはすべて「使用済み」と見なされます。生成不可のパーツはそれぞれ、プログラム・ファイル内または「共通パーツ」設定で指定されたファイル内に置かれます。そのパーツは、マイグレーション・セット内の生成可能パーツによって使用されていない場合でも、未使用パーツ・ファイル内には置かれません。「プロジェクト」フィルターは複数個指定することができます。フィルターを追加または除去するには、「追加」または「除去」ボタンを使用します。フィルターを更新するには、テーブル内で上書き入力します。フィルターに大/小文字の区別はありません。ストリングの先頭または末尾で、* をワイルドカードとして使用することもできます。

パッケージ

「パッケージ」リストには、共通パーツを含むパッケージを識別するストリングのリストを指定できます。マイグレーション・ツールは、このストリングのリストをマイグレーション・セット内の各パッケージ名と突き合わせて、パッケージに共通パーツが含まれているかどうかを判別します。ストリングがパッケージ名にマッチングする場合、そのパッケージ内のパーツはすべて「使用済み」と見なされます。生成不可のパーツはそれぞれ、プログラム・ファイル内または「共通パーツ」設定で指定されたファイル内に置かれます。

そのパーツは、マイグレーション・セット内の生成可能パーツによって使用されていない場合でも、未使用パーツ・ファイル内には置かれませんが、「パッケージ」フィルターは複数個指定することができます。フィルターを追加または除去するには、「追加」または「除去」ボタンを使用します。フィルターを更新するには、テーブル内で上書き入力します。フィルターに大/小文字の区別はありません。ストリングの先頭または末尾で、* をワイルドカードとして使用することもできます。

「名前変更」ページ

「名前変更」ページを使用して、プロジェクト名、パッケージ名、およびバージョン名を変更するための規則を指定できます。「名前変更規則 (Renaming Rules)」セクションでは、VAGen プロジェクトとパッケージの名前から得られる EGL プロジェクトとパッケージの名前を制御できます。「順序 (order)」列の番号は、ステージ 1 マイグレーション・ツールが名前変更規則を適用する順序を示し、最も小さい番号の規則が最初に適用されます。名前変更規則を追加または除去するには、「追加」または「除去」ボタンを使用します。名前変更規則を更新するには、テーブルのセルの内容に上書きして入力します。列見出しのいずれかをダブルクリックすると、その列を基準に規則をソートすることができます。規則を指定するには、次の情報を指定します。

順序 規則を適用する順序を指定します。

変更するストリング

VAGen 名の中の変更したい文字を指定します。

変更後のストリング

変更後の EGL 名で使用する文字を指定します。

ストリング・コンテキスト

名前変更を行う際にマイグレーション・ツールが VAGen 名の中で変更するストリングを検索する場所を指定します。以下の値が有効です。

前 変更するストリングがプロジェクト名、パッケージ名、またはバージョン名の先頭にある場合に規則を適用します。

後 変更するストリングがプロジェクト名、パッケージ名、またはバージョン名の最後にある場合に規則を適用します。

任意 変更するストリングがプロジェクト名、パッケージ名、またはバージョン名内のどの位置にある場合でも規則を適用します。

トークン

変更するストリングがプロジェクト名、パッケージ名、またはバージョン名と完全に一致する場合にのみ規則を適用します。

マッピング・コンテキスト

マイグレーション・ツールがプロジェクト名、パッケージ名、またはバージョン名のうちのどれに名前変更規則を適用するのかを指定します。以下の値が有効です。

プロジェクト

名前変更規則を VAGen プロジェクト名だけに適用します。

パッケージ

名前変更規則を VAGen パッケージ名のみに適用します。

両方 名前変更規則を VAGen プロジェクト名と VAGen パッケージ名の両方に適用します。

バージョン

名前変更規則をすべてのプロジェクト名のバージョン名に適用します。ご使用のバージョン名が、ディレクトリー名やファイル名に使用できないセミコロン (;) などの特殊文字を含んでいる場合は、バージョン名変更規則を使用します。デフォルトの

MigPreferences.xml ファイルには、バージョン名が無効なディレクトリー名やファイル名にならないようにするために役立つ、いくつかのバージョン名変更規則が組み込まれています。マイグレーション・ツールは、名前変更済みのバージョンを使用して、マイグレーションのステージ 1 でマイグレーション・プラン・ファイル名を作成し、ステージ 3 でディレクトリー名を作成します。

「実行」ページ

実行オプション

「実行オプション」セクションでは、ステージ 1 マイグレーション・ツールが実行する処理を指定することができます。

レポートの生成

EGL プロジェクト、パッケージ、およびファイルの構造内で各パーツが配置される場所を示すマイグレーション・レポートを作成するように指定します。このレポートは、「共通パーツ」および「未使用パーツ」のファイル名、「スパン・マップ」接尾部、プロジェクトとパッケージの「共通 ID」、および「名前変更規則」に対して指定した設定の結果を検討するのに役立ちます。「レポートの生成」を選択すると、「検証」セクションで「レポート・ファイル名」に指定したドライブ、ディレクトリー、およびファイルに、マイグレーション・ツールがレポートを作成します。

データベースの更新

ステージ 1 マイグレーション・ツールがマイグレーション・プラン情報 (パーツの外部ソース形式など) をマイグレーション・データベースに格納するように指定します。

ステージ 1 マイグレーション・ツールは、以下の例のようにいくつかのステップに分けて実行することができます。

- ステップ 1 -- 「レポートの生成」と「データベースの更新」を両方ともクリアする。これにより、作成されたマイグレーション・プラン・ファイルを検討して、「リポジトリ・フィルター」が正しく設定されて目的のプロジェクト・バージョンが処理されていることを確認できます。選択されたプロジェクト・バージョンが適切でない場合は、「リポジトリ・フィルター」を調整してこのステップを再度実行します。
- ステップ 2 -- 「データベースの更新」を選択します (「レポートの生成」は選択してもしなくてもかまいません)。

注:

- データベースの更新には多少時間を要する場合があります。このため、.pln ファイルを検討して、目的のプロジェクト・バージョンがマイグレーション・ツールの処理対象になるか確認することが最良策です。
- レポートを実行する前にデータベースを更新する必要があります。
- レポートの生成にも多少時間を要する場合があります。レポートを作成する代わりに、いくつかの単純な照会を実行して EGL ファイル名を確認することもできます。EGL ファイル名を生成する照会の例については、553 ページの『EGL ファイル名の検討』を参照してください。
- レポートを生成する場合、レポート・ファイルは上書きされます。前のレポート・ファイルを保管する必要がある場合は、レポート・ファイルを別のディレクトリに移動するか、新規レポート用のディレクトリを新しく指定する必要があります。レポート・ファイルは他のファイルにリンクしているため、レポート・ファイルの名前を変更するとそのリンクが失われ、ファイルが表示できなくなります。

データベース

「データベース」セクションでは、マイグレーション・データベースに関する詳細を以下のように指定することができます。

データベース・ドライバー

この値は、常に **app.DB2Driver** にする必要があります。

データベース名

この値は、以下の形式のいずれかにする必要があります。

- `jdbc:DB2:databaseName` (ローカル・データベースまたはローカルにカタログされたリモート・データベースを使用する場合)。

注: `databaseName` は、マイグレーション・ツールがマイグレーション・セット情報を書き込むマイグレーション・データベースの名前です。デフォルトでは、`databaseName` は **VGMIG** です。マイグレーション・データベースの作成時の **VGMIG** からデータベース名を変更した場合は、この設定に指定されているデータベース名を変更して、使用した名前と一致させる必要があります。

スキーマ

データベース・テーブル用の修飾子として使用される名前。デフォルトでは、スキーマ名は **MIGSCHEMA** です。マイグレーション・データベースの作成時の **MIGSCHEMA** からスキーマ名を変更した場合は、この設定に指定されているスキーマ名を変更して、使用した名前と一致させる必要があります。

ユーザー ID

マイグレーション・データベースへの接続に必要なユーザー ID。ユーザー ID を指定しない場合、マイグレーション・ツールはログイン・ユーザー ID を使用して接続を試行します。この試行に失敗すると、マイグレーション・ツールは情報を尋ねるダイアログ・ウィンドウを表示します。

パスワード

マイグレーション・データベースへの接続に必要なパスワード。パスワードを指定しない場合、マイグレーション・ツールはログオン・パスワードを使用して接続を試行します。この試行に失敗すると、マイグレーション・ツールは情報を尋ねるダイアログ・ウィンドウを表示します。

注: パスワードは設定ファイル内で暗号化されません。このことが問題になる場合は、設定ファイルにパスワードを入力せず、プロンプトが出されるまで待ってください。

サービス

「サービス」セクションでは、ステージ 1 の間に収集するロギングおよびデバッグ情報に関する詳細を指定することができます。以下の詳細を指定することができます。

トレース・レベル

ログ・ファイルとデバッグ・ファイルに書き込む情報のレベルを指定することができます。ドロップダウン・リストを使用して、次のいずれかの値を指定します。

致命的 致命的エラー・メッセージがログに記録されます。致命的メッセージが出された場合、マイグレーション・データベースは更新される可能性があります。マイグレーション・プラン・ファイル (.pln ファイル) は変更されず、.done ファイル拡張子は付きません。このため、.pln ファイルを再度処理することができます。

警告 致命的エラー・メッセージのほかに、警告メッセージがログに記録されます。

情報 警告メッセージおよび致命的エラー・メッセージのほかに、情報メッセージがログに記録されます。

デバッグ

情報メッセージ、警告メッセージ、および致命的エラー・メッセージのほかに、デバッグ情報がログに記録されます。このトレース・レベルの場合にのみ、マイグレーション・ツールはデバッグ・ファイルに情報を書き込みます。これはデフォルト値です。

トレース・レベルは、ログ・ファイルとデバッグ・ファイルのみに影響を及ぼします。メッセージはすべてコンソール・ウィンドウに書き込まれます。

ログ・ファイル名

ログ・ファイルのドライブ、ディレクトリー、およびファイル名を指定することができます。任意のファイル拡張子を付けてログ・ファイルを作成できますが、このファイルは .xml ファイルとして最適に表示されます。ログ・ファイル名を省略した場合、マイグレーション・ツールは、「**ログ・ファイル名**」フィールドで指定したドライブとディレクトリーにある、miglog.xml という名前のファイルにログ情報を書き込みます。ログ・ファイルのドライブとディレク

トリーを指定しない場合、マイグレーション・ツールはログ・ファイルを作業ディレクトリーに書き込みます。

デバッグ・ファイル名

IBM サポートで必要になる可能性のあるデバッグ・ファイルのドライブ、ディレクトリー、およびファイル名を指定することができます。任意のファイル拡張子を付けてデバッグ・ファイルを作成できますが、このファイルは .xml ファイルとして最適に表示されます。このファイルへの情報の書き込みは、「**トレース・レベル**」が「**デバッグ**」に設定されている場合にのみ行われます。デバッグ・ファイル名を省略し、トレース・レベルとして「**デバッグ**」を指定すると、マイグレーション・ツールは、「**デバッグ・ファイル名**」フィールドで指定したドライブとディレクトリーにあるファイル migdebug.xml にデバッグ・ファイルの情報を書き込みます。デバッグ・ファイルのドライブとディレクトリーを指定しない場合、マイグレーション・ツールはデバッグ・ファイルを作業ディレクトリーに書き込みます。

検証 「**検証**」セクションでは、検証レポートのドライブ、ディレクトリー、およびファイル名を指定することができます。このレポートは、「**実行オプション**」セクションで「**レポートの生成**」設定を選択した場合に作成されます。「**レポートの生成**」を選択した場合は、「**レポート・ファイル名**」を入力する必要があります。必ず .htm 拡張子を指定してください。ドライブとディレクトリーを指定しない場合、マイグレーション・ツールはレポート・ファイルを作業ディレクトリーに書き込みます。

MigPreferences.xml ファイルのサンプル

次に、MigPreferences.xml ファイルのサンプルを示します。

```
<preferences>
  <database>
    <driver>COM.ibm.db2.jdbc.app.DB2Driver</driver>
    <uri>jdbc:DB2:VGMIG</uri>
    <schema>MIGSCHEMA</schema>
    <userid></userid>
    <password></password>
  </database>
  <migrationSpec>
    <directory>d:\tempMig\MyMigSet</directory>
    <filename></filename>
  </migrationSpec>
  <repositoryFilters>
    <projectName>MyProject*</projectName>
    <versionName></versionName>
  </repositoryFilters>
  <service>
    <tracelevel>4</tracelevel>
    <debugfile>d:\tempMig\MyMigSet\Stage1\migdebug.xml</debugfile>
    <logfile>d:\tempMig\MyMigSet\Stage1\miglog.xml</logfile>
  </service>
  <eglMapping>
    <renameRule order = "1">
      <fromString> </fromString>
      <toString></toString>
      <stringContext>any</stringContext>
      <mappingContext>both</mappingContext>
    </renameRule>
    <renameRule order = "101">
```

```

    <fromString>Project</fromString>
    <toString></toString>
    <stringContext>any</stringContext>
    <mappingContext>project</mappingContext>
</renameRule>
<renameRule order = "301">
    <fromString>.pkg</fromString>
    <toString></toString>
    <stringContext>any</stringContext>
    <mappingContext>package</mappingContext>
</renameRule>
<renameRule order = "302">
    <fromString>.sql</fromString>
    <toString>sql</toString>
    <stringContext>any</stringContext>
    <mappingContext>package</mappingContext>
</renameRule>
<renameRule order = "501">
    <fromString>:</fromString>
    <toString>_</toString>
    <stringContext>any</stringContext>
    <mappingContext>version</mappingContext>
</renameRule>
<renameRule order = "503">
    <fromString>/</fromString>
    <toString>_</toString>
    <stringContext>any</stringContext>
    <mappingContext>version</mappingContext>
</renameRule>
<renameRule order = "504">
    <fromString>\</fromString>
    <toString>_</toString>
    <stringContext>any</stringContext>
    <mappingContext>version</mappingContext>
</renameRule>
<renameRule order = "505">
    <fromString>|</fromString>
    <toString>_</toString>
    <stringContext>any</stringContext>
    <mappingContext>version</mappingContext>
</renameRule>
<renameRule order = "506">
    <fromString>?</fromString>
    <toString>_</toString>
    <stringContext>any</stringContext>
    <mappingContext>version</mappingContext>
</renameRule>
<renameRule order = "507">
    <fromString>*</fromString>
    <toString>_</toString>
    <stringContext>any</stringContext>
    <mappingContext>version</mappingContext>
</renameRule>
<renameRule order = "508">
    <fromString>&lt;</fromString>
    <toString>_</toString>
    <stringContext>any</stringContext>
    <mappingContext>version</mappingContext>
</renameRule>
<renameRule order = "509">
    <fromString>&gt;</fromString>
    <toString>_</toString>
    <stringContext>any</stringContext>
    <mappingContext>version</mappingContext>
</renameRule>
<renameRule order = "510">
    <fromString>&quot;</fromString>

```

```

        <toString>_</toString>
        <stringContext>any</stringContext>
        <mappingContext>version</mappingContext>
    </renameRule>
    <renameRule order = "511">
        <fromString> </fromString>
        <toString>_</toString>
        <stringContext>any</stringContext>
        <mappingContext>version</mappingContext>
    </renameRule>
    <verification>
        <generateReport>true</generateReport>
        <reportName>d:\tempMig\MyMigSet\report\MyReport.htm</reportName>
    </verification>
    <dbUpdate>true</dbUpdate>
    <spanningMapsProjectSuffix>MapsProject</spanningMapsProjectSuffix>
    <spanningMapsPackageSuffix>mapspackage</spanningMapsPackageSuffix>
    <commonPartsFileName>CommonParts</commonPartsFileName>
    <unusedPartsFileName>UnusedParts</unusedPartsFileName>
    <commonParts>
        <commonProject>*Common*</commonProject>
        <commonPackage>*common*</commonPackage>
    </commonParts>
</eglMapping>
</preferences>

```

ステージ 1 ツールを実行する前に - ヒント

ステージ 1 マイグレーション・ツールを実行する前に、以下のタスクを実行する必要があります。

- ステージ 1 マイグレーション・ツールのカスタマイズ
- 文字セットの指定
- ステージ 1 マイグレーション・ツールのパフォーマンスを改善する手順の実行
- ワークスペースの保存

ステージ 1 マイグレーション・ツールのカスタマイズ

Java のステージ 1 マイグレーション・ツールには以下のカスタマイズが組み込まれており、ご使用の環境に適している場合はこれらを使用可能にできます。

- すべてのデータ項目パーツを、ItemsProject と呼ばれる新規 EGL プロジェクトに移動する。項目は、接尾部 .items を付けられたオリジナル・パッケージ名に対応するパッケージ (例えば、my.original.pkg.items) 内に配置されます。このカスタマイズは、ステージ 2 マイグレーション設定で「**共用データ項目をプリミティブ項目定義に変換**」を選択する場合に特に役に立ちます。ステージ 1 でこのカスタマイズを使用可能化すると、マイグレーション時にすべてのデータ項目パーツを単一のプロジェクトに移動して、そのプロジェクトを EGL ワークスペースから除去することができます。
- パーツ型 (オプションで、パーツ名) 別に共通パーツ・ファイルを分割する。このカスタマイズにより、共通パーツ・ファイルのサイズが削減され、また特定のパーツを見つけやすくなります。

組み込みカスタマイズの他に、Java のステージ 1 の際にプロジェクトおよびパッケージを統合する方法を説明したホワイト・ペーパーがあります。このホワイト・ペーパーの入手に関する情報については、19 ページの『参照』を参照してください。お使いの VAGen プロジェクトおよびパッケージ命名規則によっては、ホワイト・

ペーパー手法の方が 158 ページの『「名前変更」ページ』で説明されている「命名規則」設定よりも簡単に使用することができます。

ステージ 1 組み込みカスタマイズの使用可能化

カスタマイズを使用可能化するには、以下の手順を実行します。

1. 「IBM VisualAge Generator EGL Migration」プロジェクトを展開する。
com.ibm.vgj.mig.db パッケージを展開する。**MigrationSetUtility** クラスを展開する。
2. **customizedPlacementScriptSet()** メソッドを編集する。
3. すべてのデータ項目パーツを新規 EGL プロジェクトに移動するカスタマイズを使用可能化するには、以下のようになります。

- a. 次のように行を変更する。

```
boolean consolidate = false;
```

から

```
boolean consolidate = true;
```

デフォルトでは、ステージ 1 ツールにより EGL ファイル名が "Items" に設定され、データ項目名の最初の文字と連結されます。

- b. ファイル名に連結されたデータ項目名の文字数を変更する場合は、**customizedPlacementScriptSet()** メソッドの **suffixLength** の値を変更します。
 - c. パッケージの EGL プロジェクトの名前または接尾部を変更する場合は、**consolidateDataItemsQuery()** メソッドを変更します。
4. パーツ型およびパーツ名によって共通パーツ・ファイルを分割するカスタマイズを使用可能化するには、以下のようになります。

- a. 次のように行を変更する。

```
boolean redistribute = false;
```

から

```
boolean redistribute = true;
```

- b. 以下の行で、共通パーツ・ファイルの名前に追加するパーツ名の文字数を指定する。

```
int suffixLength = 1;  
suffixLength = 2;
```

デフォルトでは、**suffixLength** の値は、データ項目パーツ名の最初の文字とレコードおよび関数名の最初の 2 文字を使用するように設定されます。ただし、**suffixLength** の値は変更することができます。例えば、次のようになります。

- この数値を 0 に設定すると、共通パーツ・ファイル名は **CommonRecords.egl**、**CommonFunctions.egl**、および **CommonItems.egl** になります。
- この数値を数字 *n* に設定すると、パーツ名の最初の *n* 文字がファイル名の接尾部として使用されます。*n* を 1 に設定すると、名前は **CommonRecordsA**、**CommonRecordsB** などになります。

- c. ファイル名を別の名前にする場合は、**redistributeCommonRecords(int)**、**redistributeCommonFunctions(int)**、または **redistributeCommonItems(int)** メソッドを変更します。
5. 変更を保存し、クラスをバージョン設定してリリースする。変更の記録を保持するために、パッケージおよびプロジェクトをバージョン設定することもできます。

文字セット情報の指定

ステージ 1 マイグレーション・ツールでは、お使いの VAGen 各国語言語コードに基づいて文字セットを指定します。1 バイト言語ではこれは「iso-8859-1」になり、2 バイト言語では別の値が使用されます。別の文字セットを使用する必要がある場合は、以下の手順を実行してこの値を変更します。

1. 「**IBM VisualAge Generator EGL Migration**」プロジェクトを展開する。**com.ibm.vgj.mig** パッケージを展開する。**Preferences** クラスを展開する。
2. **determineXMLEncoding()** メソッドを編集する。
3. `xmlCharset` に指定された値を目的の文字セットに変更する。例えば、メソッドの最後の 4 行を以下のように変更します。

```
if (vgNLS.equals("PTB") )
    xmlCharset = "iso-8859-1";
return xmlCharset;
}
```

太字で示された行を追加する。

```
if (vgNLS.equals("PTB") )
    xmlCharset = "iso-8859-1";
xmlCharset = "my required character set";    // new line with value you need
return xmlCharset;
}
```

4. 変更を保存し、クラスをバージョン設定してリリースする。変更の記録を保持するために、パッケージおよびプロジェクトをバージョン設定することもできます。

パフォーマンスの向上

パフォーマンス測定の結果、新規ワークスペースから作業を開始すると、ステージ 1 マイグレーション・ツールのパフォーマンスが大幅に向上することが判明しました。一連のテストでは、新規ワークスペースから開始すると、新規ワークスペースを使用しない場合に比べてステージ 1 の時間が 25% から 30% 短縮されました。既存のワークスペースが 20 メガバイトより大きい場合は、新規ワークスペースから開始するとステージ 1 ツールのパフォーマンスが向上する可能性があります。

新規ワークスペースから作業を開始するには、以下の手順を実行します。

1. **VisualAge Generator** をシャットダウンする。
2. マイグレーションの完了後に使用するために既存のワークスペースのバックアップ・コピーを保持したい場合は、167 ページの『ワークスペースの保管』を参照してください。
3. 次に示す **VisualAge Generator** ダウンロード・サイトから、新規ワークスペースのコピー (ファイル名 `ide.icx`) をダウンロードする。

`ftp://ftp.software.ibm.com/ps/products/visualagegen/fixes/v4.5/FixPack5/windows`

4. features.sav および projects.sav ファイルを削除する。
5. VisualAge Generator を再始動する。
6. 必要な VisualAge Generator フィーチャーを追加する。
7. 「IBM VisualAge Generator EGL マイグレーション」フィーチャーを追加する。
8. VisualAge Generator をシャットダウンする。

ステージ 1 マイグレーション・ツールがマイグレーション対象のプロジェクトとバージョンの分析に費やす時間を短縮するために、マイグレーションしたいプロジェクト・バージョンのみを含むリポジトリの作成を検討してください。マイグレーション中に VisualAge Generator の継続保守を行う場合は、別個のマイグレーション・リポジトリを使用すると次のような利点もあります。

- マイグレーションするプロジェクト・バージョンのセットが安定します。これは、「バージョン深さ」設定を使用してマイグレーション対象を制御する場合に特に重要になります。
- 新規マイグレーション・リポジトリのバージョンを保守リポジトリと比較して、マイグレーションを必要とする他のプロジェクト・バージョンがまだあるかどうか判別できます。

特殊リポジトリを作成する場合は、ステージ 1 マイグレーションのパフォーマンスを高めるために、ローカル・リポジトリとして使用することを検討してください。

ワークスペースの保管

ステージ 1 マイグレーション・ツールは、ステージ 1 処理の開始時と終了時に、VAGen パーツを含むプロジェクトをワークスペースからすべて削除します。これにより、ワークスペース内でパーツの重複を防止でき、ステージ 1 の実行中に、マイグレーション・セット内のパーツのみが関連パーツ・リストの対象と見なされるようになります。保存するワークスペースがある場合は、ステージ 1 ツールを実行する前に以下の手順を行う必要があります。

1. VisualAge Generator をシャットダウンする。
2. 次のファイルのバックアップ・コピーを、`¥VisualAgeForJava-installation-directory¥ide¥program` に保管する。
 - features.sav
 - projects.sav
 - ide.icx
 - ide.ini (ステージ 1 の実行中に設定を変更しない場合は保管不要)
 - hpt.ini (ステージ 1 の実行中に設定を変更しない場合は保管不要)
3. VisualAge Generator を開始する。

ステージ 1 ツールの実行が完了したら、以下の手順を使用してワークスペースを復元します。

1. VisualAge Generator をシャットダウンする。
2. ステージ 1 ツールを実行する前にバックアップしたファイルを復元する。
3. VisualAge Generator を開始する。

ステージ 1 ツールの実行

設定の編集を完了した後、ステージ 1 マイグレーション・ツールを実行して、ソース・コードを Java リポジトリから抽出できます。このためには、次の手順で行います。

1. 「**IBM VisualAge Generator EGL マイグレーション**」プロジェクトにナビゲートする。
2. マイグレーション・プロジェクトを展開し、**com.ibm.vgj.mig** パッケージを展開する。
3. このパッケージ内で、「**VAGenToEGLMigration**」クラスを選択する。
4. 「**VAGenToEGLMigration**」クラスを右クリックしてから、「**プロパティ**」をクリックする。
5. 「**プログラム**」タブをクリックする。
6. 「プログラム」ページの「**コマンド行引数**」フィールドで以下のように指定して、使用する MigPreferences.xml ファイルを指示する。

表 61. VAGenToEGLMigration クラスの有効なコマンド行オプション

オプション	オプションの意味
-h	有効なオプションを示すヘルプ情報を表示します。
-p <i>filename</i>	<i>filename</i> は設定ファイルの名前です。ファイル名は、ドライブとディレクトリーを含めて完全に修飾する必要があります。
-o	存在する場合はマイグレーション・ファイルを上書きし、再作成します。

7. ステージ 1 ツールを初めて実行する場合は、以下の手順を実行します。
 - a. 同じ「プロパティ」ウィンドウで、「**クラスパス**」タブをクリックする。
 - b. 「クラスパス」ページで、「**追加のディレクトリー・パス**」チェック・ボックスを選択し、追加のディレクトリーの「**編集**」ボタンをクリックする。
 - c. 「**Jar/Zip の追加**」をクリックする。
 - d. 「ファイル選択」ウィンドウで、db2java.zip ファイルにナビゲートしてこのファイルを選択する。
 - DB2 をインストールしたときにデフォルトのインストール・ディレクトリーを使用していた場合、ファイルは %SQLLIB%java ディレクトリーにあります。

db2java.zip ファイルを選択した後、ファイル名が「追加のディレクトリー (Extra directories)」ウィンドウに表示されます。「追加のディレクトリー (Extra Directories)」ウィンドウの「**OK**」をクリックします。
 - e. 「クラスパス」ページで、「**計算 (Compute Now)**」をクリックしてから、プロンプトの「**はい**」をクリックする。
8. 「**OK**」をクリックして、プロパティを保存する。
9. 「**VAGenToEGLMigration**」を右クリックしてから、「**実行**」->「**メインを実行**」をクリックする。(または、ツールバーの走る人のアイコンをクリックすることもできます。) ステージ 1 マイグレーション・ツールが始動し、コンソール

ル・ウィンドウが開いて進行状況とエラー・メッセージが報告されます。マイグレーション設定の中で指定したログ・ファイルにも、マイグレーション・ツールのメッセージが書き込まれます。

パーツの数およびパーツ間の関連の複雑度によっては、ステージ 1 マイグレーション・ツールのいくつかのステップは長い時間を要する場合があります。「コンソール」ウィンドウの以下のステップは、顕著なアクティビティーがなくても特に時間を消費 (1 時間以上) します。

- partPlacementQuery 3 の実行
- partPlacementQuery 4 の実行
- setPartFilePathQuery 3 の実行
- マイグレーション・レポート生成の開始

「データベースの更新」設定を選択している場合は、ステージ 1 マイグレーション・ツールの終了時にマイグレーション・プラン情報 (外部ソース形式の VAGen コードなど) がマイグレーション・データベースに保管されます。レポートとステージ 1 メッセージを検討した後、必要に応じて VisualAge Generator 内でコードを変更し、ステージ 1 を再度実行できます。「データベースの更新」を再度選択した際に、同じ名前のマイグレーション・セットがデータベース内に既に存在している場合、ステージ 1 マイグレーション・ツールはマイグレーション・セットに関する以前の情報をデータベースから自動的に削除し、マイグレーション・セットに関する新しい情報を追加します。このため、マイグレーション・セットをデータベースからクリーンアップする必要はありません。ただし、549 ページの『ステージ 1 のマイグレーション・データベースのリセット』で説明されている手法を使用するとより速く実行することができます。

ステージ 1 の結果に問題がなく、最終の外部ソース形式コードがマイグレーション・データベースに格納された後、マイグレーションのステージ 2 を実行できます。ステージ 2 マイグレーション・ツールを実行するには、EGL 開発環境を使用します。マイグレーション・プロセスの継続については、207 ページの『第 6 章 ステージ 2 - EGL 構文への変換』を参照してください。

マイグレーション・プランと上位 PLP プロジェクト

マイグレーション・プラン・ファイルは単純な XML ファイルで、1 つ以上のマイグレーション・セットの名前を指定し、それぞれのマイグレーション・セットごとに、マイグレーション・セットを構成するプロジェクト名とバージョンのリストを指定します。ステージ 1 マイグレーション・ツールは、プロジェクトおよびバージョン名に関する「リポジトリ・フィルター」の設定に基づいて、マイグレーション・プラン・ファイルを自動的に作成するように設計されています。ステージ 1 ツールは、プロジェクト・バージョンが上位 PLP プロジェクトかどうかを判別するために、これらのフィルターを使用してプロジェクト・バージョンが検討対象かどうかを判別します。ステージ 1 ツールは、それぞれの上位 PLP プロジェクト・バージョンをマイグレーション・セットの基礎として使用します。

VAGen ソース・コードの生成時に PLP プロジェクトを使用した場合は、これらと同じ PLP プロジェクトをマイグレーションに使用します。これは、これらの PLP

プロジェクトが生成時に一緒に使用されるパーツのグループを指定しているからで、したがってこのプロジェクトは一連のプログラムの関連パーツをすべて含んでいます。

PLP プロジェクトを現在使用していない場合は、以下の手法のいずれかを使用することができます。

- VisualAge Generator によって提供されるツールを使用して、マイグレーション用の上位 PLP プロジェクトを作成します。上位 PLP プロジェクトでは、グループとしてマイグレーションするプロジェクト・バージョンのリストを指定する必要があります。VisualAge Generator によって提供されるツールを使用すると、プロジェクトおよびバージョン名のつづりが正確になり、大/小文字も訂正されます。上位 PLP プロジェクトを作成すると、ステージ 1 マイグレーション・ツールを使用して、マイグレーション・プランを作成することができます。
- 上位 PLP プロジェクトを作成しない場合は、次のいずれかの手法を使用して、マイグレーション・プラン・ファイルをユーザーが独自に作成できます。
 - Java プロジェクトのバージョンに関連して、生成に必要なものを指定する情報がデータベースや他のシステム内にある場合は、データベースからマイグレーション・プラン・ファイル (複数可) を自動的に作成するツールをユーザーが開発できます。
 - マイグレーション・プラン・ファイル (複数可) を手作業で作成できます。

上位 PLP プロジェクトの作成

注: プロジェクト名またはバージョン名に DBCS 文字が含まれている場合、VisualAge Generator は PLP をサポートしません。プロジェクト名またはバージョン名に DBCS 文字が含まれている場合に、PLP を使用せずにマイグレーション・プラン・ファイルを作成する方法については、172 ページの『マイグレーション・プラン・ファイルの手動作成』を参照してください。

マイグレーションに使用する上位 PLP プロジェクトを作成するには、VisualAge Generator 内で以下の手順を実行します。

1. 「ワークベンチ」ウィンドウの「プロジェクト」タブをクリックする。
2. プロジェクト・リスト・パーツを格納する Java プロジェクトを新規に作成する。プロジェクトには、必ず既存のプロジェクトと異なる名前を付けるようにしてください。例えば、MySubsystem1 という名前のプロジェクトを作成します。
3. 新規プロジェクトを右クリックし、「管理」->「VAGen 必須プロジェクトの構成」をクリックします。
4. 「VAGen 必須プロジェクトの構成 (Configure VAGen Required Projects)」ウィンドウで、マイグレーション・セットに組み込む各プロジェクトを選択する。それぞれのプロジェクトごとに、マイグレーション・セットに組み込む特定のバージョンを選択できます。代わりに、「最新エディション (Most recent edition)」を選択することもできます。この場合、マイグレーション・ツールは、マイグレーション中にこのプロジェクトを使用するときは常に、現時点でリストの先頭にあるバージョンを自動的に組み込みます。
5. マイグレーション・セットに必要なプロジェクト・バージョンをすべて選択した後、「OK」をクリックする。

6. 上位 PLP プロジェクトのバージョンとリリースを指定する (例: MySubsystem1)。
7. その PLP プロジェクトによってマイグレーション・セットに必要なプロジェクト・バージョンが正しくロードされることを、以下の手順で確認する。
 - a. 上位 PLP プロジェクトと他のすべての VAGen プロジェクトをワークスペースから削除する。
 - b. 「選択」->「追加」->「プロジェクト」を選択する。
 - c. 「プロジェクトの追加」ウィンドウで、以下の手順を実行する。
 - 1) 「リポジトリからプロジェクトを追加」をクリックする。
 - 2) 直前に作成した上位 PLP プロジェクト、および作成したバージョンを選択する。
 - 3) 「VAGen 必須プロジェクトの追加 (Add VAGen required projects)」も選択する。
 - 4) 「終了」をクリックします。
 - d. 上位 PLP プロジェクトと、そのプロジェクトの指定するプロジェクト・バージョンすべてがワークスペースに追加されます。
 - e. VAGen パーツ・ブラウザーで、「ツール」->「重複パーツの表示」をクリックする。リストにパーツがあってはなりません。パーツがある場合は、重複がなくなるように上位 PLP プロジェクトを変更する必要があります。マイグレーション・セット内に重複するパーツが存在する場合、ステージ 1 マイグレーション・ツールは停止します。
 - f. また、プログラムとテーブルの検証を実行して、それらが VisualAge Generator 内で有効であること、および欠落しているパーツがないことを確認することもできます。

PLP プロジェクトはチェーニングできます。例えば、すべての共通プロジェクトのプロジェクト・バージョンをリストする PLP プロジェクトを作成することができます。次に、それぞれのサブシステムごとに、サブシステム固有のプロジェクト・バージョンすべてを含むサブシステム用の上位 PLP プロジェクト、および共通プロジェクト・バージョンすべてを指定する PLP プロジェクトを作成します。この方法を使用すると、サブシステムごとに上位 PLP プロジェクトでそれぞれの共通プロジェクト・バージョンをリストする必要がなくなります。

ステージ 1 マイグレーション・ツールの実行準備完了後、以下の手順を実行します。

1. ステージ 1 設定を行う際に、「プランの作成」ページの「リポジトリ・フィルター」セクションで「プロジェクト」リストを作成して、リスト内のフィルターが作成した上位 PLP プロジェクトに一致するようにします。
2. 使用する設定ファイルをステージ 1 ツールに指定するときは、-o オプションも指定します。-o オプションを指定すると、ステージ 1 マイグレーション・ツールは上位 PLP プロジェクトに基づいてマイグレーション・プラン・ファイルを作成し、既存のマイグレーション・プラン・ファイルを上書きします。

マイグレーション・プラン・ファイルの手動作成

VisualAge Generator での生成時にワークスペースに追加するプロジェクト・バージョンを決定する外部制御手段がすでに存在する場合は、マイグレーション・プラン・ファイルを手動で作成するか、外部情報からマイグレーション・プラン・ファイルを自動的に作成するツールを開発できます。マイグレーション・プラン・ファイルの拡張子は .pln にする必要があり、書式は以下のとおりです。

```
<migrationDefinition>
  <migrationSet name="migrationSet1" version="migrationSet1Version1"
    vgName="migrationSet1" vgVersion="migrationSet1Version1">
    <project name="projectName1" version="projectName1Version1"></project>
    <project name="projectName2" version="projectName2Version1"></project>
    .
    .
    .
    <project name="projectNameN" version="projectNameNVersion1"></project>
  </migrationSet>
  <migrationSet name="migrationSet2" version="1.1"
    vgName="migrationSet2" vgVersion="1.1">
    <project name="projectNameA" version="projectNameAVersion1"></project>
    <project name="projectNameB" version="projectNameBVersion1"></project>
    .
    .
    .
    <project name="projectNameZ" version="projectNameZVersion1"></project>
  </migrationSet>
</migrationDefinition>
```

以下の説明は上の例に適用されます。

- *migrationSet1* は、同時にマイグレーションする必要があるプロジェクトのグループを参照するために使用することができる名前です。マイグレーション・セット名はマイグレーション・データベースに保管され、以降のマイグレーション・ステージで以下のようにして使用されます。
 - ステージ 1 マイグレーションでは、マップ・グループ内のマップが複数のプロジェクトにわたる場合、接尾部と連結されたマイグレーション・セット名を使用して、マップ・グループとそのすべてのマップを格納する新規 EGL プロジェクトの名前が作成されます。名前変更規則を変更する場合にも、マイグレーション・データベースから情報を除去するためにマイグレーション・セット名が使用されます。
 - ステージ 2 マイグレーションの場合、マイグレーション・セット名は、マイグレーション・データベース内で EGL に変換する対象のプロジェクト・グループを指定します。
 - ステージ 3 の場合、マイグレーション・セット名は、ワークスペースまたは一時ディレクトリーに EGL プロジェクト、パッケージ、およびファイルを作成するために使用する、マイグレーション・データベース内のプロジェクト・グループを指定します。ステージ 3 の出力を一時ディレクトリーに保管する場合、マイグレーション・セット名とマイグレーション・セット・バージョンは、上位ディレクトリー名の作成にも使用されます。

マイグレーション・セット名は、マイグレーション時にプロジェクト・グループを識別する手段としてのみ使用されます。マップが VisualAge Generator 内で複数のプロジェクトにわたっている場合を除き、マイグレーション・セット名はマイグレーション後には使用されません。

- *projectName1*、*projectName2*、...、*projectNameN* は、グループとしてマイグレーションするプロジェクトです。*projectName* は、マイグレーション・セット内で一度だけリストする必要があります。マイグレーション・ツールは、同じマイグレーション・セットの下にリストされているプロジェクト・バージョンすべてをワークスペースにロードし、グループとして処理します。
- *projectName1Version1*、*projectName2Version1*、...、*projectNameNVersion1* は、これらのプロジェクトそれぞれに対応するバージョンです。1 つのマイグレーション・セット内で、それぞれのプロジェクトごとにバージョンをただ 1 つ指定できます。
- 指定するプロジェクト名およびバージョン名は、リポジトリ内のプロジェクト名およびバージョン名と完全に一致する必要があります。名前には大/小文字の区別があります。この情報を使用して、ワークスペースにプロジェクト・バージョンが追加され、ステージ 1 マイグレーション・レポートの作成、およびデータベースのロードのためにパーツを分析できるようになります。

ただ 1 つのマイグレーション・セットを含むマイグレーション・プラン・ファイルを作成できます。また、それぞれのマイグレーション・セットごとに <migrationSet> タグと </migrationSet> タグの間の情報を繰り返すことによって、複数のマイグレーション・セットを含むマイグレーション・プラン・ファイルを作成することもできます。

マイグレーション・セットが有効であることを確認するには、以下の手順を実行してマイグレーション・プラン内の各マイグレーション・セットをテストします。

1. ワークスペースから VAGen プロジェクトをすべて削除する。
2. マイグレーション・セットで指定したプロジェクト・バージョンをワークスペースに手動で追加する。マイグレーション・セットで指定した各プロジェクトのバージョンを確実に追加してください。指定したプロジェクト・バージョンまたはパッケージ・バージョンがリポジトリ内で使用不可である場合、ステージ 1 マイグレーション・ツールは停止します。
3. VAGen パーツ・ブラウザーで、「ツール」->「重複パーツの表示」をクリックする。リストにパーツがあってはなりません。パーツがある場合は、重複しないようにマイグレーション・セット定義を変更する必要があります。マイグレーション・セット内に重複するパーツが存在する場合、ステージ 1 マイグレーション・ツールは停止します。
4. また、プログラムとテーブルの検証を実行して、それらが VisualAge Generator 内で有効であること、および欠落しているパーツがないことを確認することもできます。
5. ステージ 1 マイグレーション・ツールでは、プロジェクトがバージョン設定されている必要があります。管理照会を実行して、プロジェクトまたはパッケージのオープン・エディションまたはスクラッチ・エディションが存在するかどうかを判別します。管理照会を実行するには、以下の手順を実行します。
 - a. 「ワークベンチ」ウィンドウで、「ワークスペース」->「管理照会」をクリックする。
 - b. 「管理照会」ウィンドウで、以下の手順を実行する。

- 1) 「プログラム要素 (Program element)」セクションで、「プロジェクト」および「パッケージ」を選択する。「タイプ」がクリアされていることを確認する。
- 2) 「状況」セクションで、「スクラッチ」をクリックする。
- 3) 「スコープ」セクションで、「ワークスペース」をクリックする。
- 4) 「所有者」セクションで、「任意のユーザー」をクリックする。
- 5) 「照会の開始 (Start Query)」ボタン (ツールバーの最後のボタン) をクリックします。スクラッチ・エディションが存在してはなりません。スクラッチ・エディションが存在する場合は、プロジェクトおよびパッケージのオープン・エディションを作成します。所有者である場合は、「状況」ペインからオープン・エディションを作成することができます。
- 6) 「プログラム要素 (Program element)」セクションを変更して、「プロジェクト」、「パッケージ」、および「タイプ」を選択します。
- 7) 「状況」セクションで、「オープン・エディション」をクリックする。
- 8) 「照会の開始 (Start Query)」ボタンをクリックします。
- 9) **IBM VisualAge Generator EGL マイグレーション**・プロジェクト用以外のオープン・エディションが存在してはなりません。オープン・エディションが存在する場合は、プロジェクト、パッケージ、またはタイプをバージョン設定します。所有者である場合は、「状況」ペインからそれらをバージョン設定することができます。

ステージ 1 マイグレーション・ツールの実行準備完了後、以下の手順を実行します。

1. ステージ 1 設定を行う際に、「プランの作成」ページで「**プラン・ディレクトリー名**」をマイグレーション・プラン・ファイルの保管先のドライブとディレクトリーに設定します。作成したマイグレーション・プランを **1** つだけ使用してステージ 1 マイグレーション・ツールを実行する場合は、「プラン・ファイル名」を指定します。指定した**プラン・ディレクトリー**にあるすべてのマイグレーション・プラン・ファイルを使用してステージ 1 マイグレーション・ツールを実行する場合は、「**プラン・ファイル名 (Plan file name)**」をプランクのままにします。
2. 使用する設定ファイルをステージ 1 ツールに指定するとき、**-o** オプションは必ず省略してください。**-o** オプションを省略すると、ステージ 1 ツールは既存のマイグレーション・プラン・ファイルを使用します。つまり、ツールはマイグレーション・プラン・ファイルを新規に作成しません。

第 3 部 VisualAge Generator 4.5 on Smalltalk から EGL へのマイグレーション

第 5 章 ステージ 1 - Smalltalk からの抽出

情報を VisualAge Generator から抽出する前に、VisualAge Smalltalk 上で稼働するステージ 1 マイグレーション・ツールをインストールする必要があります。また、VisualAge Generator 4.5 (VAGen 4.5) から EGL にマイグレーションするデータの保管に使用される、DB2 マイグレーション・データベースも作成する必要があります。

VisualAge Smalltalk へのステージ 1 マイグレーション・ツールのインストール

VisualAge Generator から EGL へのステージ 1 マイグレーション・ツールは、VAGenMigST.exe という名前の自己解凍ファイルとして提供されます。このファイルをインストールするには、以下の手順を実行します。

1. フィックスパック 4 およびフィックスパック 5 を適用した VisualAge Generator 4.5 にアップグレードする。また、ユーザー個々の状況に応じてさらに必要になる可能性がある VisualAge Generator APAR については、545 ページの『付録 F. VisualAge Generator に必要な APAR』を参照してください。

注: Smalltalk の VAGen 用のフィックスパック 5 の初期のバージョンは累積フィックスパックではありませんでした。README ファイルを確認して、使用しているフィックスパック 5 のバージョンが累積フィックスパックであることを確認してください。必要な場合は、再度フィックスパックをダウンロードするか、または IBM サポートに連絡してください。

2. 使用しているシステム上で、VisualAge Smalltalk がインストールされている場所を判別する。
3. VisualAge Smalltalk をシャットダウンする。
4. 自己解凍型の VAGenMigST.exe ファイルを実行する。このファイルは、以下のディレクトリにあります。

`\installationDirectory\bin`

注: 現在ご使用の製品をインストールする前に、前のバージョンの開発者製品をインストールし、保持していた場合、対象のインストール・ディレクトリは、以前のインストール時に使用されたディレクトリである可能性があります。

5. GUI プロンプトが表示されたら、VisualAge Smalltalk がインストールされているドライブとディレクトリにナビゲートする。その後で、「Unzip」をクリックする。

自己解凍型の実行可能ファイルが実行されると、次のファイルが VisualAge Smalltalk インストール・ディレクトリに抽出されます。

- feature¥vgMigST.ctl
- image¥Messages.properties
- image¥MigPreferences.xml

- image¥VGMigReserved.txt
- import¥vgMigST.dat
- checkStage1.bat
- checkStage1.sql
- createdatabase.sql
- createindex.sql
- createtables.sql
- deletemigsets.bat
- runStats.bat
- SetupDatabase.bat
- SetupIndex.bat
- SetupTables.bat

マイグレーション・フィーチャーのロード

ステージ 1 マイグレーション・ツールを使用するには、VAGen EGL マイグレーション・フィーチャーをロードする必要があります。このためには、次の手順で行います。

1. VisualAge Generator on Smalltalk を開始する。
2. 以下の手順を実行して、VAGen EGL マイグレーション・フィーチャーをロードする。
 - a. 「システム・トランスクリプト」で、「ツール」->「フィーチャーのロード/アンロード」をクリックする。
 - b. 「必須選択 (Selection Required)」ウィンドウで、以下のステップを実行する。
 - 1) 「他のフィーチャーを表示 (Show other features)」チェック・ボックスが選択されていることを確認する。
 - 2) 「使用可能なフィーチャー」ペインで、「その他: VAGen EGL マイグレーション - *versionName* (Other: VAGen EGL Migration - *versionName*)」を選択する。
 - 3) 「>>」ボタンをクリックして、「その他: VAGen EGL マイグレーション - *versionName*」を「ロードされるフィーチャー (Loaded features)」ペインに移動する。
 - 4) 「OK」をクリックします。VAGen EGL マイグレーション・フィーチャーがインポートされ、イメージにロードされます。
3. システム・トランスクリプトに、VAGen EGL マイグレーション・フィーチャーが正常にロードされたことを示すメッセージが表示されます。また、ツールバーに「EGL マイグレーション・ツール (EGL Migration Tools)」が表示されます。VisualAge Organizer 内では、「アプリケーション (Applications)」ペインに HptEglMigrationGuiApp が表示されます。
4. VAGen EGL マイグレーション・フィーチャーのロード後、イメージを保存するためのプロンプトが出されます。フィーチャーを再度ロードしなくて済むように、「はい」を選択します。

注:

1. フィーチャーのロードに問題が生じた場合は、`abt.ini` ファイルを検査してください (`VisualAge-Smalltalk-installation-directory¥image` ディレクトリーに保管されています)。 `abt.ini` ファイルの `[EmLibraryInterface]` 見出しの下に、次のフィールドが入力されていることを確認してください。
 - `ServerAddress=myserver.somecompany.somewhere.com`。この値は、EMSRV を実行する社内のサーバーを指示する必要があります。ローカル・ライブラリーを使用している場合は、`ServerAddress=127.0.0.1` を設定します。
 - `DefaultName=path-to-mgr50.dat¥mgr50.dat`。この値は、Smalltalk ライブラリーの名前であることが必要です。
2. リモート・ライブラリー・マネージャーを使用している場合は、リモート・サーバー上で自己解凍実行可能ファイルを実行して、`abt.ini` ファイルで `installationPath` および `importDirectory` に指定された [フィーチャー・インストール] 見出しおよび値に基づいて解凍します。

マイグレーション・データベースの作成

マイグレーション・データベースの作成については、547 ページの『DB2 マイグレーション・データベースの作成』を参照してください。自己解凍型の `VAGenMigST.exe` ファイルを実行したときに `VisualAge Smalltalk インストール・ディレクトリー` に置かれた、`SetupDatabase.bat` ファイルと `SetupTables.bat` ファイルを使用する必要があります。

ステージ 1 設定の実行

ステージ 1 マイグレーション・ツールを `VisualAge Smalltalk` にインストールするときに、インストール・プロセスによってサンプル設定ファイル `MigPreferences.xml` がディレクトリー `VisualAge-Smalltalk-installation-directory¥image` に作成されます。設定を変更する前に、バックアップのために `MigPreferences.xml` ファイルのコピーを作成しておく必要があります。 `MigPreferences.xml` ファイルを `VisualAge for Smalltalk` インストール・ディレクトリーの外側にあるディレクトリーにコピーして、そのコピーに対して変更を加えることもできます。これによって、新規バージョンのマイグレーション・ツールをインストールした場合に変更内容を誤って上書きしてしまうことを防止できます。

テキスト・エディター、またはステージ 1 マイグレーション・ツールに付属の GUI エディターを使用して、`MigPreferences.xml` ファイルを編集できます。ステージ 1 GUI エディターは、以下に示す 2 つの方法のいずれかで開始することができます。

- 「システム・トランスクリプト」で、「**EGL マイグレーション・ツール**」->「**設定エディター**」をクリックする。EGL マイグレーション設定エディターが表示されます。設定エディターは、デフォルトで最後に変更した設定ファイル（また、設定をまだ一度も変更していない場合は、ステージ 1 ツールに付属の `MigPreferences.xml` ファイル）を開きます。別の設定ファイルを指定する必要がある場合は、「開く」をクリックします。
- 「システム・トランスクリプト」で、「**EGL マイグレーション・ツール**」->「**マイグレーション・ドライバー**」をクリックする。「**マイグレーション・ファイル**

設定 (Migration File Preference)」セクションで、設定ファイルのファイル名を指定して、「編集」をクリックする。EGL マイグレーション設定エディターが表示されます。この手法の利点は、設定ファイルの変更を完了した後、ステージ 1 マイグレーション・ツールを実行できる状態になることです。

どちらの手法を使用する場合も、EGL マイグレーション設定エディターによって、ステージ 1 マイグレーション・ツールを制御する設定を行うことができます。設定の編集が完了したら、「保存」または「名前を付けて保存」(新しいファイル名を指定) をクリックしてエディターを閉じます。

注:

- 現在、構成マップを使用していない場合には、199 ページの『マイグレーション・プランと上位構成マップ』を参照してください。
- ドライブとディレクトリーを必要とする設定の場合は、次に示す 2 つの方法のどちらかで情報を指定できます。
 - 絶対パス。例: d:\tempMig¥MySystem¥
 - 相対パス。この場合、パスは作業ディレクトリーに対する相対パスです。例えば、次のようになります。
 - .¥tempMig¥MySystem は、絶対パス *VisualAge-Smalltalk-installation-directory¥image¥tempMig¥MySystem* に相当します。
 - ..¥tempMig¥MySystem は、絶対パス *VisualAge-Smalltalk-installation-directory¥tempMig¥MySystem* に相当します。

以降では、変更できる設定について、GUI 内でその設定が表示されるページ別に説明します。

- 「プランの作成」ページ
- 「マッピング」ページ
- 「名前変更」ページ
- 「実行」ページ

「プランの作成」ページ

「プランの作成 (Build Plans)」ページでは、マイグレーション・プランを配置する場所に関する情報を指定できます。また「プランの作成 (Build Plans)」ページでは、マイグレーション対象にするライブラリー内の構成マップとバージョンを指示することもできます。「プランの作成 (Build Plans)」ページは、次のセクションで編成されています。

マイグレーション・プラン仕様

ステージ 1 マイグレーション・ツールがマイグレーション・プラン・ファイル (複数の場合もあります) の読み取りまたは書き込みを行う場所を指定します。

プラン・ディレクトリー

マイグレーション・プラン・ファイル (複数の場合もあります) を配置するターゲット・ディレクトリー。

プラン・ファイル名

マイグレーション・データベースをロードするために作成して使用

する、マイグレーション・プラン・ファイルのファイル名 (オプション)。「**プラン・ファイル名 (Plan File Name)**」をクリックして、プラン・ディレクトリーにある既存のプラン・ファイルを表示できます。プラン・ファイル内の詳細を表示する必要がある場合は、「**プランの表示 (View Plans)**」をクリックしてプラン・ファイルを展開し、マイグレーション・セットを表示します。

- 「**プラン・ファイル名**」を指定しない場合、マイグレーション・ツールは指定した「**プラン・ディレクトリー**」内のすべての .pln ファイルを削除してから、新規プラン・ファイルを作成します。マイグレーション・ツールは、それぞれのマイグレーション・セットごとにプラン・ファイルを 1 つずつ作成します。この場合、マイグレーション・プラン・ファイル名の形式は `migrationSetName_version.pln` です。
- 「**プラン・ファイル名**」を指定した場合、マイグレーション・ツールは指定した「**プラン・ディレクトリー**」から指定した .pln ファイルのみを削除してから、指定した「**プラン・ファイル名**」を使用して新規 .pln ファイルを作成します。この場合は、単一のプラン・ファイルにすべてのマイグレーション・セットがリストされます。

リポジトリ・フィルター

この情報を使用すると、ステージ 1 マイグレーション・ツールによって処理される Smalltalk ライブラリーの構成マップとバージョンを制御することができます。構成マップとバージョンを制限することにより、ステージ 1 マイグレーション・ツールのパフォーマンスを大幅に向上できます。複数のフィルターを指定できます。ステージ 1 マイグレーション・ツールは、以下のようにして「**構成マップ**」フィルターおよび「**バージョン名**」または「**バージョン深さ**」フィルターを使用します。

- マイグレーション・ツールは、ライブラリー内の各構成マップ名を「**構成マップ**」フィルターと突き合わせます。
 - 構成マップ名が「**構成マップ**」フィルターのいずれにも一致しない場合、その構成マップは以後の処理対象になりません。
 - 構成マップ名が「**構成マップ**」フィルターの少なくとも 1 つと一致する場合、構成マップのバージョンは以下のようにして処理されます。
 - 「**バージョン名**」フィルターを指定した場合は、構成マップの各バージョン名と「**バージョン名**」フィルターのリストとの突き合わせが行われます。バージョン名が「**バージョン名**」フィルターのいずれかに一致する場合、そのバージョンは以後の処理対象になります。
 - 「**バージョン深さ**」フィルターを指定し、「**バージョン名**」フィルターを指定しなかった場合は、構成マップの最新のいくつかのバージョン（「**バージョン深さ**」フィルターで指定された数まで）が以後の処理対象になります。デフォルトの「**バージョン深さ**」フィルターは 1 です。

注: 「バージョン深さ」と「バージョン名」は相互に排他的です。

デフォルトでは、「バージョン深さ」フィルターが

MigPreferences.xml ファイルに組み込まれています。

- 構成マップ名とバージョン名により以後の処理対象の構成マップ・バージョンが指定された場合、ステージ 1 マイグレーション・ツールは、以下のようにしてプロジェクト・バージョンを処理します。
 - 構成マップ・バージョンが上位構成マップならば、マイグレーション・ツールはその構成マップ・バージョンをマイグレーション・セットの作成の基礎として使用します。バージョン名がバージョン・フィルターとマッチングすることを前提とすると、上位構成マップのそれぞれのバージョンごとに、異なるマイグレーション・セットが作成されます。
 - 構成マップ・バージョンが上位構成マップでなければ、構成マップ・バージョンは以後の処理対象になりません。その構成マップ・バージョンは、他のマイグレーション・セットにも組み込まれている可能性があります。この構成マップ・バージョン用にマイグレーション・セットが特別に作成されることはありません。

以下のようにして「リポジトリ・フィルター」情報を指定します。

構成マップ

マイグレーション・ツールは、ライブラリー内の構成マップ名と、ユーザーが指定した「構成マップ」フィルターとを突き合わせます。「構成マップ」フィルターは複数個指定することができます。フィルターを追加、変更、または除去するには、フィルターを右クリックして、ポップアップ・メニューのオプションを使用します。フィルターに大/小文字の区別はありません。以下のようにして、フィルター内でワイルドカードを使用することができます。

- 構成マップ・フィルター `*xyz*` は、ライブラリー内にある、ストリング「xyz」を名前のどこかに含む構成マップ名すべてとマッチングします。
- 構成マップ・フィルター `xyz*` は、ライブラリー内にある、「xyz」から始まる構成マップ名すべてとマッチングします。
- 構成マップ・フィルター `*xyz` は、ライブラリー内にある、「xyz」で終わる構成マップ名すべてとマッチングします。

バージョン名

構成マップ名が「構成マップ」フィルターと一致する場合、マイグレーション・ツールは「バージョン名」フィルターを使用して、マイグレーション対象になる構成マップ・バージョン (存在する場合) を判別します。「バージョン名」フィルターは複数個指定することができます。フィルターを追加、変更、または除去するには、フィルターを右クリックして、ポップアップ・メニューのオプションを使用します。フィルターに大/小文字の区別はありません。以下のようにして、フィルター内でワイルドカードを使用することができます。

- バージョン名フィルター `*xyz*` は、ストリング「xyz」をバージョン名のどこかに含む構成マップ・バージョン名すべてとマッチングします。
- バージョン名フィルター `xyz*` は、「xyz」から始まる構成マップ・バージョン名すべてとマッチングします。

- バージョン名フィルター *xyz は、「xyz」で終わる構成マップ・バージョン名とマッチングします。

「バージョン名」フィルター・フィールドを空欄のままにした場合、マイグレーション・ツールは「バージョン深さ」フィルターを使用します。

バージョン深さ

マイグレーション対象にする、前のバージョンの数を指定できます。デフォルトは 1 で、この場合マイグレーション・ツールは構成マップの最新バージョンのみを処理します。「バージョン名」フィルターを指定した場合、「バージョン深さ」フィルターは無視されます。

「マッピング」ページ

「マッピング」ページでは、以下の情報を指定することができます。

- 共通パーツと未使用パーツに対する EGL ファイル名。
- 特定の EGL のプロジェクト名とパッケージ名の作成に使用する接尾部。
- アプリケーション名を EGL パッケージ名に変換する方法を制御するオプション。
- 共通パーツを含む VAGen 構成マップとアプリケーションに関する情報。

このセクションでは、「マッピング」ページの設定について詳しく説明します。

ファイル名

「ファイル名」セクションでは、マイグレーション時に作成される 2 つの EGL ファイルの名前を制御することができます。

共通パーツ

マイグレーション・セットの有効範囲内で、複数の固有な生成可能パーツに共通するパーツを格納する EGL ファイルの名前を指定することができます。ファイル名は、拡張子またはパスを付けずに指定します。マイグレーション・ツールは、マイグレーション・セットにある複数の生成可能パーツによって使用されて (関連付けられて) いるパーツを含んでいるか、または共通構成マップまたはアプリケーションとして指定された VAGen 構成マップまたはアプリケーション内のパーツを含んでいる、それぞれの EGL パッケージ内に 1 つの共通パーツ・ファイルを作成します。パーツがプログラムとともにファイル内に配置されるか、または共通パーツ・ファイル内に配置されるかの判別については、50 ページの『EGL ファイル内でのパーツの配置』を参照してください。

未使用パーツ

マイグレーション・セットの有効範囲内で、使用されていないパーツを格納する EGL ファイルの名前を指定することができます。ファイル名は、拡張子またはパスを付けずに指定します。マイグレーション・ツールは、対応する VAGen 構成マップおよびアプリケーションが共通の構成マップまたはアプリケーションとして指定されていない場合には、マイグレーション・セットにある生成可能パーツによって使用されて (関連付けられて) いないパーツを含んでいる、それぞれの EGL パッケージ内に 1 つの未使用パーツ・ファイルを作成します。

スパン・マップ

「スパン・マップ」セクションでは、マップ・グループの 1 つに複数の構成マップまたはアプリケーションのマップが組み込まれている場合に使用される接尾部を指定することができます。

プロジェクト接尾部

新規 EGL プロジェクト名を作成するための接尾部を指定することができます。ステージ 1 マイグレーション・ツールは、この接尾部をマイグレーション・セット名に連結して新規 EGL プロジェクト名を作成します。マイグレーション・ツールは、マップ・グループとそのマップがマイグレーション・セット内で複数の VAGen 構成マップにわたっている場合に限り、この新規 EGL プロジェクトを作成します。新規プロジェクト名は、*migrationSetName_ProjectSuffix* です。マイグレーション・ツールは、「名前変更規則」がすべて適用された後、マイグレーション・セット名に接尾部を連結します。

パッケージ接尾部

EGL プロジェクト内で新規 EGL パッケージ名を作成するための接尾部を指定することができます。ステージ 1 マイグレーション・ツールは、この接尾部をプロジェクト名に連結して新規 EGL パッケージ名を作成します。マイグレーション・ツールは、マップ・グループとそのマップが構成マップ内で複数の VAGen パッケージにわたっている場合に限り、この新規 EGL パッケージを作成します。新規パッケージ名は、*projectName.PackageSuffix* です。マイグレーション・ツールは、「名前変更規則」がすべて適用された後、接尾部を連結します。

EGL パッケージ命名オプション

「EGL パッケージ命名オプション」セクションでは、Smalltalk アプリケーション名を Java パッケージ名に変換するための一般規則を指定することができます。

パッケージ命名ドット表記の使用

このオプションを選択すると、マイグレーション・ツールはアプリケーション名の中で先頭より後にある大文字の前にドットを付けることによって、VAGen アプリケーション名を EGL パッケージ名に変換します。このオプションを選択すると、マイグレーション・ツールは例えば `MyOrderEntryApp` を `My.Order.Entry.App` に変更します。

サブアプリケーションの縮小

このオプションを選択すると、マイグレーション・ツールは各 VAGen サブアプリケーションを EGL パッケージに変換します。このオプションをクリアすると、マイグレーション・ツールはドット表記を使用してサブアプリケーションを変換します。サブアプリケーション `SubApp` を含むアプリケーション `MainApp` がある状態を考慮します。「サブアプリケーションの縮小」を選択すると、マイグレーション・ツールは 2 つのパッケージ作成します。これらのパッケージには、それぞれ `MainApp` と `SubApp` という名前が付けられ、両方のパッケージが EGL フォルダ構造の同じレベルに置かれるようになります。「サブアプリケーションの縮小」をクリアす

ると、マイグレーション・ツールは 2 つのパッケージを作成します。これらのパッケージには、それぞれ MainApp と MainApp.SubApp という名前が付けられ、EGL フォルダ構造で階層式に表示されるようになります。デフォルトは選択済みです。

パッケージ名の小文字への変換

このオプションを選択すると、マイグレーション・ツールは大文字を小文字に変換することによって、VAGen アプリケーション名を EGL パッケージ名に変換します。このオプションを選択すると、マイグレーション・ツールは例えば MyOrderEntryApp を myorderentryapp に変更します。

通常は、「**パッケージ命名ドット表記の使用**」および「**パッケージ名の小文字への変換**」の両方を選択する必要があります。オプションを両方とも選択すると、マイグレーション・ツールは MyOrderEntryApp を my.order.entry.app に変更します。「**EGL パッケージ命名オプション**」は、すべての命名規則の後で適用されます。

共通 ID

このセクションでは、共通 (共用) パーツを含む構成マップとアプリケーションを判別するためにマイグレーション・ツールが使用できるストリングのリストを、ワイルドカードとともに指定できます。共通 ID を追加または削除するには、フィールドを右クリックして、ポップアップ・メニューのオプションを使用します。ID を追加するときには、次の情報を入力するようにエディターからプロンプトが出されます。

コンテキスト

ストリングを突き合わせる対象を、構成マップ名、アプリケーション名、またはその両方に指定します。

構成マップ

共通パーツを含む構成マップを識別するストリングを指定することができます。マイグレーション・ツールは、このストリングをマイグレーション・セット内の各構成マップ名と突き合わせて、構成マップに共通パーツが含まれているかどうかを判別します。構成マップ名がストリングのいずれかとマッチングする場合、その構成マップ内のパーツすべてが「使用済み」と見なされます。生成不可のパーツはそれぞれ、プログラム・ファイル内または「**共通パーツ**」設定で指定されたファイル内に置かれます。そのパーツは、マイグレーション・セット内の生成可能パーツによって使用されていない場合でも、未使用パーツ・ファイル内には置かれません。「**構成マップ (ConfigMap)**」のストリングは複数入力できます。

アプリケーション

共通パーツを含むアプリケーションを識別するストリングを指定することができます。マイグレーション・ツールは、このストリングをマイグレーション・セット内の各アプリケーション名と突き合わせて、アプリケーションに共通パーツが含まれているかどうかを判別します。ストリングがアプリケーション名にマッチングする場合、そのアプリケーション内

のパーツはすべて「使用済み」と見なされます。生成不可のパーツはそれぞれ、プログラム・ファイル内または「**共通パーツ**」設定で指定されたファイル内に置かれます。そのパーツは、マイグレーション・セット内の生成可能パーツによって使用されていない場合でも、未使用パーツ・ファイル内には置かれませんが、「**アプリケーション (Application)**」のストリングは複数入力できます。

両方 マイグレーション・ツールがマイグレーション・セット内の構成マップ名とアプリケーション名の両方に突き合わせるストリングを指定することができます。「**両方 (Both)**」は、「**構成マップ (ConfigMap)**」のコンテキストと「**アプリケーション (Application)**」のコンテキストに同じストリングを指定した場合に相当します。

共通コードを識別するためのパターン

指定したコンテキストに基づいてマイグレーション・ツールが突き合わせるストリングを指定することができます。ストリングの先頭または末尾で、* をワイルドカードとして使用できます。フィルターに大/小文字の区別はありません。

「名前変更」ページ

「名前変更 (Renaming)」ページを使用して、VAGen 構成マップ名、アプリケーション名、およびバージョン名から得られる EGL プロジェクト、パッケージ、およびバージョンの名前を制御できます。「**順序**」列の番号は、マイグレーション・ツールが名前変更規則を適用する順序を示し、最も小さい番号の規則が最初に適用されます。名前変更規則を追加または削除するには、規則をクリックして、ポップアップ・メニューのオプションを使用します。「**規則の追加**」では、新規の規則は常にリストの末尾に置かれます。規則を追加するときには、次の情報を入力するようにエディターからプロンプトが出されます。

変更するストリング

VAGen 名の中の変更したい文字を指定します。

変更後のストリング

変更後の EGL 名で使用する文字を指定します。

ストリング・コンテキスト

名前変更を行う際にマイグレーション・ツールが VAGen 名の中で変更するストリングを検索する場所を指定します。以下の値が有効です。

- | | |
|-----------|---|
| 前 | 変更するストリングが構成マップ名、アプリケーション名、またはバージョン名の先頭にある場合に規則を適用します。 |
| 後 | 変更するストリングが構成マップ名、アプリケーション名、またはバージョン名の最後にある場合に規則を適用します。 |
| 任意 | 変更するストリングが構成マップ名、アプリケーション名、またはバージョン名の中でいずれの位置にある場合でも規則を適用します。 |

トークン

変更するストリングが構成マップ名、アプリケーション名、またはバージョン名と完全に一致する場合にのみ規則を適用します。

マッピング・コンテキスト

マイグレーション・ツールが構成マップ名、アプリケーション名、またはバージョン名のうちのどれに名前変更規則を適用するのかを指示します。以下の値が有効です。

構成マップ

名前変更規則を VAGen 構成マップ名のみ適用します。

アプリケーション

名前変更規則を VAGen アプリケーション名のみ適用します。

両方 名前変更規則を VAGen 構成マップ名と VAGen アプリケーション名の両方に適用します。

バージョン

名前変更規則をすべての構成マップのバージョン名に適用します。ご使用のバージョン名が、ディレクトリー名やファイル名に使用できないセミコロン (;) などの特殊文字を含んでいる場合は、バージョン名変更規則を使用します。デフォルトの MigPreferences.xml ファイルには、バージョン名が無効なディレクトリー名やファイル名にならないようにするために役立つ、いくつかのバージョン名変更規則が組み込まれています。マイグレーション・ツールは、名前変更済みのバージョンを使用して、マイグレーションのステージ 1 でマイグレーション・プラン・ファイル名を作成し、ステージ 3 でディレクトリー名を作成します。

「実行」ページ

「実行 (Execution)」ページでは、マイグレーション・データベースの場所に関する情報、およびステージ 1 の実行中に収集するロギング、デバッグ、およびレポートの情報を指定できます。このセクションでは、「実行」ページで指定することができる設定について詳しく説明します。

データベース

このセクションでは、マイグレーション・データベースに関する詳細を次のように指定できます。

DB 「DB」は、マイグレーション・ツールがマイグレーション・セット情報を書き込むマイグレーション・データベースの名前です。マイグレーション・データベースの作成時の VGMIG からデータベース名を変更した場合は、この設定に指定されているデータベース名を変更して、使用した名前と一致させる必要があります。

スキーマ

データベース・テーブル用の修飾子として使用される名前。スキーマを指定しない場合、マイグレーション・ツールはデフォルトとして MIGSCHEMA を使用します。マイグレーション・データベースの作成時の MIGSCHEMA からスキーマ名を変更した場合は、この設定に指定されているスキーマ名を変更して、使用した名前と一致させる必要があります。

ユーザー ID

マイグレーション・データベースへの接続に必要なユーザー ID。ユーザー ID を指定しない場合、マイグレーション・ツールはデフォルトとして、VAGen SQL 設定で指定されているユーザー ID を使用して接続を試みます。接続に失敗した場合、マイグレーション・ツールはログオン・ユーザー ID の使用を試みます。どちらの試行も失敗した場合、マイグレーション・ツールは情報を尋ねるダイアログ・ウィンドウを表示します。

パスワード

マイグレーション・データベースへの接続に必要なパスワード。パスワードを指定しない場合、マイグレーション・ツールはデフォルトとして、VAGen SQL 設定で指定されているパスワードを使用して接続を試みます。接続に失敗した場合、マイグレーション・ツールはログオン・パスワードの使用を試みます。どちらの試行も失敗した場合、マイグレーション・ツールは情報を尋ねるダイアログ・ウィンドウを表示します。

注: パスワードは設定ファイル内で暗号化されません。このことが問題になる場合は、設定ファイルにパスワードを入力せず、プロンプトが出されるまで待ってください。

サービス

このセクションでは、ステージ 1 の間に収集したいロギングとデバッグの情報に関する詳細を次のように指定できます。以下の詳細を指定することができます。

トレース・レベル

ログ・ファイルとデバッグ・ファイルに書き込む情報のレベルを指定することができます。次の値から 1 つを選択できます。

致命的 エラー・メッセージがログに記録されます。このレベルのメッセージが出された場合、マイグレーション・データベースは更新される可能性があります。マイグレーション・プラン・ファイル (.pln ファイル) は変更されず .done 拡張子は付きません。このため、.pln ファイルを再度処理することができます。

警告 警告メッセージおよびエラー・メッセージがログに記録されます。

情報 情報メッセージ、警告メッセージ、およびエラー・メッセージがログに記録されます。

デバッグ

情報メッセージ、警告メッセージ、およびエラー・メッセージのほかに、デバッグ情報が記録されます。トレース・レベルが「デバッグ (DEBUG)」の場合のみ、マイグレーション・ツールはデバッグ・ファイルに情報を書き込みます。これはデフォルト値です。

ログ・ファイル名

ログ・ファイルのドライブ、ディレクトリ、およびファイル名を

指定することができます。任意のファイル拡張子を付けてログ・ファイルを作成できますが、このファイルは .xml ファイルとして最適に表示されます。ログ・ファイル名を省略した場合は、「**ログ・ファイル名**」フィールドで指定したドライブとディレクトリーに migLog.xml という名前のファイルが書き込まれます。ドライブとディレクトリーを指定しない場合、マイグレーション・ツールはログ・ファイルをマイグレーション・プラン・ディレクトリーに書き込みます。

デバッグ・ファイル名

IBM サポートで必要になる可能性のあるデバッグ・ファイルのドライブ、ディレクトリー、およびファイル名を指定することができます。任意のファイル拡張子を付けてデバッグ・ファイルを作成できますが、このファイルは .xml ファイルとして最適に表示されます。このファイルへの情報の書き込みは、「**トレース・レベル**」が「**デバッグ**」に設定されている場合にのみ行われます。デバッグ・ファイル名を省略して「**トレース・レベル**」として「**デバッグ**」を指定した場合は、「**デバッグ・ファイル名**」フィールドで指定したドライブとディレクトリーに migDebug.xml という名前のファイルが書き込まれます。ドライブとディレクトリーを指定しない場合、マイグレーション・ツールはデバッグ・ファイルをマイグレーション・プラン・ディレクトリーに書き込みます。

検証 このセクションでは、ステージ 1 マイグレーション・ツールから出力できるレポート・ファイルに関する情報を指定できます。以下の情報を指定することができます。

レポート・ファイル名

レポート・ファイルに使用するドライブ、ディレクトリー、およびファイル名を指定することができます。このレポートには、VAGen ファイルのマイグレーション方法に関する情報があります。必ず .htm 拡張子を指定してください。レポート・ファイル名を省略した場合は、「**レポート・ファイル名**」で指定したドライブとディレクトリーに report¥MigrationReport.htm という名前のファイルが書き込まれます。ドライブとディレクトリーを指定しない場合、マイグレーション・ツールはレポート・ファイルをマイグレーション・プラン・ディレクトリーに書き込みます。

MigPreferences.xml ファイルのサンプル

次に、MigPreferences.xml ファイルのサンプルを示します。

```
<preferences>
  <database>
    <uri>VGMIG</uri>
    <schema>MIGSCHEMA</schema>
    <userid></userid>
    <password></password>
  </database>
  <migrationSpec>
    <directory>d:\TempMig\Stage1</directory>
    <filename></filename>
  </migrationSpec>
  <service>
    <traceLevel>4</traceLevel>
```

```

        <logfile>d:\TempMig\stage1\migLog.xml</logfile>
        <debugfile>d:\TempMig\stage1\migDebug.xml</debugfile>
    </service>
    <repositoryFilters>
        <projectName>MyConfigMap*</projectName>
        <versionNumber>1</versionNumber>
    </repositoryFilters>
    <verification>
        <reportName>d:\TempMig\report\MigrationReport.htm</reportName>
    </verification>
    <eglMapping>
        <commonPartsFileName>CommonParts</commonPartsFileName>
        <unusedPartsFileName>UnusedParts</unusedPartsFileName>
        <spanningMapsProjectSuffix>MapsProject</spanningMapsProjectSuffix>
        <spanningMapsPackageSuffix>mapspackage</spanningMapsPackageSuffix>
        <packageDotNotation>true</packageDotNotation>
        <collapseSubapplications>true</collapseSubapplications>
        <packageLowercase>true</packageLowercase>
        <commonParts>
            <commonConfigMap>*Common*</commonConfigMap>
            <commonApplication>*Common*</commonApplication>
        </commonParts>
        <renameRule order="1">
            <fromString> </fromString>
            <toString></toString>
            <stringContext>any</stringContext>
            <mappingContext>both</mappingContext>
        </renameRule>
        <renameRule order="101">
            <fromString>CM</fromString>
            <toString></toString>
            <stringContext>back</stringContext>
            <mappingContext>configMap</mappingContext>
        </renameRule>
        <renameRule order="301">
            <fromString>App</fromString>
            <toString></toString>
            <stringContext>back</stringContext>
            <mappingContext>application</mappingContext>
        </renameRule>
        <renameRule order="501">
            <fromString>:</fromString>
            <toString>_</toString>
            <stringContext>any</stringContext>
            <mappingContext>version</mappingContext>
        </renameRule>
        <renameRule order="502">
            <fromString>/</fromString>
            <toString>_</toString>
            <stringContext>any</stringContext>
            <mappingContext>version</mappingContext>
        </renameRule>
        <renameRule order="503">
            <fromString>\</fromString>
            <toString>_</toString>
            <stringContext>any</stringContext>
            <mappingContext>version</mappingContext>
        </renameRule>
        <renameRule order="504">
            <fromString>|</fromString>
            <toString>_</toString>
            <stringContext>any</stringContext>
            <mappingContext>version</mappingContext>
        </renameRule>
        <renameRule order="505">
            <fromString>?</fromString>
            <toString>_</toString>

```

```

        <stringContext>any</stringContext>
        <mappingContext>version</mappingContext>
    </renameRule>
    <renameRule order="506">
        <fromString>*</fromString>
        <toString>_</toString>
        <stringContext>any</stringContext>
        <mappingContext>version</mappingContext>
    </renameRule>
    <renameRule order="507">
        <fromString>&lt;</fromString>
        <toString>_</toString>
        <stringContext>any</stringContext>
        <mappingContext>version</mappingContext>
    </renameRule>
    <renameRule order="508">
        <fromString>&gt;</fromString>
        <toString>_</toString>
        <stringContext>any</stringContext>
        <mappingContext>version</mappingContext>
    </renameRule>
    <renameRule order="509">
        <fromString>&quot;</fromString>
        <toString>_</toString>
        <stringContext>any</stringContext>
        <mappingContext>version</mappingContext>
    </renameRule>
    <renameRule order="510">
        <fromString> </fromString>
        <toString>_</toString>
        <stringContext>any</stringContext>
        <mappingContext>version</mappingContext>
    </renameRule>
</eglMapping>
</preferences>

```

設定からのファイル名の取得

ステージ 1 マイグレーション・ツールは、ログ・ファイル、デバッグ・ファイル、およびレポート・ファイルの名前に使用するファイル名を同じ方法で取得します。次の表に、設定の中で指定できる名前と、その結果としてマイグレーション・ツールが使用するドライブ、ディレクトリ、およびパス名を示します。この例では、マイグレーション・プラン・ディレクトリは `d:¥myVAGenMig` です。

表 62. 設定から得られるファイル名

ログ・ファイル名の設定	ステージ 1 マイグレーション・ツールが使用するファイル名
設定はブランクのまま。	<code>d:¥myVAGenMig¥migLog.xml</code> 注: <ul style="list-style-type: none"> デバッグ・ファイルのデフォルトのファイル名は <code>migDebug.xml</code> です。 レポート・ファイルのデフォルトのファイル名は <code>¥report¥MigrationReport.xml</code> です。
<code>mine.xml</code>	<code>d:¥myVAGenMig¥mine.xml</code>
<code>logs¥mine.xml</code>	<code>d:¥myVAGenMig¥logs¥mine.xml</code>
<code>.mine.xml</code>	<code>VisualAge-Generator-installation-directory¥image¥mine.xml</code>

ステージ 1 ツールを実行する前に - ヒント

ステージ 1 マイグレーション・ツールを実行する前に、以下のタスクを実行する必要があります。

- ステージ 1 マイグレーション・ツールのカスタマイズ
- 文字セット情報の指定
- ステージ 1 マイグレーション・ツールのパフォーマンスを改善する手順の実行
- イメージの保存

ステージ 1 マイグレーション・ツールのカスタマイズ

Smalltalk のステージ 1 マイグレーション・ツールには以下のカスタマイズが組み込まれており、ご使用の環境に適している場合はこれらを使用可能にできます。

- すべてのデータ項目パーツを、ItemsProject と呼ばれる新規 EGL プロジェクトに移動する。項目は、接尾部 .items を付けられた元の Smalltalk アプリケーション名に対応するパッケージ (例えば、my.original.app.items) 内に配置されます。このカスタマイズは、ステージ 2 マイグレーション設定で「**共用データ項目をプリミティブ項目定義に変換**」を選択する場合に特に役に立ちます。ステージ 1 でこのカスタマイズを使用可能化すると、マイグレーション時にすべてのデータ項目パーツを単一のプロジェクトに移動して、そのプロジェクトを EGL ワークスペースから除去することができます。
- パーツ型 (オプションで、パーツ名) 別に共通パーツ・ファイルを分割する。このカスタマイズにより、共通パーツ・ファイルのサイズが削減され、また特定のパーツを見つけやすくなります。

組み込みカスタマイズの他に、Smalltalk のステージ 1 の際に EGL プロジェクトおよびパッケージを統合する方法を説明したホワイト・ペーパーがあります。このホワイト・ペーパーの入手に関する情報については、19 ページの『参照』を参照してください。お使いの VAGen プロジェクトおよびパッケージ命名規則によっては、ホワイト・ペーパー手法の方が 186 ページの『「名前変更」ページ』で説明されている「命名規則」設定よりも簡単に使用することができます。

ステージ 1 組み込みカスタマイズの使用可能化

カスタマイズを使用可能化するには、以下の手順を実行します。

1. VisualAge Organizer の左側のペインで、「**HptEglMigrationUtilityApp**」アプリケーションを選択する。中央のペインで、「**HptEglMigrationSetUtility**」クラスをダブルクリックします。
2. 「スクリプト・エディター」ウィンドウで、「**クラス/インスタンス**」ボタンを切り替えて「**インスタンス**」に設定する。「**専用/公開**」ボタンを切り替えて「**公開**」に設定します。
3. 「カテゴリー」ペインで、「**PP - アクセサー**」を選択する。「メソッド」ペインで、「**customizedPlacementScriptSet**」を選択します。
4. すべてのデータ項目パーツを新規 EGL プロジェクトに移動するカスタマイズを使用可能化するには、以下の手順を実行します。
 - a. 次のように行を変更する。

```
consolidate := false.
```

から

```
consolidate := true.
```

デフォルトでは、ステージ 1 ツールにより EGL ファイル名が "Items" に設定され、データ項目名の最初の文字と連結されます。

- b. ファイル名に連結されたデータ項目名の文字数を変更する場合は、**customizedPlacementScriptSet** メソッドの `suffixLength` の値を変更します。
 - c. パッケージの EGL プロジェクトの名前または接尾部を変更する場合は、**consolidateDataItemsQuery**: メソッドを変更します。
5. パーツ型およびパーツ名によって共通パーツ・ファイルを分割するカスタマイズを使用可能化するには、以下の手順を実行します。
- a. 次のように行を変更する。

```
redistribute := false.
```

から

```
redistribute := true.
```

- b. 以下の行で、共通パーツ・ファイルの名前に追加するパーツ名の文字数を指定する。

```
suffixLength := 2.  
...  
suffixLength := 1.
```

デフォルトでは、`suffixLength` の値は、データ項目パーツ名の最初の文字とレコードおよび関数名の最初の 2 文字を使用するように設定されます。ただし、`suffixLength` の値は変更することができます。例えば、次のようになります。

- この数値を 0 に設定すると、共通パーツ・ファイル名は `CommonRecords.egl`、`CommonFunctions.egl`、および `CommonItems.egl` になります。
 - この数値を数字 n に設定すると、パーツ名の最初の n 文字がファイル名の接尾部として使用されます。 n を 1 に設定すると、名前は `CommonRecordsA`、`CommonRecordsB` などになります。
- c. ファイル名を別の名前にする場合は、**redistributeCommonRecordsUsing:**、**redistributeCommonFunctionsUsing:**、または **redistributeCommonItemsUsing:** メソッドを変更します。
6. 変更を保存し、クラスをバージョン設定して解放します。変更の記録を保持するために、アプリケーションをバージョン設定することもできます。

文字セット情報の指定

ステージ 1 マイグレーション・ツールでは、お使いの VAGen 各国語言語コードに基づいて文字セットを指定します。1 バイト言語ではこれは「iso-8859-1」になり、2 バイト言語では別の値が使用されます。別の文字セットを使用する必要がある場合は、以下の手順を実行してこの値を変更します。

1. VisualAge Organizer で、「アプリケーション」->「表示」->「すべてのアプリケーションの表示」をクリックする。

2. 左側のペインで、アプリケーション「**HptEglMigrationUtilityApp**」を選択する。中央のペインで、クラス「**EslmPreferences**」をダブルクリックする。
3. 「スクリプト・エディター」ウィンドウで、「**インスタンス/クラス**」ボタンを切り替えて「**インスタンス**」に設定する。「**公用/公開**」ボタンを切り替えて「**公開**」に設定する。
4. 「カテゴリ」ペインで、「**レポート**」を選択する。「メソッド」ペインで、「**determineXmlEncoding**」を選択する。
5. xmlCharset に指定された値を目的の文字セットに変更する。例えば、メソッドの最後の 3 行を以下のように変更します。

```
(vgNLS = 'PTB')
ifTrue: [ xmlCharset := 'iso-8859-1' ].
^xmlCharset.
```

太字で示された行を追加する。

```
(vgNLS = 'PTB')
ifTrue: [ xmlCharset := 'iso-8859-1' ].
xmlCharset := 'iso-8859-1'.           "new line with the value you need"
^xmlCharset
```

メソッドを保存する。

6. クラスをバージョン設定して解放する。変更の記録を保持するために、アプリケーションをバージョン設定することもできます。

パフォーマンスの向上

メモリーの使用量を最小限にするために、ステージ 1 マイグレーション・ツールを実行する前にイメージをクリーンアップ (または「消し込み」) することが最良策です。イメージをクリーンアップ (つまり「消し込み」) するには、以下の手順を実行します。

1. 「システム・トランスクリプト」で、「**ツール**」>「**VAGen ツール・ワークスペースを開く**」を選択する。
2. 「イメージ管理 (Image Management)」セクションで、次の項目を選択する。

System abtScrubImage.

3. 右クリックし、「**実行 (Execute)**」をクリックして System abtScrubImage を実行する。
4. イメージの消し込みを行うと、VAGen EGL マイグレーション・フィーチャーの再ロードが必要になる場合があります。フィーチャーをロードする方法については、178 ページの『マイグレーション・フィーチャーのロード』を参照してください。また、「システム・トランスクリプト」ツールバーに「EGL マイグレーション・ツール」オプションを再度追加するには、以下の手順を実行します。

- a. 「システム・トランスクリプト」ウィンドウに次のように入力する。

```
HptEglMigrationGuiApp loaded
```

- b. 入力した行を選択し、右クリックして「**実行 (Execute)**」をクリックする。

ステージ 1 マイグレーション・ツールがマイグレーション対象の構成マップとバージョンの分析に費やす時間を短縮するために、マイグレーションしたい構成マップ・バージョンのみを含むライブラリーの作成を検討してください。マイグレーション

ョン中に VisualAge Generator の継続保守を行う場合は、別個のマイグレーション・ライブラリーを使用すると次のような利点もあります。

- マイグレーションする構成マップ・バージョンのセットが安定します。これは、「バージョン深さ」設定を使用してマイグレーション対象を制御する場合に特に重要になります。
- 新規マイグレーション・ライブラリーのバージョンを保守ライブラリーと比較して、マイグレーションを必要とする他の構成マップ・バージョンがまだあるかどうか判別できます。

特殊ライブラリーを作成する場合は、ステージ 1 マイグレーションのパフォーマンスを高めるために、ローカル・ライブラリーとして使用することを検討してください。

イメージの保管

ステージ 1 マイグレーション・ツールは、ステージ 1 処理の開始時に、イメージからの VAGen パーツを含むアプリケーションをワークスペースからすべてアンロードします。ステージ 1 ツールの終了時には、最後に処理されるマイグレーション・セットのみがイメージに残ります。すべてのアプリケーションをアンロードすることにより、ステージ 1 の実行中に、マイグレーション・セット内のパーツのみが関連パーツ・リストの対象と見なされるようになります。保存するイメージがある場合は、ステージ 1 ツールを実行する前に以下の手順を実行する必要があります。

1. VisualAge Generator をシャットダウンする。
2. 次のファイルのバックアップ・コピーを、`¥VisualAgeForSmalltalk-installation-directory¥image` に保管する。
 - `abt.icx`
 - `abt.ini` (ステージ 1 の実行中に設定を変更しない場合は保管不要)
 - `hpt.ini` (ステージ 1 の実行中に設定を変更しない場合は保管不要)
3. VisualAge Generator を開始する。

ステージ 1 ツールの実行が完了したら、以下の手順を使用してワークスペースを復元します。

1. VisualAge Generator をシャットダウンする。
2. ステージ 1 ツールを実行する前にバックアップしたファイルを復元する。
3. VisualAge Generator を開始する。

ステージ 1 マイグレーション・ツールの実行

設定の編集を完了した後、ステージ 1 マイグレーション・ツールを実行して、ソース・コードを Smalltalk ライブラリーから抽出できます。このためには、次の手順で行います。

1. 次のいずれかの手法を使用して、「EGL マイグレーション・ドライバー (EGL Migration Driver)」ビューを開始する。

- 設定エディターを開始して設定を変更した場合は、「システム・トランスクリプト」で「**EGL マイグレーション・ツール**」->「**マイグレーション・ドライバー**」をクリックして、「**EGL マイグレーション・ドライバー**」ビューを開始する。
 - **EGL マイグレーション・ドライバー**を開始して設定を変更した場合は、設定ファイルを保管したときに、「**EGL マイグレーション・ドライバー (EGL Migration Driver)**」ビューが再度表示されます。
2. マイグレーション設定ファイルの「**ファイル名**」が、設定を保管したファイルを指していることを確認する。「**参照**」をクリックして、別の設定ファイルを指定する。「**編集**」をクリックして、設定の検討または最終的な変更を行う。
 3. 設定に問題がない場合は、使用する「**実行オプション**」を選択する。「**実行オプション**」は、ステージ 1 マイグレーション・ツールの出力を以下のようにして制御します。

PLN の上書き

マイグレーション・プラン・ファイルを、以下のようにして制御します。

- 「**PLN の上書き**」を選択すると、ステージ 1 マイグレーション・ツールは設定内で指定した「**プラン・ディレクトリー**」に基づいて、以下のようにして新規プラン・ファイルを作成します。
 - 設定ファイルで .pln ファイルのファイル名が指定されていない場合、マイグレーション・ツールは指定した「**プラン・ディレクトリー**」内の .pln ファイルをすべて削除し、ファイルを新規に作成します。
 - 設定ファイルで .pln ファイルのファイル名が指定されている場合、マイグレーション・ツールは指定した「**プラン・ディレクトリー**」から同じ名前のファイルのみを削除してから、.pln ファイルを新規に作成します。

リポジトリ・フィルターと上位構成マップに基づいて、ステージ 1 マイグレーション・ツールにマイグレーション・プラン・ファイルを作成させたい場合は、「**PLN の上書き (Overwrite PLN)**」オプションを使用します。マイグレーションに使用する構成マップを作成するために支援が必要な場合は、200 ページの『上位構成マップの作成』を参照してください。

- 「**PLN の上書き**」をクリアすると、ステージ 1 マイグレーション・ツールはマイグレーション・プラン・ファイルを新規に作成しません。代わりに、ステージ 1 マイグレーション・ツールは、設定内で指定した「**プラン・ディレクトリー**」および「**プラン・ファイル名**」に基づいて、以下のように既存のプラン・ファイルを使用します。
 - 設定ファイルで .pln ファイルのファイル名が指定されていない場合、マイグレーション・ツールは指定した「**プラン・ディレクトリー**」内のすべてのプランを使用して実行されます。
 - 設定ファイルに .pln ファイルのファイル名が指定されている場合、マイグレーション・ツールはその .pln ファイルのみを使用して実行されます。

マイグレーション・プラン・ファイルが既に作成されており、これらのファイルを使用してステージ 1 マイグレーション・ツールを実行し、マイグレーション・データベースをロードする場合は、「PLN の上書き」オプションをクリアします。ユーザー独自のマイグレーション・プラン・ファイルの作成について詳しくは、202 ページの『マイグレーション・プラン・ファイルの手動作成』を参照してください。

レポートの生成

マイグレーション・ツールが設定ファイルの「検証」セクションで指定されたレポートを作成するかどうかを制御します。このオプションを選択解除した場合、レポートは作成されません。このオプションを選択すると、マイグレーション・ツールは設定内で指定した「レポート・ファイル名」を使用してレポートを作成します。このレポートは、マイグレーション時に構成マップ、アプリケーション、および VAGen パーツが EGL のプロジェクト、パッケージ、およびファイルにどのように割り当てられるかを示します。

データベースの更新

マイグレーション・ツールがマイグレーション・プラン情報を使用してマイグレーション・データベースを更新するかどうかを制御します。このオプションを選択解除した場合、マイグレーション・データベースは更新されません。このオプションを選択すると、設定内で指定したマイグレーション・データベースが、マイグレーション・プランの情報 (マイグレーション・プランに含まれるすべての VAGen パーツの外部ソース形式など) を使用して更新されます。

4. 「実行オプション」を選択した後、「OK」をクリックしてマイグレーション・ツールのステージ 1 を実行します。マイグレーション設定の中で指定したログ・ファイルに、マイグレーション・ツールのメッセージが書き込まれます。また、システム・トランスクリプトにもツールの同じメッセージが書き込まれます。

パーツの数およびパーツ間の関連の複雑度によっては、ステージ 1 マイグレーション・ツールのいくつかのステップは長い時間を要する場合があります。「システム・トランスクリプト」ウィンドウの以下のステップは、顕著なアクティビティーがなくても特に時間を消費 (1 時間以上) します。

- partPlacementQuery 3 の実行
- partPlacementQuery 8 の実行
- eglPathBuilderQuery 8 の実行
- マイグレーション・レポート生成の開始

ステージ 1 マイグレーション・ツールは、次のようにいくつかのステップに分けて実行できます。

1. 「レポートの生成」および「データベースの更新」の両方をクリアする。これにより、作成されたマイグレーション・プラン・ファイルを検討して、目的の構成マップ・バージョンが処理されるようにリポジトリ・フィルタが正しく設定されていることを確認できます。選択された構成マップ・バージョンが適切でな

い場合は、マイグレーション・ツールが正しい構成マップ・バージョンを処理するようになるまで、リポジトリ・フィルターを調整してこのステップを再度実行することができます。

2. 「データベースの更新」を選択します (「レポートの生成」は選択してもしなくてもかまいません)。

注:

- データベースの更新には多少時間を要する場合があります。このため、.pln ファイルを検討して、目的の構成マップ・バージョンがマイグレーション・ツールの処理対象になるか確認することが最良策です。
- レポートを実行する前にデータベースを更新する必要があります。
- レポートの生成にも多少時間を要する場合があります。レポートを作成する代わりに、いくつかの単純な照会を実行して EGL ファイル名を確認することもできます。EGL ファイル名を生成する照会の例については、553 ページの『EGL ファイル名の検討』を参照してください。
- レポートを生成する場合、レポート・ファイルは上書きされます。前のレポート・ファイルを保管する必要がある場合は、レポート・ファイルを別のディレクトリに移動するか、新規レポート用のディレクトリを新しく指定する必要があります。レポート・ファイルは他のファイルにリンクしているため、レポート・ファイルの名前を変更するとそのリンクが失われ、ファイルが表示できなくなります。
- 「メッセージ・テーブル関連付けの修復 (Repair Message Table Associates)」オプションは、前のバージョンのステージ 1 マイグレーション・ツールを使用して作成されたマイグレーション・データベースを修復する必要がある場合にのみ使用されます。詳しくは、199 ページの『以前のバージョンのステージ 1 ツールを使用して作成されたマイグレーション・データベースの修復』を参照してください。

「データベースの更新」オプションを選択した場合、ステージ 1 ツールの終了時に、マイグレーション・プラン情報 (外部ソース形式の VAGen コードなど) がマイグレーション・データベースに格納されます。レポートとステージ 1 メッセージを検討した後、必要に応じて VisualAge Generator 内でコードを変更し、ステージ 1 を再度実行できます。「データベースの更新」を再度選択した際に、同じ名前のマイグレーション・セットがデータベース内に既に存在している場合、ステージ 1 マイグレーション・ツールはマイグレーション・セットに関する以前の情報をデータベースから自動的に削除し、マイグレーション・セットに関する新しい情報を追加します。このため、マイグレーション・セットをデータベースからクリーンアップする必要はありません。ただし、549 ページの『ステージ 1 のマイグレーション・データベースのリセット』で説明されている手法を使用するとより速く実行することができます。

ステージ 1 の結果に問題がなく、最終の外部ソース形式コードがマイグレーション・データベースに格納された後、マイグレーションのステージ 2 を実行できます。ステージ 2 マイグレーション・ツールを実行するには、EGL 開発環境を使用します。マイグレーション・プロセスの継続については、207 ページの『第 6 章 ステージ 2 - EGL 構文への変換』を参照してください。

以前のバージョンのステージ 1 ツールを使用して作成されたマイグレーション・データベースの修復

注: このセクションは、ツールのバージョンが 20071022_1400 より前のステージ 1 ツールを使用して作成されたマイグレーション・データベースが存在する場合にのみ適用されます。

メッセージ・テーブルを使用していて、20071022_1400 より前のバージョンのステージ 1 ツールを使用して作成されたマイグレーション・データベースが存在する場合は、マイグレーション・データベースを更新してメッセージ・テーブルをプログラムの関連付けとして組み込むこともできます。修復ツールを使用してからステージ 2 および 3 ツールを実行すると、これらのツールにより EGL ビルド・パスおよび **import** ステートメントを設定して、メッセージ・テーブルをプログラム・パーツの関連付けとして組み込むことができます。以前のステージ 1 データベースを更新するには、以下の手順を実行します。

1. 179 ページの『ステージ 1 設定の実行』で説明されているようにマイグレーションを設定する。修復ツールでとして使用する場合には、「実行」ページで「データベース」および「サービス」情報を済みます。
2. 「システム・トランスクリプト」で、「EGL マイグレーション・ツール」->「マイグレーション・ドライバ」をクリックする。
3. マイグレーション設定ファイルの「ファイル名」が、設定を保管したファイルを指していることを確認する。
4. 「メッセージ・テーブル関連付けの修復 (Repair Message Table Associates)」を選択する。
5. 「OK」をクリックします。修復ツールが実行され、メッセージ・テーブルの接尾部を指定する任意のプログラムに関するプログラム関連付けを更新し、接尾部に一致するすべてのメッセージ・テーブルを組み込みます。

マイグレーション・プランと上位構成マップ

マイグレーション・プラン・ファイルは、XML ファイルであり、1 つ以上のマイグレーション・セットの名前を指定し、それぞれのマイグレーション・セットごとに 1 つの上位構成マップとバージョンを指定して、マイグレーション・セットのアプリケーションとバージョンを指定します。上位構成マップは、他の構成マップとそのバージョンを必須マップとして指定することもできます。ただし、マイグレーション・セットに対して指定できる上位構成マップ・バージョンはただ 1 つです。ステージ 1 マイグレーション・ツールは、構成マップとバージョン名に関する「リボジトリ・フィルター」の設定に基づいてマイグレーション・プラン・ファイルを自動的に作成するように設計されています。ステージ 1 ツールは、どの構成マップ・バージョンが上位構成マップ・バージョンであるかを判別するために、これらのフィルターを使用して検討対象になる構成マップ・バージョンを選択します。ステージ 1 ツールは、それぞれの上位構成マップ・バージョンをマイグレーション・セットの基礎として使用します。

VAGen ソース・コードの生成時に上位構成マップを使用した場合は、これらと同じ上位構成マップをマイグレーションに使用します。これは、それぞれの上位構成マ

マップが生成時に一緒に使用されるパーツのグループを指定しているからで、したがって上位構成マップは一連のプログラムの関連パーツをすべて含んでいます。

構成マップを現在まったく使用していない場合は、マイグレーションに使用する構成マップを作成する必要があります。この場合、最も簡単な手法としては、グループとしてマイグレーションしたいアプリケーション・バージョンすべて (共通アプリケーション・バージョンを含む) を組み込んだ構成マップ・バージョンを 1 つ作成します。詳しくは、200 ページの『上位構成マップの作成』を参照してください。構成マップを作成した後、ステージ 1 マイグレーション・ツールを使用して、マイグレーション・プランを自動的に作成できます。

構成マップを現在使用している場合は、上位構成マップが存在しない可能性があります。例えば、共通アプリケーション用の構成マップと、サブシステム用の別の構成マップがあるとします。生成時に、それぞれの構成マップのどのバージョンをイメージにロードするか決定します。この場合は、以下の手法のいずれかを実行して、グループとしてマイグレーションする対象を指定することができます。

- マイグレーション中に使用する上位構成マップを作成する。この上位構成マップには、アプリケーション・バージョンのリスト、必要な構成マップ・バージョンのリスト、またはアプリケーション・バージョンと必要な構成マップ・バージョンの組み合わせを指定できます。例えば、上位構成マップで共通構成マップとサブシステム構成マップをリストして、両方の構成マップがマイグレーション時にグループとして処理されるようにすることができます。詳しくは、200 ページの『上位構成マップの作成』を参照してください。上位構成マップを作成した後、ステージ 1 マイグレーション・ツールを使用して、マイグレーション・プランを自動的に作成できます。
- 上位構成マップを作成しない場合は、次のいずれかの手法を使用して、マイグレーション・プラン・ファイルをユーザーが独自に作成できます。
 - Smalltalk 構成マップ・バージョンに関連して、生成に必要なものを指定する情報がデータベースや他のシステム内にある場合は、データベースからマイグレーション・プラン・ファイル (複数可) を自動的に作成するツールをユーザーが開発できます。
 - マイグレーション・プラン・ファイル (複数可) を手作業で作成できます。

上位構成マップの作成

マイグレーションに使用する上位構成マップを作成するには、VisualAge Generator 内で以下の手順を実行します。

1. VisualAge Organizer で「オプション」をクリックして、「全メニュー」が選択されていることを確認する。
2. VisualAge Organizer で「ツール」->「構成マップ」をクリックする。
3. 構成マップ・ブラウザーで「名前」->「作成」をクリックする。
4. 「必須情報」ウィンドウに構成マップの名前を入力して、「OK」をクリックする。構成マップの新規エディションが自動的に作成され、選択されます。
5. 「アプリケーション」->「追加」をクリックする。続いて、マイグレーションするそれぞれのアプリケーションと、そのアプリケーションのバージョンを選択す

る。それぞれのアプリケーションごとに、バージョンをただ 1 つ指定できます。組み込む必要があるそれぞれのアプリケーションごとにバージョンを選択したら、「OK」をクリックします。

6. 「エディション」->「バージョン」をクリックして、構成マップのバージョンを設定する。
7. 「エディション」->「ロード」をクリックして、構成マップ・バージョンを選択し、イメージにロードする。
8. 新しい上位構成マップをロードした後、プログラムとテーブルの検証を実行して、これらが VAGen 内で有効であることと、欠落しているパーツがないことを確認することもできます。プログラムを検証する際には、/GENMAPS、/GENHELPMAPS、および /NOGENTABLES を組み込みます。これら 2 つのマップ・オプションを指定すると、マップが使用されるプログラム内で有効であることを確認できます。/NOGENTABLES を指定すると、テーブルが使用されるすべてのプログラムを対象にテーブルを再検証せずに、テーブルを一度限り検証できます。

構成マップのチェーニング

構成マップはチェーニングできます。例えば、それぞれの共通アプリケーションのバージョンをリストする構成マップを作成できます。その後、サブシステムごとに、サブシステム固有のアプリケーションそれぞれの必要なバージョンを組み込んで、そのサブシステム用の上位構成マップを作成します。以下の手順を実行して、サブシステム構成マップに共通アプリケーションの構成マップを組み込むことができます。

1. 構成マップ・ブラウザーから、サブシステム構成マップのオープン・エディションを選択する。
2. 「式」->「追加」をクリックする。
3. 「必須情報」ウィンドウで「OK」をクリックして、式として「真」を受け入れる。
4. 「式の構成」ペインで「真」をクリックする。次に、「必須マップ」->「追加」->「先頭として」をクリックする。続いて、共通アプリケーションを含む構成マップ・バージョンを選択する。
5. 「エディション」->「バージョン」をクリックして、構成マップのバージョンを設定する。
6. 「エディション」->「必須マップとともにロード」をクリックして、構成マップ・バージョンを選択し、イメージにロードする。

必須マップを使用すると共通アプリケーション・バージョンを簡単に指定することができ、すべてのサブシステムの上位構成マップ内にすべての共通アプリケーション・バージョンを明示的にリストする必要がなくなります。

ステージ 1 ツールに対する構成マップの使用

ステージ 1 マイグレーション・ツールの実行準備完了後、以下の手順を実行します。

1. ステージ 1 設定を行う際に、「プランの作成」ページの「リポジトリ・フィルター」セクションで「構成マップ」リストを設定し、作成した上位構成マップにリスト内のフィルターが一致するようにします。

2. 使用する設定ファイルをステージ 1 ツールに指定するときに、「**PLN の上書き (Overwrite PLN)**」オプションも選択します。このオプションを指定すると、ステージ 1 マイグレーション・ツールは上位構成マップに基づいてマイグレーション・プラン・ファイルを作成し、既存のマイグレーション・プラン・ファイルを上書きします。

マイグレーション・プラン・ファイルの手動作成

VisualAge Generator での生成時にイメージにロードする構成マップ・バージョンを決定する外部制御手段がすでに存在する場合は、マイグレーション・プラン・ファイルを手動で作成するか、外部情報からマイグレーション・プラン・ファイルを自動的に作成するツールを開発できます。マイグレーション・プラン・ファイルのファイル拡張子は *.pln* にする必要があり、フォーマットは次のとおりです。

```
<migrationDefinition>
  <migrationSet
    name="migrationSet1"
    version="migrationSet1Version1"
    vgName="migrationSet1"
    vgVersion="migrationSet1Version1">
    <configMap
      name="configurationMap1"
      version="configurationMap1Version1">
    </configMap>
    <configMap
      name="configurationMap2"
      version="configurationMap2Version1">
    </configMap>
    .
    .
    .
    <configMap
      name="configurationMapN"
      version="configurationMapNVersion1">
    </configMap>
  </migrationSet>
</migrationDefinition>
```

以下の説明は上の例に適用されます。

- *migrationSet1* は、同時にマイグレーションする必要がある構成マップのグループを参照するために使用することができる名前です。マイグレーション・セット名はマイグレーション・データベースに保管され、以降のマイグレーション・ステージで以下のようにして使用されます。
 - ステージ 1 マイグレーションでは、マップ・グループ内のマップが複数の構成マップにわたる場合、接尾部と連結されたマイグレーション・セット名を使用して、マップ・グループとそのすべてのマップを格納する新規 EGL プロジェクトの名前が作成されます。名前変更規則を変更する場合にも、マイグレーション・データベースから情報を除去するためにマイグレーション・セット名が使用されます。
 - ステージ 2 マイグレーションの場合、マイグレーション・セット名は、マイグレーション・データベース内で EGL に変換する対象の構成マップ・グループを指定します。
 - ステージ 3 の場合、マイグレーション・セット名は、ワークスペースまたは一時ディレクトリーに EGL プロジェクト、パッケージ、およびファイルを作成するために使用する、マイグレーション・データベース内の構成マップ・グ

ループを指定します。ステージ 3 の出力を一時ディレクトリーに保管する場合、マイグレーション・セット名とマイグレーション・セット・バージョンは、上位ディレクトリー名の作成にも使用されます。

マイグレーション・セット名は、マイグレーション時に構成マップ・グループを識別する手段としてのみ使用されます。マップが VisualAge Generator 内で複数の構成マップにわたっている場合を除き、マイグレーション・セット名はマイグレーション後には使用されません。

- *configurationMap1*、*configurationMap2*、...、*configurationMapN* は、グループとしてマイグレーションする構成マップです。*configurationMap* は、マイグレーション・セット内で一度限りリストする必要があります。
- *configurationMap1Version1*、*configurationMap2Version1*、...、*configurationMapNVersion1* は、これらの各構成マップに対応するバージョンです。1 つのマイグレーション・セット内で、それぞれの構成マップごとにバージョンをただ 1 つ指定できます。
- 指定する構成マップ名およびバージョン名は、ライブラリー内の構成マップ名およびバージョン名と完全に一致している必要があります。名前には大/小文字の区別があります。この情報を使用して、イメージに構成マップ・バージョンが追加され、ステージ 1 マイグレーション・レポートの作成、およびマイグレーション・データベースのロードのためにパーツを分析できるようになります。

ステージ 1 マイグレーション・ツールの実行準備完了後、以下の手順を実行します。

1. ステージ 1 設定を行う際に、「プランの作成」ページで「**プラン・ディレクトリー**」をマイグレーション・プラン・ファイルの保管先のドライブとディレクトリーに設定する。作成したマイグレーション・プランを **1** つだけ使用してステージ 1 マイグレーション・ツールを実行する場合は、「**プラン・ファイル名**」を指定します。指定した「**プラン・ディレクトリー**」にあるすべてのマイグレーション・プラン・ファイルを使用してステージ 1 マイグレーション・ツールを実行する場合は、「**プラン・ファイル名**」をブランクのままにします。
2. 作成する出力をステージ 1 ツールに指示する場合は、必ず「**PLN の上書き**」をクリアしてください。こうすると、「**プラン・ファイル名**」の設定に基づいて以前に作成した .pln ファイルを使用してマイグレーション・ツールが実行されます。

第 4 部 ステージ 2 および 3 - 共通のマイグレーション・ステップ

以降のマイグレーション手順は、VisualAge Generator on Java、または VisualAge Generator on Smalltalk のどちらからマイグレーションする場合も同じです。

第 6 章 ステージ 2 - EGL 構文への変換

マイグレーションのステージ 2 は、Java または Smalltalk のどちらからマイグレーションする場合も同じです。

別のマイグレーション・ツールを実行して、ソースを外部ソース形式の構文から EGL 構文に変換する必要があります。このマイグレーション・ツールは EGL のインストール後に使用可能になるプラグインです。ツールは、バッチ・モードまたは対話モードで実行できます。オプションで、ステージ 2 の完了後にステージ 3 を自動的に実行するように指定できます。

DB2 パフォーマンス情報の設定

ステージ 1 を実行した後で、ステージ 2 を実行する前に、DB2 **runStats** コマンドを使用して DB2 テーブルのパフォーマンス情報を評価および設定する必要があります。パフォーマンス情報を設定するには、次のステップを実行します。

1. DB2 コマンド・ウィンドウから、runStats.bat があるディレクトリーにナビゲートします。
 - Java の場合、このディレクトリーは *VisualAge-for-Java-install-directory\ide\vgmigration* です。
 - Smalltalk の場合、このディレクトリーは *VisualAge-Smalltalk-install-directory* です。
2. デフォルトのマイグレーション・データベース名 (VGMIG) またはデフォルトのスキーマ名 (MIGSCHEMA) を変更した場合、runStats.bat ファイルを変更してご使用のデータベース名およびスキーマ名を使用します。
3. runStats.bat を実行します。

マイグレーション・データベースをバックアップすることもできます。これにより、ステージ 2 で別のマイグレーション設定またはオプションを試行する場合に、データベースをステージ 1 の終了時の状態に復元できるようになります。詳しくは、555 ページの『マイグレーション・データベースのバックアップおよび復元』を参照してください。

ワークベンチの設定

マイグレーションを開始する前に、次のセクションに説明されているように、ワークベンチの設定を行う必要があります。

- 始動パラメーター
- 必要な EGL 設定
- 推奨設定
- VAGen マイグレーションの設定
- その他の推奨設定

始動パラメーター

EGL 開発環境のパフォーマンスを向上させるには、初期設定ファイルにいくつかの始動パラメーターの設定が必要になる場合があります。使用する製品に関わらず、パラメーターは同じです。初期設定ファイルは常に製品のインストール・ディレクトリにあります。ファイルの名前はご使用の製品によって異なります。例えば、Rational Business Developer を使用している場合、初期設定ファイルの名前は eclipse.ini です。始動パラメーターを設定するには、次のステップを実行します。

1. テキスト・エディターを使用して、初期設定ファイルを編集する。
2. 次のパラメーターを変更します。

```
-Xms256m  
-Xmx1024m
```

-Xms の設定により、製品を始動するときに使用されるメモリーの量が増加します。この値を、-Xmx の設定値より小さいか等しい値にします。

-Xmx を設定することにより使用可能メモリーが増加します。通常、これを実メモリー値未満に設定します。例えば、実メモリーが 2000K であれば、-Xmx を 1024 に設定します。これが仮想メモリーの使用の回避に役立ちます。

3. 初期設定ファイルを保管する。
4. EGL 開発環境を開始する。例えば、Rational Business Developer を使用している場合は、その製品を開始してください。

必要な EGL 設定

マイグレーション・ツールを使用する前に、EGL とマイグレーション・ツールの機能を両方とも使用可能にする必要があります。これらの機能を使用可能にするには、次のステップを実行します。

1. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」をクリックします。
2. 「一般」設定の横にある「+」を選択して設定を展開し、「機能」を選択します。
3. 「拡張」をクリックします。
4. 「拡張機能設定 (Advanced Capabilities Settings)」ウィンドウで、「EGL デベロッパー」を展開します。
5. 「EGL VAGen マイグレーション」を選択します。マイグレーションを予定しているアプリケーションの種類によっては、次の機能の選択も必要になる場合があります。
 - EGL DLI
 - EGL MQ
 - EGL テキスト UI
 - EGL VG Web トランザクション
6. 「OK」をクリックして、「拡張機能設定 (Advanced Capabilities Settings)」ウィンドウを閉じます。
7. 「OK」をクリックして、「設定」ウィンドウを閉じます。

マイグレーションの前に、VisualAge Generator 互換モードも設定する必要があります。VisualAge Generator 互換モードの設定により、「問題」ビューに膨大な数のメッセージが表示されなくなります。この設定を行うには、次のステップを実行します。

1. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」をクリックします。
2. 「EGL」を選択します。
3. 「VisualAge Generator との互換性」設定を選択します。
4. 「OK」をクリックします。
5. ワークスペースの完全再ビルドのプロンプトが出されたら、「OK」をクリックします。

VAGen Web トランザクション・プログラムをマイグレーションする予定の場合は、EGL プロジェクトのタイプを設定しておくべきです。この設定を行うには、次のステップを実行します。

1. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」をクリックします。
2. 「EGL」を選択します。
3. 「デフォルト EGL Web プロジェクト・ファセットの選択 (Default EGL Web Project Facet Choices)」セクションで、次のステップを実行します。
 - a. 「EGL の JSF サポート」および「EGL の JSF コンポーネント・インターフェースのサポート」を選択解除します。
 - b. 「EGL のレガシー Web トランザクションのサポート」を選択します。
4. 「コード・マイグレーションのデフォルト Web ランタイム」セクションで、使用する Web アプリケーション・サーバーを選択します。「新規」を選択し、ウィザードを使用してサーバーに必要な追加オプションを指定します。これらのオプションは、有効な EGL Web プロジェクトを作成するためにマイグレーションのステージ 3 で使用されます。
5. 「OK」をクリックします。

直接 MQ API 呼び出しを使用する VAGen プログラムのマイグレーションを予定している場合は、次のステップを実行して、MQ API のサポート設定を選択する必要があります。

1. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」をクリックします。
2. 「EGL」を選択します。
3. 「デフォルトの EGL プロジェクト機能の選択項目 (Default EGL Project Features Choices)」セクションで、「EGL の下位 MQ API サポート (EGL with low-level MQ API support)」を選択します。
4. 「OK」をクリックします。

すぐに EGL を使用してレポートを作成する予定がない場合は、次のステップを実行して、レポート関連設定をクリアする必要があります。

1. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」をクリックします。

2. 「EGL」を選択します。
3. 「デフォルトの EGL プロジェクト機能の選択項目 (Default EGL Project Features Choices)」セクションで、次のオプションをクリアします。
 - BIRT レポートがサポートされている EGL
 - jasper レポートがサポートされている EGL
4. 「OK」をクリックします。

VAGen ソース・コードをマイグレーションする際にサービスをすぐに作成する予定がない場合は、次の方法で EGL デプロイメント記述子の設定をクリアすることができます。

1. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」をクリックします。
2. 「デフォルトの EGL プロジェクト機能の選択項目 (Default EGL Project Features Choices)」セクションで、次のオプションをクリアします。
 - EGL サービスのデプロイメント記述子の作成
3. 「OK」をクリックします。

VAGen 制御パーツで英語以外の特殊文字を使用している場合は、次のステップを実行して、エンコード設定を行うことができます。

1. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」をクリックします。
2. 「EGL」を選択します。
3. 「eglbld ファイルの作成 (Creating .eglbld Files)」セクションで、ドロップダウン・リストを使用して「エンコード」設定にコード・ページを選択します。
4. 「OK」をクリックします。

推奨設定

他の設定を行うには、「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」とクリックして、適切な設定ページを選択します。次の設定は、「問題」ビューで EGL 検証メッセージを解決する際に役立ちます。

- 「EGL」->「エディター」。「行番号の表示」を選択します。
- 「一般」->「エディター」->「テキスト・エディター」。「行番号の表示」を選択します。
- 「一般」->「ワークスペース」。「自動的にビルド」を選択するか、クリアするかを決めてください。このオプションを選択すると、ファイルを保管するたびに、EGL によってワークスペース内のものがすべて再ビルドされ、検証が実行されます。このオプションを選択する場合の利点は、行った変更のフィードバックが即時に得られることです。欠点は、ワークスペース内のパーツの数によっては再ビルドに時間がかかる可能性があることです。このオプションを選択解除すると、ファイルの保管時に、EGL による再ビルドは行われません。このオプションをクリアした場合の利点は、複数のファイルを変更するときに再ビルドが複数回行われないことです。ただし、メッセージに対する変更の結果を「問題」ビューに表示するには、プロジェクトをそのたびに再作成する必要があります (「プロジェクト」->「プロジェクトのビルド」または「プロジェクト」->「すべてビルド」)。マイグレーション・ログのメッセージのリストを処理しているときは、

「自動的にビルド」をクリアできます。これにより、再ビルドが行われる時期を制御できます。通常のコード開発を行うときには、このオプションを選択できます。ステージ 3 マイグレーション・ツールは常に、「自動的にビルド」を使用不可にします。

- 「一般」->「ワークスペース」->「ローカル・ヒストリー」。マイグレーション・ツールによって作成される .egl ファイルは、非常に大きくなる可能性があります。このため、「最大ファイル・サイズ (MB)」を大きな値に変更する必要があります (例: 5)。また、バックアップ要件に応じて、「ファイルを保持する日数」と「ファイル当たりの最大のエントリー数」を変更する必要が生じることもあります。
- 「一般」->「パースペクティブ」。EGL パースペクティブ、または Web パースペクティブをデフォルト・パースペクティブとして設定できます。Web アプリケーションを開発しない場合は、EGL パースペクティブを使用します。Web トランザクションをマイグレーションしたり Web アプリケーションを開発したりする予定の場合は、Web パースペクティブを使用します。デフォルト・パースペクティブを設定するには、使用するパースペクティブを選択して、「デフォルトにする」をクリックします。

VAGen マイグレーションの設定

以下のセクションで説明する設定は、マイグレーション・プロセス全体を制御します。特に注記がない限り、これらの設定は、ステージ 1 から 3 のマイグレーション中のステージ 2 と、単一ファイル・マイグレーションの両方に使用されます。これらの設定を行うには、「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」->「VAGen マイグレーション」をクリックします。

シングル・ファイル・モード設定:

- **パーツを EGL ファイルに分離。**この設定は、単一ファイル・マイグレーション時にのみ使用されます。この設定を選択すると、それぞれのプログラム、マップ・グループ、およびテーブルが個別のファイルに配置されます。その他のパーツは、「VAGen 外部ソース形式ファイルのインポート (Import VAGen External Source Format File)」ウィザードに指定したファイルに配置されます。これは、ファイルごとに 1 つの生成可能パーツを配置するという EGL の要件に準拠しています。この設定を選択解除した場合は、「VAGen 外部ソース形式ファイルのインポート (Import VAGen External Source Format File)」ウィザードに指定した EGL ファイルに、すべてのパーツが配置されます。単一ファイル・モードのパーツ配置アルゴリズムの詳細については、32 ページの『単一ファイル・マイグレーションの概要』を参照してください。

名前変更ユーザー出口情報:

- **ユーザー出口の名前変更。**VAGen マイグレーション・ツールは、パーツ名またはパーツ参照に接頭部を追加することによって、データ項目、レコード、関数、およびマップの単純な名前変更を行います。オプションで、ユーザー出口ルーチンを作成して、より複雑な名前変更を行うことができます。例えば、マイグレーション時にハイフン (-) を下線 (_) に変更できます。パーツの名前を変更するためのユーザー出口ルーチンを提供する場合は、「ユーザー出口の名前変更」の設定を選択します。このオプションを選択する場合は、「ユーザー出口の名前変更」ルーチンに関する追加情報を指定する必要があります。「ユーザー出口の名前変

更」の作成に関する詳細、およびサンプル・コードについては、ホワイト・ペーパー「Using the Rename User Exit in the VisualAge Generator to EGL Migration Tool」を参照してください。

- **JAR ファイルのロケーション (JAR file location)**。「ユーザー出口の名前変更」ルーチンを含む、ご使用のシステム上にある .jar ファイルのロケーションを指定します。
- **パッケージ名 (Package name)**。「ユーザー出口の名前変更」ルーチンを含む、.jar ファイル内のパッケージの名前を指定します。
- **クラス名**。名前変更ロジックを含む、パッケージ内のクラスの名前を指定します。このクラスには、メソッド `renameUserExit(String s, Connection c)` が含まれている必要があります。
- **データベースを使用 (Use a database)**。ご使用の「ユーザー出口の名前変更」が、データベースを使用して前のパーツ名と新規パーツ名の間の関係を取得する場合は、このオプションを選択します。

VisualAge Generator 互換モードを最小化する: この設定グループによって、VisualAge Generator 互換性モードを必要とするマイグレーション技法が自動的に使用される回数を最小化することができます。これらの設定を選択すると、結果として EGL「問題」ビューのエラー・メッセージやランタイムの振る舞いの変更点が増加する可能性がありますので注意してください。

- **以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)**。この設定を選択すると、マイグレーション・ツールは、EZESYS の以前の VAGen 値を組みこむための宣言および初期化ステートメントを、追加の変数でそれぞれのプログラム中に組み込みません。マイグレーション・ツールは、IF、WHILE、または TEST 以外の EZESYS を含むステートメントを変換するときには、追加の変数を使用します。この設定を選択した場合に、IF、WHILE、または TEST 以外の EZESYS を含むステートメントが存在すると、EGL 検証ではマイグレーションの後で「問題」ビューにエラー・メッセージが表示されます。すべてのステートメントを変換して新規の EGL `sysVar.systemType` 値を使用する予定である場合や、IF、WHILE、または TEST 以外の EZESYS をステートメントに使用していないことがわかっている場合に、この設定の選択を検討します。詳しくは、142 ページの『EZESYS』を参照してください。
- **テーブルの deleteAfterUse を組み込まない**。この設定を選択すると、マイグレーション・ツールは常に、それぞれのプログラムで `DataTable use` 宣言ステートメントの `deleteAfterUse` プロパティを省略します。マイグレーション・ツールは、`deleteAfterUse` プロパティが省略される状態について、警告メッセージを出します。VisualAge Generator バージョン 4.5 フィックスパック 4 以降から直接マイグレーションしており、かつすべての製品プログラムが、そのバージョンおよびフィックスパックで生成される場合には、この設定を検討してください。システム共通プロダクトまたは VisualAge Generator の前のバージョンからのマイグレーション中にこの設定を選択する場合は、必ず、`DataTable` の `use` 宣言から `deleteAfterUse` プロパティが省略されるすべてのプログラムを十分にテストしてください。
- **項目または変数の evensql=y を受け入れない**。この設定を選択すると、マイグレーション・ツールは、PACK データ項目部分またはレコードの非共用項目をマイグレーションするときに、常に奇数精度 (項目が最大長である場合は 18) を使用します。マイグレーション・ツールは、偶数精度 (`evensql=y`) を指定された

VAGen 項目がある状態で、警告メッセージを出します。ご使用の SQL 表で、SQL の WHERE 文節または EGL の **prepare** ステートメントで参照する可能性のある列で偶数精度を使用していないことが確実である場合、この設定を検討してください。あるいは、この設定を選択してマイグレーションした後で、マイグレーション・ツールが警告メッセージを出した項目定義を検査することができます。SQL 表定義に指定されていない精度を使用すると、データベース・アクセスのローパフォーマンスの原因になります。

- **互換モードを設定しない (Do not set compatibility mode)**。この設定を選択すると、マイグレーション・ツールは、生成オプションの部分を変換するときはいつても、**vagCompatibility="YES"** ビルド記述子を省略します。この設定は、以下のステップをすべて実行する場合にのみ選択する必要があります。
 1. 「VisualAge Generator 互換モードの最小化」セクションの他の 3 つの設定をすべて選択します。
 2. VisualAge Generator 互換モードが使用不可になったときに、マイグレーションされたすべてのパーツ名が EGL パーツの命名規則に従っている必要があります。このために「ユーザー出口の名前変更」を指定し、マイグレーション中に VAGen パーツを名前変更するための出口をコーディングすることができます。
 3. マイグレーションを終了した後に、VisualAge Generator 互換性に関する EGL 設定を使用不可にします。注: 設定内容に関わらず、マイグレーション・ツールはワークスペースをリフレッシュするときに、VisualAge Generator 互換モードを常にオンにします。

VAGen マイグレーション・データベースの I/O 設定

次のセクションで説明する設定によって、SQL および DL/I データベース I/O に関する VAGen 構文から EGL 構文へのマイグレーションが制御されます。特に注記がない限り、これらの設定は、ステージ 1 から 3 のマイグレーション中のステージ 2 と、単一ファイル・マイグレーションの両方に使用されます。これらを設定するには、「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」->「VAGen マイグレーション」->「VAGen マイグレーション・データベースの I/O 設定」をクリックします。

SQL 設定:

- **結果セット接尾部**。VisualAge Generator の場合、SQL REPLACE 関数が対応する UPDATE 関数または SETUPD 関数を参照する必要がある場合に、REPLACE 関数は関数名を指定する必要があります。EGL の場合は、単一の関数内で複数の入出力がサポートされます。**get** ステートメントまたは **open** ステートメントを一意的に識別するために、結果セット ID が使用されます。マイグレーション・ツールは、関数名に結果セット接尾部を付加して結果セット ID を作成します。例えば、関数の名前が MY-SETUPD で、デフォルトの結果セット接尾部 **_RSI01** を使用する場合、関数の **open** ステートメントに組み込まれる結果セット ID は MY-SETUPD_RSI01 になります。結果セット接尾部は、ブランク以外の任意の値に設定できます。ただし、どのパーツ名とも競合を起こさないものを必ず選択してください。
- **Prepare 接尾部**。VisualAge Generator では、SQL 入出力ステートメントを実行時に準備する必要がある場合に、Execution time statement build を選択します。対応する EGL ステートメントは、**prepare** ステートメントです。 **close**、**get**、

execute、および **open** などの他の I/O ステートメントが、どの **prepare** ステートメントと関連するのかを指定できるように、**prepare** ステートメントには **prepare** ステートメント ID が組み込まれます。マイグレーション・ツールは、関数名に **Prepare** 接尾部を付加して **prepare** ステートメント ID を作成します。例えば、MY-EXEC-TIME-BUILD という名前の関数が存在し、デフォルトの **Prepare** 接尾部 **_PREP01** を使用する場合、**prepare** ステートメントに組み込まれる **prepare** ステートメント ID は MY-EXEC-TIME-BUILD_PREP01 です。**Prepare** 接尾部は、ブランク以外の任意の値に設定できます。ただし、どのパーツ名とも競合を起こさないものを必ず選択してください。

- **列名を省略**。VisualAge Generator 内では、SQL レコード内のフィールドに対して SQL 列名が常に指定されます。EGL の場合は、フィールド名と同じならば SQL 列名を省略できます。「**列名を省略**」を選択すると、SQL 列名がフィールド名と同じである場合に、マイグレーション・ツールは EGL の列プロパティを省略します。列名を省略すると、EGL ソース・コードを読みやすくなります。
- **isSQLNullable プロパティを省略**。VisualAge Generator 内では、SQL レコードのすべてのフィールドに NULL 標識変数が常に組み込まれます。VisualAge Generator の振る舞いを保持するために、マイグレーション・ツールは SQL レコードのすべてのフィールドに **isSQLNullable** プロパティを常に組み込みます。ただし、EGL の場合、列が SQL で非 NULL として定義されていれば、NULL 標識変数を省略できます。「**isSQLNullable プロパティを省略**」を選択すると、マイグレーション・ツールは SQL レコード内のすべてのフィールドから **isSQLNullable** プロパティを省略します。**isSQLNullable** プロパティを省略すると、実行時のパフォーマンスを高め、DB2 プリコンパイラの制限を超えるリスクを削減できます。ただし、「**isSQLNullable プロパティを省略**」を選択するのは、SQL 列がすべて非 NULL として定義されていることが確実である場合に限る必要があります。いずれかの SQL 列が NULL 可能である場合は、マイグレーション時に「**isSQLNullable プロパティを省略**」を選択解除してください。マイグレーション後に、パフォーマンスを高めたい場合には、SQL レコード内で選択したフィールドの **isSQLNullable** プロパティを除去できます。
- **isReadOnly プロパティを省略**。VisualAge Generator の場合は、SQL レコード内のそれぞれのフィールドごとに、「読み取り専用」プロパティを明示的に設定する必要があります。SQL レコードに対して複数の表が指定されている場合、「読み取り専用」は常に yes にする必要があります。デフォルトでは、マイグレーション・ツールは複数の SQL 表を参照する SQL レコード内の、すべてのフィールドに **isReadOnly** プロパティを組み込みます。ただし、SQL 表がただ 1 つ存在する場合、EGL **isReadOnly** プロパティはデフォルトで NO に設定され、SQL レコードに対して複数の SQL 表が指定されている場合は YES に設定されます。「**isReadOnly プロパティを省略**」を選択した場合は、SQL レコードに対して単一の表が指定されていて、かつ VisualAge Generator の「読み取り専用」プロパティが yes に設定されている場合に限り、マイグレーション・ツールによって **isReadOnly** が組み込まれます。**isReadOnly** プロパティを省略すると、EGL ソース・コードが読みやすくなります。

DL/I 設定:

- **データベース PCB 接尾部**。VisualAge Generator では、同じデータベース名を PSB 内で複数回使用できます。EGL では、PSB はレコードです。データベース名はフィールド名になるものであり、固有でなければなりません。マイグレーション・ツールは、PSB 内のフィールド名をデータベース名、数値 (固有にするた

め必要なとき)、およびデータベース PCB 接尾部から作成します。これによって、データベース名とプログラム内の他の名前が競合しないようにします。例えば、2 つの異なる PCB に対してデータベース名 **COURSE** が使用され、かつ **COURSE** が **DL/I** セグメント・レコード名としても使用されている場合があります。データベース PCB のデフォルト接尾部 **_dbPCB** を使用すると、最初の **COURSE PCB** のフィールド名は **COURSE_dbPCB** になります。2 番目の **COURSE PCB** のフィールド名は **COURSE_n_dbPCB** になります。ここで *n* は **VAGen PSB** 内の **PCB** 数です。**DL/I** セグメント名は **COURSE** のままです。データベース PCB 接尾部は、ブランク以外の任意の値に設定できます。ただし、どのパーツ名とも競合を起こさないものを必ず選択してください。

- **GSAM PCB 接尾部** データベース PCB 接尾部がデータベース PCB に対して使われるのと同じ方法で、**GSAM PCB 接尾部**が **GSAM PCB** に対して使用されます。**GSAM PCB 接尾部**は、ブランク以外の任意の値に設定できます。ただし、どのパーツ名とも競合を起こさないものを必ず選択してください。

VAGen マイグレーション構文の設定

次のセクションで説明する設定によって、**VAGen** 構文から **EGL** 構文へのマイグレーションが制御されます。特に注記がない限り、これらの設定は、ステージ 1 から 3 のマイグレーション中のステージ 2 と、単一ファイル・マイグレーションの両方に使用されます。これらの設定を行うには、「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」->「**VAGen マイグレーション**」->「**VAGen マイグレーション構文の設定**」をクリックします。

名前変更設定:

- **名前変更接頭部 (Renaming prefix)**。マイグレーション・ツールは、**VAGen** データ項目名、レコード名、関数名、またはマップ名がマイグレーション・ツールの拡張予約語リストに存在する場合、あるいは **#** 記号または **@** 記号から始まる場合には、常にこの接頭部を使用します。マイグレーション・ツールは、名前変更接頭部を **VAGen** パーツ名の先頭に追加して、有効な **EGL** パーツ名を作成します。例えば、**"date"** は **EGL** の予約語です。**DATE** という名前の関数がある場合に、デフォルトの名前変更接頭部 **VAGen_** を使用すると、マイグレーション・ツールは関数 **DATE** を **VAGen_DATE** に変更します。また、ツールはこの関数の参照もすべて **DATE** から **VAGen_DATE** に変更します。名前変更接頭部は、ブランク、**EZE**、あるいは **#** 記号または **@** 記号から始まる値以外の任意の値に設定できます。必ず、どのパーツ名とも競合を起こさないものを選択してください。
- **レベル 77 接尾部**。**VAGen** 作業用ストレージ・レコードにレベル 77 項目が含まれている場合に、マイグレーション・ツールはこの接尾部を使用します。**EGL** はレベル 77 項目をサポートしません。マイグレーション・ツールは、**VAGen** 作業用ストレージ・レコードを 2 つの **EGL** レコードに分割します。1 つ目のレコードは、オリジナルの **VAGen** 作業用ストレージ・レコードと同じ名前で、レベル 77 以外の項目をすべて含みます。2 つ目のレコードは、レベル 77 項目をすべて含みます。マイグレーション・ツールは、オリジナルの作業用ストレージ・レコード名にレベル 77 接尾部を付加してこの 2 つ目のレコードの名前を付けます。例えば、オリジナルの作業用ストレージ・レコードの名前が **MYRECORD** で、デフォルトのレベル 77 接尾部 **_Level77Items** を使用する場合、レベル 77 項目を含む **EGL** レコードの名前は **MYRECORD_Level77Items** になります。レベル 77

接尾部は、ブランク以外の任意の値に設定できます。ただし、どのパーツ名とも競合を起こさないものを必ず選択してください。

- **ヘルプ・マップ接尾部。**プログラムのメインのマップ・グループと、ヘルプ・マップ・グループに同名のマップが含まれている場合、マイグレーション・ツールはこの接尾部を使用します。EGL の場合、プログラムの両方の書式グループにある `FormGroups` すべてに固有の名前が付いている必要があります。マイグレーション・ツールは、メインのマップ・グループのマップ名と競合するヘルプ・マップ・グループのマップ名を変更します。次の例を考えてみましょう。プログラムのメインのマップ・グループは `MAPGP1` で、`MAP1` を含んでいます。同じプログラムのヘルプ・マップ・グループは `MAPGP2` で、`MAP1` を含んでいます。デフォルトのヘルプ・マップ接尾部 `_helpmap` を使用した場合、マイグレーション・ツールは `MAPGP2` 内の `MAP1` の名前を `MAP1_helpmap` に変更します。ヘルプ・マップ接尾部は、ブランク以外の任意の値に設定できます。ただし、どのパーツ名とも競合を起こさないものを必ず選択してください。

他の変換オプション:

- **共用データ項目をプリミティブ項目定義に変換。**この設定を選択した場合は、共用データ項目がレコード、テーブル、関数ローカル・ストレージ、関数仮パラメーター・リスト、またはプログラム・パラメーター・リスト内で使用されているときに、マイグレーション・ツールは共用項目をプリミティブ定義に変換します。現在の組織の標準では、新規アプリケーションに共用データ項目を使用することを推奨していない場合は、このオプションを使用すれば、マイグレーション時に既存アプリケーションが共用データ項目を使用しないようにすることができます。**警告:** プリミティブ定義のみがレコードにインクルードされます。したがって、UI レコードの共用項目を使用する場合、共用項目の UI 編集カスタマイズが EGL VGUI レコードにコピーされないため、この設定は選択解除してください。
- **10 進数のコンマを小数点に変更。**ご使用のワークステーションが、小数点を示すために通常はコンマを使用するロケールを指定している場合、マイグレーション・ツールは小数点を使用するように数値リテラルを自動的に変換します。例えば、フランス語およびドイツ語のロケールの場合、マイグレーション・ツールはコンマを小数点に自動的に変換します。ただし、お客様によっては、ロケールとして英語を指定し、通常的小数点設定をコンマに指定変更している場合もあります。この手法を使用している場合は、「**10 進数のコンマを小数点に変更**」の設定を選択する必要があります。これにより、マイグレーション・ツールは数値リテラルのコンマを小数点に変換します。EGL は、内部ストレージの小数点位の標識として点のみをサポートします。
- **VAGen 論理演算子 (AND および OR) の使用。**この設定を選択した場合、マイグレーション・ツールは、EGL `if` ステートメントおよび `while` ステートメントの論理演算子として「`and`」および「`or`」を使用します。この設定をクリアすると、マイグレーション・ツールは、論理演算子として `&&` および `||` を使用します。新規 EGL プログラムを作成する場合、使用するコーディング標準に基づいてこの設定を行います。
- **CALL および DXFR プログラム名を引用符で囲む。**この設定を選択すると、マイグレーション・ツールは、EGL `call` ステートメントまたは `transfer to program` ステートメントで使用する場合には常に、プログラム名を引用符で囲み

ます。この設定をクリアすると、マイグレーション・ツールは、プログラム名を引用符で囲みません。この設定に関係なく、マイグレーション・ツールは次の処理を行います。

- **sysVar.transferName** (EZEAPP) を引用符で囲みません。
- **isExternal** プロパティが yes (NONCSP) に設定されている場合は、プログラム名を引用符で常に囲みます。これには、EZCHART の使用が含まれます。

この設定を選択する利点は次のとおりです。

- ご使用のワークスペースに存在しないプログラムを参照する **call** ステートメントおよび **transfer to program** ステートメントがいずれもエラーと見なされないため、そのプログラムを生成することができます。
- ご使用のワークスペースでこの名前を持つプログラムが 1 つのみ存在する場合、呼び出しまたは転送プログラムをデバッグできます。

この設定を選択する欠点は、エラー・メッセージを受け取れないため、Java 用のプログラムを生成する場合に、パッケージ情報を提供するリンケージ・オプション・パーツを用意する必要があることが指摘されない点です。

- **XFER** プログラム名を引用符で囲む。この設定を選択すると、マイグレーション・ツールは、EGL **transfer to transaction** ステートメントおよび **show** ステートメントのプログラム名を引用符で囲みます。この設定をクリアすると、マイグレーション・ツールは、プログラム名を引用符で囲みません。この設定に関係なく、マイグレーション・ツールは次の特殊処理を行います。

- **sysVar.transferName** (EZEAPP) を引用符で囲みません。
- **isExternal** プロパティが yes (NONCSP) に設定されている場合は、プログラム名を引用符で常に囲みます。

この設定を選択する利点は次のとおりです。

- ご使用のワークスペースに存在しないプログラムを参照する **transfer to transaction** ステートメントおよび **show** ステートメントはいずれもエラーと見なされません。
- **transfer-to** の名前がプログラムになるランタイム環境では、ご使用のワークスペースでこの名前を持つプログラムが 1 つのみ存在する場合、転送プログラムをデバッグできます。**transfer-to** の名前がトランザクション名になるトランザクションのランタイム環境 (例えば、CICS または IMS/VS) では、VisualAge Generator で行うように、デバッグ時にはプログラム名を使用し、実行時にはトランザクション名を使用して、デバッグを行えます。

この設定を選択する欠点は、エラー・メッセージを受け取れないため、Java 用のプログラムを生成する場合に、パッケージ情報を提供するリンケージ・オプション・パーツを用意する必要があることが指摘されない点です。

その他の推奨設定

次の設定を検討することができます。

- (オプション)「ようこそ」ビューを閉じる。
- まだ EGL パースペクティブを使用していない場合は、「ウィンドウ」->「パースペクティブを開く」->「その他 (Other)」->「EGL」->「OK」とクリックし

て、このパースペクティブに切り替えます。また、VAGen Web トランザクション・プログラムをマイグレーションするか、または新規の EGL JSFHandler を開発する場合は、Web パースペクティブに切り替える必要があります。ウィンドウの右上隅のタブにあるパースペクティブのアイコンを右クリックし、「閉じる (Close)」をクリックすると、他のパースペクティブを閉じることができます。

- 使用しているパースペクティブに「ナビゲーター」ビューがまだ組み込まれていない場合は、このビューを追加できます。「ナビゲーター」ビューを追加するには、「ウィンドウ」->「ビューの表示」->「その他 (Other)」をクリックします。「ビューの表示」ウィンドウから、「一般」を展開し、「ナビゲーター」を選択します。

注: エラー・マーカの表示や EGL パッケージの削除など、一部のアクティビティは「ナビゲーター」ビューで完全にはサポートされません。プロジェクトまたはパッケージの再構築を行う場合は、「プロジェクト・エクスプローラー」ビューを使用してください。

- 「問題」ビューで、右上隅の「メニュー (Menu)」ボタン (下向き三角形) をクリックしてから「コンテンツの構成」をクリックします。「コンテンツの構成」ウィンドウで、「選択された要素のみ (On selected element only)」ラジオ・ボタンをクリックします。これにより、「問題」ビューのエラー・メッセージが、現在選択されているファイルに関するメッセージのみに制限されます。このようにすると、非常に多くのエラーが発生したときに、一度に 1 つずつのファイルに注意を集中することができます。「問題」ビューのタイトル・バーには、フィルターにマッチングしたメッセージの数と、ワークスペース内のプロジェクトすべてに関するメッセージの合計数が表示されます。
- 「問題」ビューで、「メニュー」ボタンをクリックしてから「設定」をクリックします。「設定」ウィンドウで、「マーカ使用の制限」オプションをクリアします。このオプションをクリアすると、「問題」ビューにメッセージがすべて表示されます。
- 編集のために複数のファイルを開いているとき、「プロジェクト・エクスプローラー」ビューまたは「ナビゲーター」ビューでオープン・ファイルを選択するたびに、そのオープン・ファイルが自動的に前景に配置される (そのエディター・セッションをアクティブ・エディターにする) ように「プロジェクト・エクスプローラー」ビューまたは「ナビゲーター」ビューを構成できます。このためには、「プロジェクト・エクスプローラー」ビューまたは「ナビゲーター」ビューのツールバーにある「開いているエディターとナビゲーター内のコンテンツをリンク」アイコンをクリックします。

ステージ 2 VAGen マイグレーション・ファイルの設定

マイグレーションのステージ 2 を実行するツールは、ウィザードから呼び出すことができます。ステージ 2 マイグレーション設定を作成するには、次のステップを実行します。

1. EGL 開発環境を開始する。必ず、207 ページの『ワークベンチの設定』セクションの説明どおりにワークベンチを設定してください。
2. ステージ 2 ウィザードから、データベース・ドライバーのロケーションを尋ねられます。次のステップを実行すると、この値を格納するクラスパス変数を設定して、この値がウィザードに自動的に取り込まれるようにできます。

- a. 「ウィンドウ」->「設定」から、「Java」->「ビルド・パス」->「クラスパス変数」をクリックする。
 - b. 「新規」をクリックします。
 - c. 「名前」に **DB2_DRIVER_PATH** と入力します。
 - d. 「パス」に対しては、「ファイル」をクリックし、db2java.zip ファイルにナビゲートする。(これは、ステージ 1 で使用した db2java.zip ファイルと同じです。デフォルトでは、ファイルは %SQLLIB%java\db2java.zip にあります。)
 - e. 「新規変数エントリー」ウィンドウの「OK」をクリックし、「設定」ウィンドウで「OK」をクリックする。
3. (オプション) ステージ 2 設定ファイルを保管する場合にファイルを格納できる、シンプルなプロジェクトを作成します。これは、ステージ 2 をバッチ・モードで実行する場合に必要です。ステージ 2 をオンライン・モードで実行する場合でも、設定の記録が保管されていると役立ちます。シンプルなプロジェクトを作成するには、次のステップを実行してください。
 - a. 「ファイル」->「新規」->「プロジェクト」とクリックして、「新規プロジェクト」ウィザードを開始します。「一般」を展開し、「プロジェクト」を選択して、「次へ」をクリックします。
 - b. VAGENMIG など、プロジェクトに名前を付けます。「終了」をクリックします。
 4. プロジェクトを右クリックし、「新規」->「その他」を選択します。
 5. 「EGL への VAGen マイグレーション」を展開し、「VAGen マイグレーション・ファイル (VAGen Migration File)」を選択する。「次へ」をクリックします。
 6. 以下の適切なステージ 2 設定を入力する。
 - a. ウィザードの先頭ページで、次の表に説明するように設定を編集する。Tab キーを押してこのフィールドから出るまで、マイグレーション・ツールによる「データベース情報」フィールドの検証は行われません。これにより、情報の入力中にデータベースへの接続が繰り返し試行されることを防止できます。

表 63. ウィザードの先頭ページに入力する設定

設定	意味	値
既存ファイルをロード	この設定により、前に保管したステージ 2 設定ファイルを選択できます。「ファイルを選択」をクリックして、既存の .vgmig ファイルを選択します。「ロード・ファイル」をクリックすると、そのファイルから設定が取り込まれ、ウィザードに表示されます。	(オプション) 既存の .vgmig ファイルを選択してロードします。
データベース・ドライバのロケーション	これは、DB2 ドライバのロケーションです。	path_to_db2java.zip\db2java.zip
データベース・ドライバ	これは、DB2 ドライバの名前です。	この値は、常に COM.ibm.db2.jdbc.app.DB2Driver にする必要があります。この値は、ローカル・データベースと、ローカル側でカタログされたりモート・データベースの両方に有効です。

表 63. ウィザードの先頭ページに入力する設定 (続き)

設定	意味	値
データベース名	これは、マイグレーションのステージ 1 で使用した DB2 データベースの名前です。	この値は、次のフォーマットで指定する必要があります。 <ul style="list-style-type: none"> jdbc:DB2:databaseName databaseName は、マイグレーションのステージ 1 で使用した DB2 データベースの名前です。ステージ 1 のデフォルト値は VGMIG です。
データベース・スキーマ (Database schema)	これは、マイグレーションのステージ 1 で使用した DB2 データベース・スキーマの名前です。	この値は、マイグレーションのステージ 1 で使用した DB2 スキーマの名前です。ステージ 1 のデフォルト値は MIGSCHEMA です。
データベース・ユーザー ID	これは、マイグレーションのステージ 1 で使用したデータベース・ユーザー ID です。	ステージ 1 に使用したものと同一データベース・ユーザー ID を使用します (デフォルト値は、ご使用のログオン ID)。
データベース・パスワード	これは、マイグレーションのステージ 1 で使用したデータベース・パスワードです。	ステージ 1 に使用したものと同一データベース・パスワードを使用します (デフォルト値は、ご使用のログオン・パスワード)。
ログ・ファイルのロケーション	これは、ステージ 2 メッセージのログ・ファイルのロケーションです。	ファイル・システム内の有効なロケーション (ドライブとディレクトリー) を入力します。
ログ・ファイル名	これは、ステージ 2 メッセージのログ・ファイルの名前です。	有効なファイル名を入力します。

b. ウィザードの 2 ページ目で、次の表の説明のとおり、設定を編集する。

表 64. ウィザードの 2 ページ目に入力する設定

設定	意味	値
「Java」または「COBOL」ラジオ・ボタン	この選択により、マイグレーション・ツールが Java ソース・フォルダーを含むプロジェクトを作成するかどうかが決まります。	COBOL のみを生成する場合は、 COBOL をクリックします。Java を生成する場合は、 Java をクリックします。
残りの VAGen パーツをマイグレーションする	この設定により、マイグレーション・セット内の生成可能パーツによって参照されないパーツを EGL に変換するかどうかが決まります。 注: 残りの VAGen パーツをマイグレーションする 設定については、UI レコードは他のレコードと同じように扱われます。この設定では、UI レコードを生成可能パーツとはみなされません。	参照されないパーツを EGL に変換する場合は、チェック・ボックスを選択します。通常は、「残りのパーツをマイグレーションする」を選択する必要があります。「残りのパーツをマイグレーションする」を選択解除した場合、マイグレーション・セット内で使用されない制御パーツやその他のパーツは、EGL ソースにマイグレーションされません。 このオプションを選択した場合の影響については詳しくは、58 ページの『ファイルの上書きとマージ』を参照してください。

表 64. ウィザードの 2 ページ目に入力する設定 (続き)

設定	意味	値
ワークスペースにインポート	<p>この設定により、ステージ 2 の完了後に、ステージ 3 (現行ワークスペース内のファイルに EGL をインポートする) を自動的に開始するかどうかが決まります。</p> <p>注: このチェック・ボックスを選択する場合は、ボックスの下にあるラジオ・ボタンの 1 つをクリックして、インポートするバージョン (最新または最も古いバージョン。この表の次の行を参照) を指定する必要があります。これは、一度にワークスペースに存在できるプロジェクトのバージョンが 1 つのみであるからです。</p>	<p>パーツを EGL に変換した直後に EGL ファイルをインポートする場合は、このチェック・ボックスを選択します。ファイルを後で (ステップ 3 で) インポートする場合は、このチェック・ボックスをクリアします。</p> <p>注: このオプションを選択した場合、ステージ 3 を別個に実行する必要はありません。マイグレーション・ツールは、ステージ 2 (変換) の直後にステージ 3 (インポート) を自動的に開始し、マイグレーション・プロセスを完了します。</p> <p>このオプションを選択した場合の影響については、58 ページの『ファイルの上書きとマージ』を参照してください。</p>
「最新バージョン」または「最も古いバージョン」	<p>このオプションでは、対象のマイグレーション・セットのどのバージョンをインポートするかを指定します。</p>	<p>「ワークスペースにインポート」オプションが選択されている場合、これらのラジオ・ボタンの 1 つをクリックする必要があります。</p>
既存ファイルを上書き	<p>ステージ 3 (インポート・プロセス) では、ステージ 2 で作成した EGL を使用して、ステージ 1 レポートの中で指定された EGL ファイルを作成し、インポートします。ステージ 3 でインポートしようとしている EGL ファイルと同名の EGL ファイルがワークスペースにすでに存在する場合は、このオプションによってこれらのファイルを上書きするかどうかが決まります。</p>	<p>このオプションは、「ワークスペースにインポート」オプションを選択した場合にのみ選択できます。「既存ファイルを上書き」オプションを選択すると、現在マイグレーション中のマイグレーション・セットに、ワークスペース内の既存ファイルに配置する必要があるパーツが含まれている場合に、ステージ 3 マイグレーション・ツールがその状況をどのように処理するかを指定できます。「既存ファイルを上書き」オプションを選択すると、ステージ 3 マイグレーション・ツールは既存のファイルを置き換え、現行マイグレーション・セット内にあるパーツのみを組み込みます。「既存ファイルを上書き」オプションを選択解除した場合、ステージ 3 マイグレーション・ツールは新規パーツすべてを既存ファイルにマージします。新規パーツは、パーツ型別にアルファベット順に配置されます。</p> <p>このオプションを選択した場合の影響については、58 ページの『ファイルの上書きとマージ』を参照してください。</p>

表 64. ウィザードの 2 ページ目に入力する設定 (続き)

設定	意味	値
マイグレーションしたファイルを一時ディレクトリに保管	このオプションを選択すると、EGL ファイルをファイル・システム内のロケーションに保管できます。これにより、プロジェクトの複数バージョン用の EGL ファイルに同時にアクセスできます (ただし、ワークスペース内で一度に表示できるバージョンはただ 1 つ)。このロケーションから、リポジトリに EGL ファイルを直接移動できます。	<p>マイグレーション・セットの複数バージョンをマイグレーションする予定の場合は、次のステップを実行します。</p> <ol style="list-style-type: none"> それぞれのバージョンを別々のサブディレクトリに書き込むことができるように、このチェック・ボックスを選択します。 これらのバージョンを配置するサブディレクトリを格納する「フォルダー」を指定します。 「今すぐマイグレーションを実行」オプションを選択解除する。リソース所要量が大きいので、一時ディレクトリへのマイグレーションはバッチ・モードでのみ行う必要があります。「今すぐマイグレーションを実行」を選択した場合は、本当にオンライン・モードで実行するかどうか、マイグレーション・ツールから確認を求められます。 「現行構成をファイルに保管」オプションを選択します。また、現行構成を .vgmig ファイルとして保管する先のプロジェクトとファイル名も指定する必要があります。オンラインまたはバッチのどちらのモードでマイグレーションを行うかに関係なく、「マイグレーションしたファイルを一時ディレクトリに保管」オプションを選択した場合は .vgmig ファイルが必要です。バッチ・モードでステージ 2 を実行する場合は、マイグレーション設定を指定するために、保管した .vgmig ファイルを指示します。
フォルダー (Folder)	これは、EGL ファイルを保管するディレクトリです。「 フォルダー (Folder) 」に指定したディレクトリの下に、それぞれのマイグレーション・セット・バージョンのサブディレクトリが作成されます。	ファイル・システム内の既存ディレクトリを指定します。
すぐにマイグレーションを実行	後でマイグレーションを行うために設定ファイルをセットアップするだけでなく、この時点でステージ 2 を実行するように指定します。	<p>「今すぐマイグレーションを実行」オプションを選択してステージ 2 をオンライン・モードで実行するか、または後でステージ 2 をバッチ・モードで実行できるように、「現行構成をファイルに保管」オプションを選択して設定を保管する必要があります。後で参照できるように設定のコピーを保持したい場合は、両方のオプションを選択できます。「マイグレーションしたファイルを一時ディレクトリに保管」をすでに選択した場合は、「今すぐマイグレーションを実行」を選択解除してください。一時ディレクトリへの保管は、バッチ・モードでのみ実行できます。「今すぐマイグレーションを実行」の選択は、オンライン・モードでマイグレーションを行うように指示します。</p>

表 64. ウィザードの 2 ページ目に入力する設定 (続き)

設定	意味	値
現行構成をファイルに保管	<p>この設定により、指定中の設定をファイルに保管できます。後で、次のどちらかの方法でステージ 2 を実行できます。</p> <ul style="list-style-type: none"> オンライン・モードの場合は、保管した .vgmig ファイルを右クリックし、「マイグレーションの開始 (Start Migration)」をクリックする。 バッチ・モードの場合は、-importFile オプションを使用して、保管した .vgmig ファイルを指定する。詳しくは、225 ページの『バッチ・モードでのステージ 2 の実行』を参照してください。 	<p>「今すぐマイグレーションを実行」オプションを選択してステージ 2 をオンライン・モードで実行するか、または後でステージ 2 を実行できるように、「現行構成をファイルに保管」オプションを選択して設定を保管する必要があります。後で参照できるように設定のコピーを保持したい場合は、両方のオプションを選択できます。</p> <p>このオプションを選択した場合は、現行構成を .vgmig ファイルとして保管する先のパスとファイル名も指定する必要があります。後でステージ 2 を実行する場合は、マイグレーション設定を指定するために、保管した .vgmig ファイルを指示します。</p>
パス	ファイルを保管する先のプロジェクトを指定します。	<code>¥projectName</code> 。ここで、 <code>projectName</code> は保管するファイルに格納するプロジェクトの名前です。
ファイル名	設定を保管する先のファイルの名前を指定します。	<code>fileName</code> 。ここで、 <code>fileName</code> は目的のファイルの名前で、ファイル拡張子を付けずに 指定します。拡張子 .vgmig が自動的に付加されます。

- c. ウィザードの 3 ページ目では、指定したデータベース内にあるマイグレーション・セットがウィザードによってリストされます。マイグレーションするマイグレーション・セットを選択します。マイグレーション・セットを選択しない場合、マイグレーション・ツールはデータベース内にあるすべてのマイグレーション・セットをマイグレーションします。「終了」をクリックします。

指定したオプションの組み合わせによって、ウィザードが実行するアクションが決まります。

- 「**現行構成をファイルに保管**」を選択した場合は、「終了」をクリックした後、すべてのオプションが指定したファイルに保管されます。
- 「**今すぐマイグレーションを実行**」を選択した場合は、「終了」をクリックした後、ステージ 2 マイグレーションが実行されます。
- また、「**ワークスペースにインポート**」または「**マイグレーションしたファイルを一時ディレクトリに保管**」のどちらかを選択した場合は、ステージ 2 の完了後にステージ 3 が自動的に開始されます。

次に、ステージ 2 設定ファイル stage2.vgmig の例を示します。

```

databaseDriverLocation=d:\SQLLIB\java\db2java.zip
databaseDriver=COM.ibm.db2.jdbc.app.DB2Driver
databaseName=jdbc:DB2:VGMIG
databaseSchema=MIGSCHEMA
databaseUserId=myuserId
databasePassword=encoded:AAEDAwQFBwYKCwo+Pw==
configurationPlan=MyMigrationSetA,1.1
migrateRemainingParts=yes

```



```
workspaceImport=latest
overrideExistingFiles=yes
tempDirectory=
logFileLocation=D:\tempmig\MyMigrationSetA\stage2\MyMigrationSetA.log
migrateNow=yes
projectType=COBOL
```

ステージ 2 の実行

ステージ 2 マイグレーション・ツールは、バッチ・モードで実行することも、EGL 開発環境のユーザー・インターフェースから実行することもできます。

ユーザー・インターフェースからのステージ 2 の実行

218 ページの『ステージ 2 VAGen マイグレーション・ファイルの設定』で説明したウィザードには、.vgmig ファイルに設定を保管するオプションがあります。ウィザードの「**今すぐマイグレーションを実行**」オプションを選択すると、ウィザードの「**終了**」をクリックしたときにステージ 2 マイグレーションが開始されます。ウィザードの「**今すぐマイグレーションを実行**」を選択解除した場合は、保管した .vgmig ファイルを使用してステージ 2 マイグレーションを実行する準備ができたなら、次のステップを実行します。

1. 「プロジェクト・エクスプローラー」ビューまたは「ナビゲーター」ビューで、プロジェクト名の隣にある「+」記号をクリックして、ステージ 2 設定ファイルを含むプロジェクトを展開します。
2. .vgmig 設定ファイルを右クリックして、「**マイグレーションの開始**」をクリックします。

ステージ 2 マイグレーションが開始され、指定したマイグレーション・セットの外部ソース形式が EGL ソースに変換されて、EGL ソースがマイグレーション・データベースに格納されます。「**ワークスペースにインポート**」または「**マイグレーションしたファイルを一時ディレクトリに保管**」のどちらかを選択した場合は、ステージ 2 の完了後にステージ 3 が自動的に開始されます。ステージ 3 の後、マイグレーション・ツールはワークスペースのリフレッシュを自動的に開始します。特にパーツ数が多い場合は、リフレッシュ・ステップに時間がかかる可能性があります。リフレッシュ・ステップが完了すると、マイグレーションが完了したことを通知するポップアップ・ウィンドウが表示されます。必ず、ポップアップ・ウィンドウが表示されるまで待ってください。

マイグレーションとリフレッシュ・ステップが完了すると、次の出力が使用可能になります。

- ステージ 2 マイグレーション・ログ・ファイル。このログ・ファイルは、「**ログ・ファイルのロケーション**」として指定したディレクトリにあります。このログ・ファイルには、マイグレーションされたパーツに関する情報と、ステージ 2 マイグレーション中に出された情報メッセージ、警告メッセージ、またはエラー・メッセージが保管されています。
- マイグレーション・セットの「to do」リスト・ログ・ファイル。この「to do」リスト・ファイルは、ステージ 3 の開始時に作成される、ステージ 2 で生成されたメッセージを統合したリストで、マイグレーションを完了するために必要になる可能性がある追加作業を示しています。それぞれの生成可能パーツとその関連パーツに関するメッセージが、グループとしてリストされているという点では、

この「to do」リストは VisualAge Generator 生成メッセージと似ています。あるパーツに対してメッセージが存在し、そのパーツが複数のプログラムに関連している場合、これらのメッセージはそれぞれのプログラムごとに 1 回リストされます。未使用の生成不可パーツに関するメッセージは、プロジェクト、パッケージ、およびファイル名別に、「to do」リストの末尾にリストされます。この点では、この「to do」リストは VisualAge Generator 生成メッセージと異なっています。「to do」リストは、ステージ 2 マイグレーション・ログ・ファイルと同じディレクトリに置かれます。マイグレーション・ツールは、処理するそれぞれのマイグレーション・セットのバージョンごとに、「to do」リスト・ファイルを 1 つずつ作成します。「to do」リスト・ファイル名の形式は、`TODO_migrationSetName_migrationSetVersion` です。

- 「ワークスペースにインポート」オプションを選択した場合、ステージ 3 が自動的に開始され、マイグレーション・セットに必要な EGL プロジェクト、ソース・フォルダー、パッケージ、および EGL ファイルが作成されます。またステージ 3 ツールは、プロジェクトをワークスペースにインポートし、EGL 検証を実行するためにプロジェクトを再ビルドします。
- 「マイグレーションしたファイルを一時ディレクトリに保管」オプションを選択した場合、ステージ 3 が自動的に開始され、マイグレーション・セットに必要な EGL プロジェクト、ソース・フォルダー、パッケージ、および EGL ファイルが作成されます。ステージ 3 ツールは、それぞれのマイグレーション・セットのバージョンごとにプロジェクトを、指定した一時ディレクトリの下にある別々のサブディレクトリに配置します。サブディレクトリ名の形式は、`migrationSetName_migrationSetVersion` です。これにより、プロジェクトの複数のバージョンを一度にマイグレーションして、後でワークスペースにインポートできます。

バッチ・モードでのステージ 2 の実行

ステージ 2 ウィザードを使用して、1 つ以上のマイグレーション・セットを選択し、即時にマイグレーションできます。また、後でバッチ・モードでのマイグレーションを行うために、ファイルに情報を保管することもできます。バッチ・モードを使用する場合は、次の制限事項を検討する必要があります。

- ご使用の EGL 設定に関係なく、`.egldd` ファイルは作成されません。
- EGL プロジェクト、パッケージ、およびファイルは、現行ワークスペースまたは一時ディレクトリのどちらかに送信することができます。現行ワークスペースおよび一時ディレクトリの両方を指定した場合は、マイグレーション・ツールは現行ワークスペースを無視します。
- 一時ディレクトリに作成する場合は、次のスキームと同様にディレクトリとワークスペースの構造を作成する必要があります。

```
temporaryDirectoryName << temporaryDirectoryName can be any name
stub                  << "stub" is the required workspace name
    projectName       << projectName can be any name
        xxxx.vgmig    << xxxx can be any name
```

次の方法で stub ワークスペースをセットアップします。

- 207 ページの『ワークベンチの設定』で説明されているとおりに、EGL および VAGen のマイグレーション設定を指定します。

- stub ワークスペースが配置されている *temporaryDirectoryName* を指すように *xxx.vgmig* ファイルの **tempDirectory** オプションを設定します。マイグレーション・ツールは、ステージ 3 の実行中に、それぞれのマイグレーション・セットのバージョンごとに *temporaryDirectoryName* の下にワークスペースを 1 つずつ作成します。マイグレーション・ツールは、stub ワークスペースからそれぞれの新規ワークスペースに、機能、設定、プロジェクトなどをすべてコピーします。
- *temporaryDirectoryName*¥stub を指すようにバッチ・コマンド・ファイルの **-data** オプションを設定します。

バッチ・モードで実行するには、次のステップを実行します。

1. 218 ページの『ステージ 2 VAGen マイグレーション・ファイルの設定』で説明されているステップに従いますが、以下の点が異なります。
 - ・「**現行構成をファイルに保管**」を選択し、パスとファイル名を指定する。自動的に作成されるファイルには接尾部 *.vgmig* が付けられ、バッチ・モードでステージ 2 またはステージ 3 を実行するときには、このファイルを **-importFile** として指定する必要があります。
 - ・「**今すぐマイグレーションを実行**」を必ずクリアします。このオプションをクリアすると、後でマイグレーションを行うための情報が保管されます。
 - ・複数の *.vgmig* ファイルを定義して、後で単一のバッチとしてマイグレーションできます。
2. *.bat* ファイル拡張子の付いたファイルを作成する。

Windows 環境の場合、*.bat* ファイルの内容は次のようにする必要があります。

```
set INSTALL_PATH=InstallDirectory
set SHARED_INSTALL_PATH=SharedInstallDirectory
set JDK_PATH=jdk\jre\bin
set PLUGIN_PATH=plugins
set MIG_JAR=com.ibm.etools.egl.vagenmigration_version.jar
set STARTUP_JAR=org.eclipse.equinox.launcher_version.jar
set path=%INSTALL_PATH%\%JDK_PATH%;%path%
set classpath=%SHARED_INSTALL_PATH%\%PLUGIN_PATH%\%MIG_JAR%;
               %INSTALL_PATH%\%PLUGIN_PATH%\%STARTUP_JAR%
cd InstallDirectory
java com.ibm.etools.egl.internal.vagenmigration.batch.VGMIG
  -importFile Path\vgmigFileName.vgmig
  -data Path\workspace
  >Path\LogName.log
```

注:

- ・次のステートメントは、ステートメント全体が 1 行になるように入力する必要があります。
 - Windows 環境の場合は、**set classpath** ステートメント。
 - **java** ステートメント。
- ・バッチ・モードでマイグレーションするそれぞれの *.vgmig* ファイルごとに、**java** ステートメントを 1 つずつ繰り返します。ただし、*.vgmig* ファイルを作成したときに「**ワークスペースにインポート**」を選択した場合は、*.vgmig* ファイルから得られる EGL プロジェクト名が同じにならないようにしてください。同じ *.bat* ファイル内で同じ EGL プロジェクトに

関する複数の .vgmig ファイルをマイグレーションしようとする、最後に マイグレーションされる .vgmig ファイルのみが EGL プロジェクトに反映されます。

- *InstallDirectory* は、EGL デベロッパー製品をインストールしたドライブとディレクトリーです。set INSTALL_PATH ステートメントおよび cd (ディレクトリーの変更) ステートメントに、*InstallDirectory* を組み込む必要があります。
- *SharedInstallDirectory* は、EGL デベロッパー製品の共用リソースがインストールされたドライブとディレクトリーです。set SHARED_INSTALL_PATH ステートメントに *SharedInstallDirectory* を組み込む必要があります。

注: 現在ご使用の製品をインストールする前に、前のバージョンの EGL デベロッパー製品をインストールし、保持していた場合、対象になるインストール・ディレクトリーまたは共用インストール・ディレクトリーは、以前のインストール時に使用されたディレクトリーである可能性があります。

- *version* は、プラグイン・バージョン番号です。バージョン番号を判別するには、以下のプラグイン・ディレクトリーを確認します。
 - MIG_JAR プラグインは、*SharedInstallDirectory\plugins* にあります。バージョン番号は、7.5.0.RFB_20080811_1638 などです。
 - STARTUP_JAR プラグインは、*InstallDirectory\plugins* にあります。バージョン番号は、1.0.100.v20080509-1800 などです。

通常は、対応するプラグイン・ディレクトリーの中の .jar ファイルに付けられている、最も大きいバージョン番号を使用する必要があります。

- *Path¥vgmigFileName.vgmig* は、マイグレーション・データベースからマイグレーションするマイグレーション・セットを指定する、.vgmig ファイルのドライブ、ディレクトリー、およびファイル名を示します。このディレクトリーにはワークスペース名が含まれている必要があります。これは、ステップ 1 で保管した .vgmig ファイルです。(例:
d:¥myworkspace¥mySimpleProject¥myMigrationInformation.vgmig)
- *Path¥workspace* は、EGL および VAGen マイグレーション設定を行ったドライブ名、ディレクトリー名、およびワークスペース名です。詳しくは、207 ページの『ワークベンチの設定』を参照してください。また、マイグレーション・ツールは、次のように *path¥workspace* を使用します。
 - ご使用の .vgmig ファイルにより、ステージ 3 の出力が現行ディレクトリーに書き込まれるように指定されている場合、*path¥workspace* は、マイグレーション・ツールが EGL ファイルを配置するドライブ名、ディレクトリー名、およびワークスペース名です。この場合ワークスペース名は、例えば、d:¥mypath¥myworkspace など任意の名前です。
 - ご使用の .vgmig ファイルにより、ステージ 3 の出力が一時ディレクトリーに書き込まれるように指定されている場合、*path* は、マイグレーション・ツールが .vgmig ファイルによって指定されるそれぞれのマイグレーション・セットのバージョンごとにワークスペースを 1 つずつ配置するドライブおよびディレクトリーです。ワークスペース名は stub である必要があります。

- *Path\LogName.log* は、java コマンドに対して作成するログ・ファイルのドライブ、ディレクトリー、およびファイル名を指定します。このログ・ファイルは、java コマンド自体の問題をリストします。ステージ 2 またはステージ 3 で生成されるログ・メッセージは、マイグレーション・ウィザードの先頭ページで指定したログ・ファイルに格納され、その後 .vgmig ファイルに保管されます。同じ .bat ファイルに複数の java コマンドがある場合は、それぞれの java コマンドごとに異なるログ・ファイル名を必ず指定してください。

Windows 環境の場合、**java** コマンドの例は次のようなものです (ただし、コマンド全体を 1 つの行に入力する必要があります)。

```
java com.ibm.etools.egl.internal.vagenmigration.batch.VGMIG
-importFile d:\myTempDirectory\stub\myProject\myMigrationInfo.vgmig
-data d:\myTempDirectory\stub\
>d:\migrationLogs\myMigrationInformationPiped.log
```

3. EGL 開発環境をシャットダウンする。
4. コマンド・プロンプト・ウィンドウを開き、.bat ファイルがあるディレクトリーにナビゲートして、.bat ファイルを実行する。

注: 次のメッセージは無視して構いません。

PolicyClassLoader は、ポリシー com.ibm.jxesupport.JxeClassLoaderPolicy を見つけることができませんでした (PolicyClassLoader could not find policy com.ibm.jxesupport.JxeClassLoaderPolicy)

5. 処理が完了すると、指定されたディレクトリーに EGL プロジェクト、ソース・フォルダー、パッケージ、およびファイルが保管されます。それぞれの **java** コマンドに対応するログ・ファイルには、マイグレーション済みパーツのリストと、エラー・メッセージが保管されています。メッセージは、ユーザー・インターフェースを使用してステージ 2 を実行した場合にログ・ファイルに書き込まれるメッセージと同じです。同様に、「to do」リスト・ファイルに保管されるメッセージは、ユーザー・インターフェースを使用してステージ 2 を実行した場合にこのファイルに書き込まれるメッセージと同じです。
6. EGL 開発環境を開始する。
7. 「ワークスペースにインポート」オプションを選択した場合、EGL プロジェクト、ソース・フォルダー、パッケージ、およびファイルがワークスペースに表示されます。

第 7 章 ステージ 3 - インポート

マイグレーションのステージ 3 も、EGL に付属のプラグインを使用して実行します。このステージでは、別のマイグレーション・ツールを実行して、ステージ 2 でマイグレーション・データベースに格納された EGL 構文から EGL ファイルを作成します。

ステージ 3 ツールの実行

ステージ 3 ツールは、以下の 2 つの方法で実行できます。

- ステージ 2 ウィザードでオプションを設定するとき（218 ページの『ステージ 2 VAGen マイグレーション・ファイルの設定』で説明）「ワークスペースにインポート」または「マイグレーションしたファイルを一時ディレクトリーに保管」オプションを選択すると、ステージ 2 が終了するとステージ 3 が自動的に開始されます。通常、これは、自動的に開始されるので、ステージ 3 を実行する最も簡単な方法です。
- ステージ 3 を別個の手順として実行します。EGL 開発環境のワークベンチ・ウィンドウで以下の手順を実行します。
 1. 「ファイル」->「インポート」をクリックします。
 2. 「その他」を展開し、「データベースからの VAGen マイグレーション」を選択します。「次へ」をクリックします。
 3. このマイグレーション・ステージの設定を次のように指定する。
 - a. ウィザードの先頭ページで、次の表に説明するように設定を編集する。Tab キーを押してこのフィールドから出るまで、マイグレーション・ツールによる「データベース情報」フィールドの検証は行われません。これにより、情報の入力中にデータベースへの接続が繰り返し試行されることを防止できます。

表 65. ウィザードの先頭ページに入力する設定

設定	意味	値
既存ファイルをロード	この設定により、前に保管したステージ 3 設定ファイルを選択できます。「ファイルを選択」をクリックして、既存の .vgmig ファイルを選択します。「ロード・ファイル」をクリックすると、そのファイルから設定が取り込まれ、ウィザードに表示されます。	(オプション) 既存の .vgmig ファイルを選択してロードします。
データベース・ドライバーのロケーション	これは、DB2 ドライバーのロケーションです。	<code>path_to_db2java.zip¥db2java.zip</code>
データベース・ドライバー	これは、DB2 ドライバーの名前です。	この値は、常に <code>COM.ibm.db2.jdbc.app.DB2Driver</code> にする必要があります。この値は、ローカル・データベースと、ローカル側でカタログされたりリモート・データベースの両方に有効です。

表 65. ウィザードの先頭ページに入力する設定 (続き)

設定	意味	値
データベース名	これは、マイグレーションのステージ 1 で使用した DB2 データベースの名前です。	この値は、次のフォーマットで指定する必要があります。 <ul style="list-style-type: none"> jdbc:DB2:databaseName databaseName は、ステージ 1 で使用した DB2 データベースの名前です。ステージ 1 のデフォルト値は VGMIG です。
データベース・スキーマ (Database schema)	これは、マイグレーションのステージ 1 で使用した DB2 データベース・スキーマの名前です。	この値は、ステージ 1 で使用した DB2 スキーマの名前です。ステージ 1 のデフォルト値は MIGSCHEMA です。
データベース・ユーザー ID	これは、マイグレーションのステージ 1 で使用したデータベース・ユーザー ID です。	ステージ 1 に使用したのと同じデータベース・ユーザー ID を使用します (デフォルト値は、ご使用のログオン ID)。
データベース・パスワード	これは、マイグレーションのステージ 1 で使用したデータベース・パスワードです。	ステージ 1 に使用したのと同じデータベース・パスワードを使用します (デフォルト値は、ご使用のログオン・パスワード)。
ログ・ファイルのロケーション	これは、ステージ 3 メッセージ用のログ・ファイルのロケーションです。	ファイル・システム内の有効なロケーション (ドライブとディレクトリー) を入力します。
ログ・ファイル名	これは、ステージ 3 メッセージ用のログ・ファイルの名前です。	有効なファイル名を入力します。

b. ウィザードの 2 ページ目で、次の表の説明のとおり、設定を編集する。

表 66. ウィザードの 2 ページ目に入力する設定

設定	意味	値
「Java」または「COBOL」ラジオ・ボタン	この選択により、マイグレーション・ツールが Java ソース・フォルダーを含むプロジェクトを作成するかどうかが決まります。	COBOL のみを生成する場合は、 COBOL をクリックします。Java を生成する場合は、 Java をクリックします。
「最新バージョン」または「最も古いバージョン」	このオプションでは、対象のマイグレーション・セットのどのバージョンをワークスペースにインポートするかを指定します。	ラジオ・ボタンの 1 つをクリックします。

表 66. ウィザードの 2 ページ目に入力する設定 (続き)

設定	意味	値
既存ファイルを上書き	ステージ 3 (インポート・プロセス) では、ステージ 2 で作成した EGL を使用して、ステージ 1 レポートの中で指定された EGL ファイルを作成し、インポートします。ステージ 3 でインポートしようとしている EGL ファイルと同名の EGL ファイルがワークスペースにすでに存在する場合は、このオプションによってこれらのファイルを上書きするかどうかが決まります。	<p>「既存ファイルを上書き」オプションを選択すると、現在マイグレーション中のマイグレーション・セットに、ワークスペース内の既存ファイルに配置する必要があるパーツが含まれている場合に、ステージ 3 マイグレーション・ツールがどのような処理を行うのかを指定できます。「既存ファイルを上書き」を選択すると、ステージ 3 マイグレーション・ツールは既存のファイルを置き換え、現行マイグレーション・セット内にあるパーツのみを組み込みます。「既存ファイルを上書き」を選択解除した場合、ステージ 3 マイグレーション・ツールは新規パーツすべてを既存ファイルにマージします。新規パーツは、パーツ型別にアルファベット順に配置されます。</p> <p>このオプションを選択した場合の影響については、58 ページの『ファイルの上書きとマージ』を参照してください。</p>
マイグレーションしたファイルを一時ディレクトリーに保管	このオプションを選択すると、EGL ファイルをファイル・システム内のロケーションに保管できます。これにより、プロジェクトの複数のバージョンの EGL ファイルに同時にアクセスできるようになります (ワークスペースで一度に表示できるバージョンは 1 つのみです)。このロケーションから、リポジトリに EGL ファイルを直接移動できます。	<p>マイグレーション・セットの複数バージョンをマイグレーションする予定の場合は、次の手順を実行します。</p> <ol style="list-style-type: none"> 1. それぞれのバージョンを別々のサブディレクトリーに書き込むことができるように、このチェック・ボックスを選択する。 2. これらのバージョンを配置するサブディレクトリーを格納する「フォルダー」を指定する。 3. 「今すぐマイグレーションを実行」オプションを選択解除する。リソース所要量が大きいので、一時ディレクトリーへのマイグレーションはバッチ・モードでのみ行う必要があります。「今すぐマイグレーションを実行」を選択した場合は、本当にオンライン・モードで実行するかどうか、マイグレーション・ツールから確認を求められます。 4. 「現行構成をファイルに保管」オプションを選択する。また、現行構成を .vgmig ファイルとして保管する先のプロジェクトとファイル名も指定する必要があります。オンラインまたはバッチのどちらのモードでマイグレーションを行うかに関係なく、「マイグレーションしたファイルを一時ディレクトリーに保管」オプションを選択した場合は .vgmig ファイルが必要です。バッチ・モードでステージ 3 を実行する場合は、マイグレーション設定を指定するために、保管した .vgmig ファイルを指示します。

表 66. ウィザードの 2 ページ目に入力する設定 (続き)

設定	意味	値
フォルダー (Folder)	これは、EGL ファイルを保管するディレクトリです。「 フォルダー (Folder) 」に指定したディレクトリの下に、それぞれのマイグレーション・セット・バージョンのサブディレクトリが作成されます。	ファイル・システム内の既存ディレクトリを指定します。
今すぐマイグレーションを実行	後でマイグレーションを行うために設定ファイルをセットアップするだけでなく、この時点でステージ 3 を実行するように指定します。	「 今すぐマイグレーションを実行 」オプションを選択してステージ 3 をオンライン・モードで実行するか、または後でステージ 3 をバッチ・モードで実行できるように、「 現行構成をファイルに保管 」オプションを選択して設定を保管する必要があります。後で参照できるように設定のコピーを保持したい場合は、両方のオプションを選択できます。「 マイグレーションしたファイルを一時ディレクトリに保管 」をすでに選択した場合は、「 今すぐマイグレーションを実行 」を選択解除してください。一時ディレクトリへの保管は、バッチ・モードでのみ実行できます。「 今すぐマイグレーションを実行 」の選択は、オンライン・モードでマイグレーションを行うように指示します。
現行構成をファイルに保管	この設定により、指定中の設定をファイルに保管できます。後で、次のどちらかの方法でステージ 3 を実行できます。 <ul style="list-style-type: none"> オンライン・モードの場合は、保管した .vgmig ファイルを右クリックし、「マイグレーションの開始 (Start Migration)」をクリックする。 バッチ・モードの場合は、-importFile オプションを使用して、保管した .vgmig ファイルを指定する。詳しくは、225 ページの『バッチ・モードでのステージ 2 の実行』を参照してください。 	「 今すぐマイグレーションを実行 」オプションを選択してステージ 3 をオンライン・モードで実行するか、または後でステージ 3 をバッチ・モードで実行できるように、「 現行構成をファイルに保管 」オプションを選択して設定を保管する必要があります。後で参照できるように設定のコピーを保持したい場合は、両方のオプションを選択できます。 このオプションを選択した場合は、現行構成を .vgmig ファイルとして保管する先のパスとファイル名も指定する必要があります。後でステージ 3 を実行する場合は、マイグレーション設定を指定するために、保管した .vgmig ファイルを指示します。
パス	ファイルを保管する先のプロジェクトを指定します。	¥projectName。ここで、projectName は保管するファイルに格納するプロジェクトの名前です。
ファイル名	設定を保管する先のファイルの名前を指定します。	fileName。ここで、fileName は目的のファイルの名前で、ファイル拡張子を付けずに 指定します。拡張子 .vgmig が自動的に付加されます。

- c. ウィザードの 3 ページ目で、インポートするマイグレーション・セットを選択する。

注: 「データベースからの VAGen マイグレーション・インポート」ウィザードには、EGL にマイグレーション済みのマイグレーション・セッ

トのみがリストされます。このため、ステージ 3 を実行する前に、必ずステージ 2 を実行してマイグレーション・セットを EGL ソースに変換し、EGL をマイグレーション・データベースに格納してください。マイグレーション・セットがリストされない場合は、ステージ 2 のマイグレーションを実行したことを確認してください。

4. 「終了」をクリックします。
5. マイグレーション・ツールは、選択したマイグレーション・セットに基づいて、EGL プロジェクト、EGL ソース・フォルダー、および EGL パッケージを作成します。このツールは、EGL ソースをマイグレーション・データベースから抽出し、マイグレーション・セットに基づいて EGL ファイルを作成します。マイグレーション・ツールはまた、**import** ステートメントを組み込み、パーツ参照を解決できるようにプロジェクトの EGL ビルド・パスを更新します。

バッチ・モードでのステージ 3 の実行

ステージ 2 とステージ 3 をバッチ・モードで実行する場合の違いは、.vgmig ファイルの作成に使用するウィザードのみです。ステージ 3 のみを実行するために .vgmig をセットアップする方法については、229 ページの『ステージ 3 ツールの実行』を参照してください。.bat ファイルに組み込むコマンドの詳細、およびバッチ・モードのために指定できるオプションについては、225 ページの『バッチ・モードでのステージ 2 の実行』を参照してください。

一時ディレクトリーに書き込まれたマイグレーション・セットの使用

ステージ 3 の出力を一時ディレクトリーに送った場合は、マイグレーション・ツールによって、それぞれのマイグレーション・セット・バージョンごとにサブディレクトリーが 1 つ作成されます。サブディレクトリー名の形式は *migrationSetName_versionName* です。

プロジェクトをワークスペースに取り込むには、次の 2 つの手法を使用できます。

- **手法 1** はサブディレクトリーにあるプロジェクトが少数のみである場合に便利です。ただし、この手法は EGL Web プロジェクト (VAGen Web トランザクションまたは UI レコード) をサポートしていません。この手法では、既存のワークスペースをそれぞれのプロジェクトに指示することができます。この手法では、プロジェクトをワークスペースにコピーせずに、単にワークスペースからプロジェクトを使用できるようにします。プロジェクト内のファイルを変更または削除すると、ワークスペースが指示しているファイル・システムのディレクトリー内で変更が行われます。
 1. 既存のワークスペースから、「ウィンドウ」->「設定」->「一般」->「ワークスペース」をクリックし、「自動的にビルド」をクリアします。これにより、それぞれのプロジェクトを取り込むときに再ビルドが複数回行われなくなります。
 2. ワークベンチ・ビューから、「ファイル」->「インポート」をクリックします。「一般」を展開し、「既存プロジェクトをワークスペースへ」を選択します。「次へ」をクリックします。

3. 「ルート・ディレクトリーの選択」をクリックします。「参照」をクリックして、マイグレーション・セット・バージョン用のサブディレクトリーを *migrationSetName_migrationSetVersion* という形式で指定します。
 4. 現行ワークスペースに組み込むプロジェクトを選択します。
 5. 「プロジェクトをワークスペースにコピー」を選択します (オプション)。
 6. 「完了」をクリックして、プロジェクトをワークスペースにインポートします。
- **手法 2** はサブディレクトリーに多数のプロジェクトがある場合に便利です。EGL Web プロジェクトを使用する場合、この手法は必須です。この手法では、以下の手順によってサブディレクトリーのワークスペースを起動します。
 1. EGL 開発環境を開始する。
 2. ワークスペース名を指定するためのプロンプトが出されたら、マイグレーション・セット・バージョンを含むサブディレクトリーを指定して、「OK」をクリックする。
 3. 以下のワークベンチ設定を変更する。
 - 208 ページの『必要な EGL 設定』で説明したとおりに EGL 機能および設定を設定する。
 - 新規ワークスペースに対して、その他の通常の設定を行う。
 4. 次のいずれかの手法を使用して、ワークスペースをビルドする。
 - 「プロジェクト」->「自動的にビルド」をクリックする。
 - 「プロジェクト」->「クリーン」をクリックする。「クリーン」ウィンドウで、「すべてのプロジェクトをクリーン」をクリックする。その後、「OK」をクリックする。
 5. プロジェクトがビルドできないことを示すエラーが「問題」ビューに表示された場合は、「プロジェクト・エクスプローラー」ビューを使用して、閉じたプロジェクトを見つける。閉じたプロジェクトは、プロジェクト名の左側にプラス (+) 記号が付いていません。ポップアップ・メニューから「プロジェクトを開く」をクリックして、閉じたプロジェクトを開きます。これで、「プロジェクト・エクスプローラー」ビューにプロジェクトが表示されます。

第 8 章 単一ファイル・モードでのマイグレーションの実行

ステージ 1 から 3 を使用してマイグレーションを実行する代わりに、単一ファイル・モードでマイグレーションを実行できます。このプロセスにより、1 つの外部ソース形式ファイルを直接 EGL ファイルにマイグレーションできます。このモードでマイグレーションを実行するには、まず VisualAge Generator パーツを外部ソース形式ファイルにエクスポートしてから、外部ソース形式ファイルを EGL にインポートする必要があります。インポート・プロセス中に、外部ソース形式ファイルは、設定に応じて 1 つ以上の EGL ファイルにマイグレーションされます。

パーツを VisualAge Generator からエクスポートするには、次の手順で行います。

1. VisualAge Generator on Java (または VisualAge Generator on Smalltalk) を開始し、VAGen パーツ・ブラウザを開く。
2. エクスポートするパーツを選択し、選択項目を右クリックする。
3. ポップアップ・メニューから、「インポート/エクスポート」->「VAGen エクスポート (VAGen Export)」(または「関連パーツを含む VAGen エクスポート (VAGen Export with Associates)」) をクリックする。
4. 外部ソース形式ファイルの名前をボックスに入力し、「保管 (Save)」をクリックする。(既存ファイルの名前を入力した場合は、そのファイルにパーツを追加するか、ファイルを上書きするかを尋ねられます。どちらか適当な方法を選択してください。)

単一ファイル・モード用に準備するには、次の手順で行います。

1. EGL 開発環境を開始し、ワークスペースを指定する。(例: d:\workspaces\myworkspace)
2. 208 ページの『必要な EGL 設定』で説明したとおりに EGL 機能および設定を設定する。
3. マイグレーションの設定を行う。これを行うための方法については、211 ページの『VAGen マイグレーションの設定』を参照してください。
4. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」->「VAGen マイグレーション」をクリックする。通常は、「パーツを EGL ファイルに分離」の設定が選択されていることを必ず確認してください。この設定を選択すると、それぞれのプログラム、マップ・グループ、テーブル、および UI レコードが独自のファイルに配置されます。これは、ファイルごとに 1 つの生成可能パーツを配置するという EGL の要件に準拠しています。「パーツを EGL ファイルに分離」を選択解除した場合は、UI レコード以外のすべてのパーツが同じ EGL ファイルに配置されます。単一ファイル・モードのパーツ配置アルゴリズムについて詳しくは、32 ページの『単一ファイル・マイグレーションの概要』を参照してください。
5. 新規 EGL プロジェクトを作成する。(例: MyProject)。VAGen Web トランザクションをマイグレーションする予定がない場合は「**一般プロジェクト (General Project)**」を使用し、VAGen Web トランザクションをマイグレーションするか、または新規の EGL JSFHandlers を開発する予定の場合は、「**Web プロジェクト**」を使用します。

6. EGL プロジェクトの EGLSource ディレクトリーの下に、新規 EGL パッケージを作成する。(例: my.pkg)
7. オンライン・モードでの実行について詳しくは、236 ページの『ユーザー・インターフェースを使用した単一ファイル・マイグレーションの実行』を参照してください。また、バッチ・コマンド・ファイルの作成により、単一のコマンド・ファイルを使用して複数の外部ソース形式ファイル进行处理する方法については、237 ページの『バッチ・モードを使用した単一ファイル・マイグレーションの実行』を参照してください。

ユーザー・インターフェースを使用した単一ファイル・マイグレーションの実行

外部ソース形式ファイルを EGL にインポートするには、次の手順で行います。

1. 「プロジェクト・エクスプローラー」ビューから、作成される EGL ファイルを取り込む EGL パッケージを選択する。
2. パッケージを右クリックして、「インポート」をクリックする。
3. 「インポート」ウィンドウから、「その他」を展開します。「**VAGen 外部ソース形式ファイル**」を選択し、「次へ」をクリックする。
4. 「入力ファイル名」フィールドに、インポートする外部ソース形式ファイルの名前を入力する。
5. 「ソース・フォルダー」フィールドに、EGL ファイルを配置するプロジェクトとソース・フォルダーの名前を入力する。(例: MyProject\EGLSource)
6. 「パッケージ名」フィールドに、EGL ファイルを配置するパッケージの名前を入力する。マイグレーション・ツールは、EGL ファイル内の **package** ステートメントに指定したパッケージ名も使用します。(例: my.pkg)
7. 「**EGL ファイル名**」フィールドに、外部ソース形式ファイルから作成される EGL ファイルの名前を入力する。デフォルトでは、EGL ファイル名は外部ソース形式ファイルと同じですが、.egl ファイル拡張子が付けられます。マイグレーション・ツールが、「**パーツを EGL ファイルに分離**」設定、および外部ソース形式ファイル内のパーツ型を使用して、単一ファイル・モードでのマイグレーション中に作成するファイルを決定する方法については、32 ページの『単一ファイル・マイグレーションの概要』を参照してください。
8. 「**ログ・ファイルのロケーション**」フィールドに、マイグレーション・ログ・ファイルを配置するドライブとディレクトリーを入力する。「**ログ・ファイル名**」フィールドに、マイグレーション・ログ・ファイルの名前を入力する。**ログ・ファイル名**のデフォルトは、指定した外部ソース形式ファイルの名前と一致します。マイグレーション・ログ・ファイルには、マイグレーション中に書き込まれたメッセージがすべて保管されます。
9. 「終了」をクリックします。「EGL ファイル名」フィールドに指定したファイル名が、指定したパッケージにすでに存在する場合は、そのファイルに追加するか上書きするかを尋ねるプロンプトが出されます。上書きプロンプトへお客様の応答に基づいて、マイグレーション・ツールは次の方法でパーツを EGL ファイルに置きます。

- 既存の `targetFile` への上書きを希望しないことを指定した場合、2 番目のインポートのデータ項目、関数、PSB、および非 VGUI レコードは、`targetFile` に追加されます。2 番目のインポートにある共通パーツは、`targetFile` 内で重複パーツになります。
 - 既存の `targetFile` への上書きを希望することを指定した場合、2 番目のインポートのデータ項目、関数、PSB、および非 VGUI レコードは、既存の `targetFile` を完全に置き換えます。このため、最初のインポートに含まれていて、2 番目のインポートに含まれていないパーツはすべて失われます。
 - マイグレーション設定として「パーツを EGL ファイルに分離」を選択した場合、マイグレーション・ツールはプログラム、FormGroup、および DataTable 用に作成されたファイルを上書きします。マイグレーション設定を選択解除した場合、これらのパーツは `targetFile` に配置され、上書きプロンプトに対する応答に従って追加または上書きされます。
 - マイグレーション・ツールは、VGUI レコード用のファイルと `.eglbld` ファイルを常に上書きします。
10. マイグレーションが完了すると、以下の出力が表示されます。
- 指定したプロジェクト、EGLSource フォルダ、およびパッケージに、1 つ以上の EGL ファイルがリストされます。マイグレーション・ツールが、「パーツを EGL ファイルに分離」設定、および外部ソース形式ファイル内のパーツ型を使用して、単一ファイル・モードでのマイグレーション中に作成するファイルを決定する方法については、32 ページの『単一ファイル・マイグレーションの概要』を参照してください。
 - マイグレーション・ツールは、ポップアップ・ウィンドウにエラー・メッセージを表示します。ログ・ファイルのロケーションを指定しなかった場合は、「ファイルに保管 (Save to File)」をクリックして、メッセージをファイルに保管できます。必ず、ポップアップ・ウィンドウを閉じてください。
11. 「自動的にビルド」オプションが選択されていれば、自動的に検査が実行されます。その他の場合は、プロジェクトを右クリックし、「プロジェクト」->「プロジェクトのビルド」を選択します。これにより、検証が実行されるので、プロジェクト内のファイルすべてに関する最新メッセージが「問題」ビューに反映されます。

バッチ・モードを使用した単一ファイル・マイグレーションの実行

ユーザー・インターフェースを使用する場合は、外部ソース形式ファイルを一度に 1 つずつマイグレーションできます。バッチ・モードでは、単一のコマンド・ファイルによって複数の外部ソース形式ファイルをマイグレーションできます。バッチ・モードを使用するには、次のステップを実行します。

1. `.bat` ファイル拡張子の付いたファイルを作成する。Windows 環境の場合、`.bat` ファイルの内容は次のようにする必要があります。

```
set INSTALL_PATH=InstallDirectory
set SHARED_INSTALL_PATH=SharedInstallDirectory
set JDK_PATH=jdk\jre\bin
set PLUGIN_PATH=plugins
set MIG_JAR=com.ibm.etools.egl.vagenmigration_version.jar
set STARTUP_JAR=org.eclipse.equinox.launcher_version.jar
set path=%INSTALL_PATH%\%JDK_PATH%;
set classpath=%SHARED_INSTALL_PATH%\%PLUGIN_PATH%\%MIG_JAR%;
%INSTALL_PATH%\%PLUGIN_PATH%\%STARTUP_JAR%
```

```
cd InstallDirectory
java com.ibm.etools.egl.internal.vagenmigration.batch.VGMIG
  -importFile Path\ExternalSourceFormatFile.esf
  -eglFile Path\EGLFile.egl
  -data Path\workspace
  -package packageName
  -overwrite
>Path\LogName.log
```

注:

- 次のステートメントは、ステートメント全体が 1 行になるように入力する必要があります。
 - Windows 環境の場合は、**set classpath** ステートメント。
 - **java** ステートメント。
- マイグレーションするそれぞれの外部ソース形式ファイルごとに、**java** ステートメントを 1 つずつ繰り返します。
- *InstallDirectory* は、EGL デベロッパー製品をインストールしたドライブとディレクトリーです。set INSTALL_PATH ステートメントおよび cd (ディレクトリーの変更) ステートメントに、*InstallDirectory* を組み込む必要があります。
- *SharedInstallDirectory* は、EGL デベロッパー製品の共用リソースをインストールしたドライブとディレクトリーです。set SHARED_INSTALL_PATH ステートメントに *SharedInstallDirectory* を組み込む必要があります。

注: 現在ご使用の製品をインストールする前に、前のバージョンの EGL デベロッパー製品をインストールし、保持していた場合、対象になるインストール・ディレクトリーまたは共用インストール・ディレクトリーは、以前のインストール時に使用されたディレクトリーである可能性があります。

- *version* は、プラグイン・バージョン番号です。バージョン番号を判別するには、以下のプラグイン・ディレクトリーを確認します。
 - MIG_JAR プラグインは、*SharedInstallDirectory\plugins* にあります。バージョン番号は、7.5.0.RFB_20080811_1638 などです。
 - STARTUP_JAR プラグインは、*InstallDirectory\plugins* にあります。バージョン番号は、1.0.100.v20080509-1800 などです。

通常は、対応するプラグイン・ディレクトリーの中の .jar ファイルに付けられている、最も大きいバージョン番号を使用する必要があります。

- *Path\ExternalSourceFormatFile.esf* は、マイグレーションする外部ソース形式ファイルのドライブ、ディレクトリー、およびファイル名を指します。(例: d:\temp\VA GenFiles\PROG1.esf)
- *Path\EGLFile.egl* は、作成する EGL ファイルのドライブ、ディレクトリー、およびファイル名を指します。ディレクトリーには EGL ソース・ファイルを入れるワークスペース、EGL ソース・フォルダー、およびパッケージが含まれている必要があります。(例: d:\myworkspace\MyProject\EGLSource\my\pkg\prog1.egl) *EGLFile.egl* は、「VAGen 外部ソース形式ファイルのインポート (Import VAGen External Source Format File)」ウィザードを使用するときに指定する「EGL ファイ

ル名」フィールドと同じように使用されます。マイグレーション・ツールが、「**パーツを EGL ファイルに分離**」設定、および外部ソース形式ファイル内のパーツ型を使用して、単一ファイル・モードでのマイグレーション中に作成するファイルを決定する方法については、32 ページの『単一ファイル・マイグレーションの概要』を参照してください。

- *Path¥workspace* は、EGL ファイルを配置するドライブ、ディレクトリー、およびワークスペース名です (例えば、d:¥workspaces¥myworkspace)。-data オプションを指定しない場合、「VAGen マイグレーション設定」に指定した内容は無視され、マイグレーション・ツールはデフォルトの VAGen マイグレーション設定を使用します。VAGen マイグレーション設定を指定する場合は、-data オプションを指定し、指定対象のワークスペースをポイントする必要があります。
- *packageName* は、EGL ファイルに関連付けるパッケージの名前です。(例: my.pkg) パッケージ名は、マイグレーション・ツールが作成する .egl ファイルの **package** ステートメントにも使用されます。
- -overwrite パラメーターはオプションです。このパラメーターは、指定されたディレクトリーにある、指定された名前の既存 EGL ファイルを上書きするかどうかマイグレーション・ツールに指示します。
- *Path¥LogName* は、対応する外部ソース形式ファイルのマイグレーションに関して作成するログ・ファイルのドライブ、ディレクトリー、およびファイル名を指定します。マイグレーション・メッセージをログ・ファイルに送る設定はオプションですが、バッチ・モードでマイグレーション・ツールからメッセージを受け取る唯一の方法です。同じ .bat ファイルに複数の **java** コマンドがある場合は、それぞれの **java** コマンドごとに異なるログ・ファイル名を必ず指定してください。

Windows 環境の場合、**java** コマンドの例は次のようなものです (ただし、コマンド全体を 1 つの行に入力する必要があります)。

```
java com.ibm.etools.egl.internal.vagenmigration.batch.VGMIG
-importFile d:\temp\VAGenFiles\progl.esf
-eglFile d:\workspaces\myworkspace\MyEGLProject\EGLSource\my\pkg\progl.egl
-data d:\workspaces\myworkspace
-package my.pkg -overwrite >d:\temp\EGLog\progl.log
```

2. EGL 開発環境をシャットダウンする。
3. コマンド・プロンプト・ウィンドウを開き、.bat ファイルがあるディレクトリーにナビゲートして、.bat ファイルを実行する。

注: 次のメッセージは無視してかまいません。

PolicyClassLoader は、ポリシー com.ibm.jxesupport.JxeClassLoaderPolicy を見つけることができませんでした (PolicyClassLoader could not find policy com.ibm.jxesupport.JxeClassLoaderPolicy)

4. 処理が完了すると、EGL ファイルとログ・ファイルが、それぞれ指定したディレクトリーに保管されます。ログ・ファイルには、マイグレーション済みパーツのリストと、エラー・メッセージが保管されています。メッセージは、オンライン・モードで「インポート」ウィザードを使用するときにポップアップ・ウィンドウにリストされるメッセージと同じです。
5. EGL 開発環境を開始する。

6. 外部ソース形式ファイルをインポートした先のプロジェクトを右クリックし、「**更新**」をクリックする。この最新表示によってファイル・システムのプロジェクトが最新表示になり、バッチ・モードでのマイグレーション中に作成、追加、または上書きされた EGL ファイルが EGL に認識されます。「**自動的にビルド**」オプションを選択している場合、これにより、検証が実行されるので、プロジェクト内のファイルすべてに関する最新メッセージが「問題」ビューに反映されます。次に、作成したパッケージを展開して EGL ファイルを表示できます。

第 5 部 マイグレーションの完了

第 9 章 マイグレーションの完了

ステージ 1 から 3 までのマイグレーション、または単一ファイル・マイグレーションを使用してソース・コードをマイグレーションした後、いくつかの追加作業を行う必要があります。以下のようなタスクがあります。

- ビルド順序の設定。
- 設定のエクスポート。
- ソース・コード・リポジトリ内の EGL プロジェクトとパッケージに関するベースラインの保管。
- 単一ファイルのマイグレーションを完了するための予備タスク。
- EGL ソース・コードの検討。
- EGL ビルド記述子パーツの検討。
- EGL リンケージ・オプション・パーツの検討。
- EGL リソース関連パーツの検討。
- テンプレートとして使用するバインド制御パーツの確立。
- プログラム固有のバインド制御パーツの設定。
- リンク・エディット・コマンドの検討。
- VGWebTransactions の検討。
- デバッグの準備。
- EGL サーバー製品のインストール。
- VAGen の準備テンプレートおよび手順の EGL ビルド・スクリプトへの変換。
- VAGen ランタイム・テンプレートの変換。
- VAGen 予約語ファイルの変換。
- COBOL 生成を行う場合の生成とテスト。
- Java 生成を行う場合の生成とテスト。
- 標準の検討。
- ソース・コードの二重メンテナンスの計画。
- VisualAge Generator 互換モードの使用を排除するかどうかの検討。

ビルド順序の設定

EGL によるワークスペースのビルド時には、**ビルド順序**設定に基づいてプロジェクトがビルドされます。この設定のデフォルトでは、最適なパフォーマンスを得ることができません。一般的に、最も頻繁に参照されるプロジェクト (共通プロジェクトなど) を最初にビルドします。これにより、別のプロジェクト内のファイルがパッケージをインポートする場合に、共通パーツのロケーションが既知になっています。現在のビルド順序を確認するか、または変更するには、以下のステップを実行します。

1. 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」をクリックします。

2. 「一般」を展開し、次に「ワークスペース」を展開します。「ビルド順序」を選択します。
3. ビルド順序は、以下のステップにより手動で変更できます。
 - a. 「デフォルトのビルド順序の使用」オプションをクリアします。
 - b. 「プロジェクトのビルド順序」リストでプロジェクトを選択し、上/下ボタンを使用してビルド順序を変更します。
 - c. すべての変更が完了した後、「適用」をクリックし、「OK」をクリックします。
4. また、「サイクルでビルドするときの最大反復」を変更して、検査メッセージの解決時にビルド・ツールがプロジェクトの処理を反復する回数を削減または増加することもできます。

EGL プロジェクトには、ビルド順序の設定に役立つツールがあります。ステージ 3 のマイグレーション・ツールでは、このツールが自動的に呼び出されて、ビルドの開始前にビルド順序を設定します。このビルド順序ツールは、別のプロジェクトをワークスペースに追加する場合にも実行する必要があります。ビルド順序ツールを実行するには、以下のステップを実行します。

1. 「ワークベンチ」ウィンドウから、「プロジェクト」->「EGL プロジェクトのビルド順序の最適化」をクリックします。ツールにより、「デフォルトのビルド順序の使用」オプションがクリアされて、「プロジェクトのビルド順序」リストが更新され、最も頻繁に参照されるプロジェクトがビルド・リストの最初に表示されるようになります。
2. 前述のようにビルド順序を検討し、プロジェクト参照に関する認識に基づいて、追加の変更が必要であるかどうかを判断します。

設定のエクスポート

パイロット・プロジェクトでの EGL に関する処理を完了した後で、208 ページの『必要な EGL 設定』、210 ページの『推奨設定』、211 ページの『VAGen マイグレーションの設定』、および 243 ページの『ビルド順序の設定』で説明されている必須または推奨の設定以外の追加設定が行われている場合があります。例えば、ご使用のソース・コード・リポジトリと、選択したライブラリー管理プロセスに関連して、別途設定を行う場合があります。その設定をファイルにエクスポートすれば、他の開発者がその設定をインポートして、個別の設定を開始するための基礎として使用できます。このエクスポート手法は、設定をワークスペース間で簡単に移動するための手段にもなります。そのような設定をエクスポートするには、以下のステップを実行します。

1. 「ワークベンチ」ウィンドウで、「ファイル」->「エクスポート」をクリックします。
2. 「エクスポート」ページで、「一般」を展開します。「設定」を選択します。「次へ」をクリックします。
3. 「設定のエクスポート」ページで、「すべてエクスポート」をクリックします。「参照」をクリックして、設定を保存するディレクトリーおよびファイル名を指定します。
4. 「終了」をクリックします。

他の開発者は、以下のステップを実行することにより、ワークスペースに設定をインポートできます。

1. 「ワークベンチ」ウィンドウで、「ファイル」->「インポート」をクリックします。
2. 「インポート」ページで、「一般」を展開します。「設定」を選択します。「次へ」をクリックします。
3. 「設定のインポート」ページで、「すべてインポート」をクリックします。「参照」をクリックして、その設定が収容されているファイルを特定します。
4. 「終了」をクリックします。

注: この手法は、パースペクティブとビューの設定値には影響を及ぼしません。設定のみが変更されます。

EGL プロジェクトとパッケージに関するベースラインの保管

「問題」ビューのメッセージを解決したり、マイグレーション済みの EGL コードを変更したりする前に、ソース・コード・リポジトリ内で EGL プロジェクトとパッケージのバージョンを作成できます。マイグレーションの直後に EGL のプロジェクトとパッケージを保管し、バージョン管理を行うと、これがベースラインとなるので、マイグレーション・ツールによって作成されたソース・コードが正確に分かります。このベースラインは、手作業で行う必要があるコード変更を追跡する手段にもなります。特に、パイロット・プロジェクト中の変更内容をすべて収集して、必要な変更のタイプを文書化するための手段として有効です。この文書は、他のサブシステムをマイグレーションするための補助になります。

単一ファイルのマイグレーションを完了するための予備タスク

単一ファイル・マイグレーションは、ステージ 1 から 3 のマイグレーションが行うことの一部を行いません。以下のタスクを手動で実行する必要があります。

- その対応する FormGroup 内で書式をネストする。このタスクは、2 つの FormGroup を同じパッケージにマイグレーションし、その 2 つの FormGroup に同じ書式名が含まれている場合に必要になります。(例: MAP1)
- 同じ EGL パッケージ内で重複するパーツを解決する。この作業は、2 つのプログラムを関連パーツとともに同じ EGL パッケージにマイグレーションする場合に、これら 2 つのプログラムが共通のパーツを共用していると必要になることがあります。共通のパーツ定義を中央の共通ファイルに分離することもでき、どちらかのファイルから重複パーツを除去することもできます。すべてのファイルが同じパッケージにある場合は、EGL ビルド・パス・プロパティを変更したり、**import** ステートメントを追加したりする必要はありません。
- 現行プロジェクトの EGL ビルド・パス・プロパティを更新して、現行プロジェクトがパーツ名を解決するために参照する必要があるプロジェクトをすべてリストする。EGL ビルド・パスを更新するには、「プロジェクト・エクスプローラー」ビュー内の現在のプロジェクトを右クリックし、「プロパティ」をクリックします。「EGL ビルド・パス」を選択する。「EGL ビルド・パス」ページで、「プロジェクト」タブをクリックして現行プロジェクトが参照する必要がある追加プロジェクトを選択します。EGL ビルド・パスには、現行プロジェクト内のファイルがインポートする必要があるパッケージを含むプロジェクトをすべて

組み込んでください。例えば、FileA が ProjectB 内にあり、FileA が packageC をインポートする必要がある場合は、packageC が置かれているプロジェクトを ProjectB の EGL ビルド・パスに必ず組み込みます。

- 任意の **import** ステートメントを EGL ファイルに追加して、そのファイルが参照する必要がある共通パッケージを指定する。 **import** ステートメントに指定するパッケージは、現行の EGL ファイルが置かれているプロジェクトの EGL ビルド・パスに指定されているプロジェクトに存在している必要があります。例えば、FileA が ProjectB にある場合、FileA 内の **import** ステートメントは、ProjectB の EGL ビルド・パスに指定されたプロジェクトにあるパッケージのみを参照できます。

ステージ 1 から 3 までのマイグレーションと単一ファイル・マイグレーションに共通するタスク

EGL ソース・コードの検討

ステージ 1 から 3 までのマイグレーション、または単一ファイル・マイグレーションのいずれを使用した場合にも、次のタスクを実行する必要があります。

- マイグレーション・ログ、または「TODO」リスト・ログにあるエラーを検討し、解決します。これらのエラーは、マイグレーション・ツールが解決できなかった未確定状態を反映しています。EGL ソース・コードを変更して、これらのエラーを解決します。例えば、VAGen RETR ステートメントを使用し、検索列を明示的に指定しなかった場合に、テーブルがマイグレーション中に使用できなければ、EGL 構文には EZE_UNKNOWN_SEARCH_COLUMN が含まれています。DataTable 定義に基づいた正しい検索列名を指定して、EGL ソース・コードを更新する必要があります。マイグレーション・ログまたは「TODO」リスト・ログのメッセージを解決するためのヘルプについては、471 ページの『付録 C. マイグレーション・ツールからのメッセージ』を参照してください。マイグレーション・ツールが意図的に無効な EGL 構文を作成する場合に使用する特定のストリングの検索および解決については、509 ページの『付録 D. 「問題」ビューのメッセージ』を参照してください。
- 予約語であるプログラム名、DataTable 名、または FormGroup 名が存在する場合は、パーツ名を変更する必要があります。COBOL を生成する場合は、パーツに **alias** プロパティを設定して、オリジナルのパーツ名を指定できます。alias を指定することにより、プログラム、DataTable、または FormGroup に対する外部参照を変更する必要がなくなります。プログラム名を変更する場合は、そのプログラムと名前が同じである、すべてのプログラム固有のバインド制御パーツまたはリンク・エディット・パーツの名前も変更する必要があります。
- 「問題」ビューにあるその他のエラーを検討し、解決します。マイグレーション・プロセスの結果として「問題」ビューに表示される共通メッセージの解決法については、519 ページの『付録 E. 「問題」ビューの IWN.xxx メッセージ』を参照してください。
- レコードまたは関数に関して **containerContextDependent** プロパティを設定する必要があるかどうか判別します。詳しくは、48 ページの『containerContextDependent プロパティ』を参照してください。

EGL ビルド記述子パーツの検討

マイグレーション・ツールは、VAGen 生成オプション・パーツを EGL ビルド記述子パーツに変換します。ただし、一部の VAGen オプションには同等な EGL オプションがありません。また、EGL には新しいビルド記述子オプションがいくつかあり、これらの設定が必要になることがあります。これらの変更が原因で、「問題」ビューにエラーが表示される場合があります。マイグレーション・プロセスの結果として「問題」ビューに表示される共通メッセージの解決法については、519 ページの『付録 E. 「問題」ビューの IWN.xxx メッセージ』を参照してください。問題を解決するには、テキスト・エディターが必要になることがあります。ステージ 1 から 3 までのマイグレーション、または単一ファイル・マイグレーションのいずれれを使用した場合にも、次のタスクを実行する必要があります。

- 汎用ビルド記述子オプションの検討。
- COBOL 生成ビルド記述子オプションの検討。
- Java 生成ビルド記述子オプションの検討。
- デバッグ・ビルド記述子パーツの設定。

汎用ビルド記述子オプションの検討

COBOL または Java のどちらの生成を予定している場合にも、次のビルド記述子オプションを検討する必要があります。

- 小数点がコンマ記号である各国語を対象にプログラムを開発する場合は、以下について検討し、EGL **decimalSymbol** ビルド記述子オプションを設定するかどうかを決定します。
 - COBOL を生成する場合は、**decimalSymbol** ビルド記述子オプションが、実行時に使用される言語依存オプション・モジュール内の情報よりも優先されます。**decimalSymbol** ビルド記述子オプションを設定しない場合、実行時に使用される小数点記号は、VisualAge Generator の場合と同様に、言語依存オプション・モジュールによって決まります。
 - Java を生成する場合は、実行時のパフォーマンスが向上するように、**decimalSymbol** ビルド記述子オプションを設定できます。**decimalSymbol** ビルド記述子オプションを設定しない場合は、実行時に使用されるプロパティ・ファイルに **vgj.nls.number.decimal** プロパティを設定できます。
- VAGen EZESYS 特殊機能語を使用してランタイム環境を決定する場合は、EGL ビルド記述子オプション **eliminateSystemDependentCode** の設定が必要になることがあります。このオプションについて詳しくは、オンライン・ヘルプを参照してください。
- マスター・ビルド記述子について詳しくは、オンライン・ヘルプを参照してください。この技法は、VAGen のデフォルト生成オプション・パーツ設定の代替になります。マイグレーション・ツールは、**NOOVERRIDE** 属性を含む生成オプション・パーツを 2 つのビルド記述子パーツに自動的に分割します。ビルド記述子パーツの 1 つは **xxxxx**、もう 1 つは **xxxxx_NOOVERRIDE** という名前になります。ここで、**xxxxx** はオリジナルの VAGen 生成オプション・パーツ名です。**xxxxx** という名前のパーツには、**NOOVERRIDE** 属性を指定していない VAGen 生成オプションすべての EGL 置換表現が含まれています。パーツ名 **xxxxx_NOOVERRIDE** には、**NOOVERRIDE** 属性を指定したすべての VAGen 生成オプションの EGL 置換表現が含まれています。この 2 つのパーツへの分割は、マスター・ビルド記述子を使用する場合に必要です。

- VAGen /OPTIONS 生成オプションを使用して生成オプション・パーツをチェーンしていた場合は、**nextBuildDescriptor** オプションを使用した EGL ビルド記述子パーツのチェーン方法について検討してください。ビルド記述子オプションのセットを VisualAge Generator 内で使用していたものと同じにするには、このチェーンを変更する必要があることがあります。
- COBOL 環境用の Web トランザクション・パーツを生成する場合は、オンライン・ヘルプで、VGUI レコードに関連付けされた Java パーツの生成に使用される **secondaryTargetBuildDescriptor** に関する情報を参照してください。マイグレーション・オプション・パーツが、次のオプションのいずれか 1 つでも含む場合、マイグレーション・ツールは生成オプション・パーツを 2 つのビルド記述子パーツに自動的に分割します。対象となるオプションは、/javadestdir、/javadesthost、/javadestpassword、/javadestuserid、または /javasystem です。ビルド記述子パーツの 1 つは、xxxxx という名前、もう 1 つは xxxxx_TARGET2 という名前です。ここで、xxxxx はオリジナルの VAGen 生成オプション・パーツ名です。xxxxx という名前のパーツには、1 次 (COBOL) ランタイム環境用の EGL VGWebTransaction プログラムを生成する時に使用するすべての VAGen 生成オプションに関する、EGL 置換表現が含まれています。パーツ xxxxx_TARGET2 には、2 次 (Java) ランタイム環境用の EGL VGUI レコードを生成する場合に使用するすべての VAGen 生成オプションに関する、EGL 置換表現が含まれています。マイグレーション・ツールは、2 次ビルド記述子パーツにある次の生成オプションに関して EGL 同等物を配置します。対象の生成オプションは、/javadestdir、/javadesthost、/javadestpassword、/javadestuserid、および /javasystem です。マイグレーション・ツールは、1 次および 2 次ビルド記述子パーツにある次のオプションに関して、EGL 同等物を配置します。対象のオプションは、/genout、/genresourcebundle、/msgtableprefix、/resourcebundlelocale、および /targnls です。マイグレーション・ツールは、1 次ビルド記述子パーツに **secondaryTargetBuildDescriptor** オプションをインクルードし、オプションの値を 2 次ビルド記述子パーツの名前にセットします。
- Web トランザクション・パーツを生成し、メッセージ・テーブルを使用する場合は、**msgTablePrefix** ビルド記述子オプションの変更が必要になることがあります。メッセージ・テーブルは、VGUI レコードを使用するプログラムによって指定されます。メッセージ・テーブルと VGUI レコードが別のパッケージにある場合、2 次ビルド記述子パーツを変更し、パッケージ名 (例えば msgTablePrefix = "packageName.prefixID") をインクルードする必要があります。
- マイグレーション・ツールは、EGL 非 Web プロジェクトを作成するときにデフォルトのビルド記述子を作成しません。これによりユーザーは、マイグレーション済みビルド記述子パーツの 1 つを、デフォルトのビルド記述子に指定することができます。ファイル、パッケージ、EGL ソース・フォルダー、プロジェクト、またはワークベンチのレベルで、デフォルトのビルド記述子を設定できます。生成可能パーツに最も近いデフォルトのビルド記述子を使用されます。例えば、ただ 1 つのファイルに対してデフォルトのビルド記述子を指定し、ワークベンチに対しては別のデフォルトのビルド記述子を指定できます。この場合、そのファイルに含まれているプログラムを生成すると、ファイルのデフォルトのビルド記述子を使用されます。その他のプログラムを生成すると、ワークベンチのデフォルトのビルド記述子を使用されます。デフォルトのビルド記述子を設定するには、以下のいずれかの手法を使用します。

- 特定のファイル、パッケージ、EGL ソース・フォルダー、またはプロジェクトの設定を行うには、リソース (ファイル、パッケージ、フォルダー、またはプロジェクト) を右クリックし、「プロパティ」をクリックします。左側のペインにある「EGL デフォルト・ビルド記述子」を選択します。このリソース内の生成可能パーツすべてのデフォルトとして使用するビルド記述子を選択します。類似した EGL デフォルト・ビルド記述子が存在しない場合、このリソースで生成を実行するときに使用されるデフォルトは「ターゲット・システムのビルド記述子」になります。デバッグ・ツールを使用する場合には、「デバッグ・ビルド記述子」がデフォルトで使用されます。
- ビルド記述子パーツのワークベンチ設定を行うには、「ウィンドウ」->「設定」->「EGL」->「デフォルトのビルド記述子」をクリックします。この設定は、明示的にオーバーライドしなければ、すべてのプロジェクト、パッケージ、ソース・フォルダー、およびファイルに適用されます。生成に使用する「ターゲット・システムのビルド記述子」と、デバッグ・ツールに対して使用する「デバッグ・ビルド記述子」の両方を設定できます。
- マイグレーション・ツールは、マイグレーション・セットの最初の EGL Web プロジェクトを作成するときに自動的にデフォルトのビルド記述子を作成します。これにより、ワークスペースがステージ 3 の終了で更新されるときに、VGUI レコードが JSP に生成されるようになります。プロジェクト、EGL ソース・フォルダー、あるいはプロジェクトが含むパッケージまたはファイルのための、デフォルトのビルド記述子を変更することができます。
- 制御パーツ (ビルド記述子、リンケージ・オプション、リソース関連、バインド制御、およびリンク・エディット・パーツ) がすべて同じファイルにない場合は、現行ファイルを変更して、現行ファイルから参照する他のビルド・パーツを含むファイルのための **import** ステートメントを組み込む必要があります。例えば、buildDescriptorPartA が別のファイルにある linkageTableB を参照している場合は、buildDescriptorPartA を含むファイルに、linkageTableB を含むファイルの **import** ステートメントを組み込む必要があります。 **import** ステートメントを追加するには、EGL ビルド・パーツ・エディターを使用します。

COBOL 生成ビルド記述子オプションの検討

COBOL の生成を予定している場合は、次のビルド記述子オプションを検討または設定する必要があります。

- VisualAge Generator の場合は、COBOL 生成出力がホストに転送されて、準備ステップが実行されます。
 - 転送ステップでは、コード・ページの変換が必要です。使用するすべての特殊文字が正常に転送されるかテストする必要があります。例えば、~ (NOT 符号) は、米国英語用のデフォルトの VAGen およびデフォルトの EGL 変換テーブルを使用する場合、別のコード・ポイントに変換されます。VAGen の場合、生成オプション /contable のデフォルト値は ELACNENU です。これに相当する EGL ビルド記述子オプションは、serverCodeSet="IBM-037" および clientCodeSet="IBM-1252" です。ただし、これらのビルド記述子オプションに対する EGL のデフォルト設定は、serverCodeSet="IBM-037" および clientCodeSet="IBM-850" です。そのため、特殊文字がある場合や、米国英語以外の言語を使用する場合は、ビルド記述子オプション **serverCodeSet** および **clientCodeSet** の設定が必要になることがあります。

- EGL では、ビルド・サーバーを使用して準備ステップを処理します。EGL z/OS および iSeries のビルド・サーバーの場合は、生成の出力の転送に使用するビルド記述子オプション **destPort** を指定する必要があります。リモート・ビルド・サーバーがビルド要求を listen するポート番号を判別するには、Build Server のインストールと構成を行った担当者に問い合わせてください。
- ご使用の COBOL コンパイラーでサポートする数値フィールドの最大が 18 桁のみである場合は、ビルド記述子オプション **maxNumericDigits** を NO に設定します。
- Web トランザクションを使用せず、EGL ページ・ハンドラーを作成する計画がない場合は、COBOL ランタイム環境用に生成するビルド記述子パーツで EGL ビルド記述子オプション **genResourceBundle** を NO に設定します。これにより、DataTable の Java 生成が回避されます。
- z/OS 環境用の COBOL を生成する場合は、以下のタスクを実行する必要もあります。
 - テンプレートとして使用するバインド制御パーツの確立。(254 ページの『テンプレートとして使用するバインド制御パーツの設定』を参照してください。)
 - プログラム固有のバインド制御パーツの設定。(256 ページの『プログラム固有のバインド制御パーツの設定』を参照してください。)
 - リンク・エディット・コマンドの検討。(256 ページの『リンク・エディット・コマンドの検討』を参照してください。)
 - VSE 環境用の COBOL を生成する場合は、リンク・エディット・コマンドを確認する必要もあります。(256 ページの『リンク・エディット・コマンドの検討』を参照してください。)

Java 生成ビルド記述子オプションの検討

Java の生成を予定している場合は、次のビルド記述子オプションを検討または設定する必要があります。

- ビルド記述子オプション **genProject** を追加して、Java 生成の出力を配置する場所を指定します。EGL の **genProject** ビルド記述子オプションにマイグレーションされる VAGen 生成オプションはありません。オプション **genProject** は、次の場合に必要です。
 - HP-UX または SOLARIS 向けに生成する場合
 - VGWebTransactions または VGUI レコード、またはそれらの関連パーツ (DataTable など) を生成する場合。この場合には、必ず **genProject** オプションで EGL Web プロジェクトを指定してください。
- 一部の EGL ビルド記述子オプションの振る舞いは、対応する VAGen 生成オプションとは異なります。オンライン・ヘルプにある、次のビルド記述子オプションに関する情報を参照し、ご使用の環境に応じてこれらのオプションを設定または変更する必要があるかどうか判別してください。
 - **genProperties** (VAGen の /genproperties オプションに基づいてマイグレーション・ツールによって設定される)
 - **enableJavaWrapperGen** および **wrapperCompatibility** (VAGen の /system=JAVAWRAPPER オプションに基づいてマイグレーション・ツールによって設定される)

- 一部の新しい EGL ビルド記述子オプションには、対応する VAGen 生成オプションがありません。オンライン・ヘルプにある、次のビルド記述子オプションに関する情報を参照し、ご使用の環境に応じてこれらのオプションを設定する必要があるかどうか判断してください。

- **dateMask**
- **sessionBeanID**
- **sqlJDBCClass**
- **sqlValidationConnectionURL**
- **tempDirectory** (VGUI レコードの場合のみ)

デバッグ・ビルド記述子パーツの設定

デバッグ時に使用するビルド記述子オプションを含むビルド記述子パーツを作成します。デバッグ・ビルド記述子パーツを作成するためのガイドについては、オンライン・ヘルプを参照してください。

EGL リンケージ・オプション・パーツの検討

マイグレーション・ツールは、VAGen リンケージ・テーブル・パーツを EGL リンケージ・オプション・パーツに変換します。ただし、一部の VAGen オプションには同等な EGL オプションがありません。また、EGL には新しいリンケージ・オプションがいくつかあり、これらの設定が必要になることがあります。これらの変更が原因で、「問題」ビューにエラーが表示される場合があります。マイグレーション・プロセスの結果として「問題」ビューに表示される共通メッセージの解決法については、519 ページの『付録 E. 「問題」ビューの IWN.xxx メッセージ』を参照してください。また、ご使用の環境に応じてサポートされるリンケージ・オプションについて詳しくは、オンライン・ヘルプを参照してください。ステージ 1 から 3 までのマイグレーション、または単一ファイル・マイグレーションのいずれを使用した場合にも、次のタスクを実行する必要があります。

- マイグレーション・ログと「問題」ビューにあるメッセージを検討し、解決します。問題を解決するには、テキスト・エディターが必要になることがあります。
- **callLink** エレメントでは、以下の点についても考慮してください。
 - VisualAge Generator の **linktype** の一部は、EGL でサポートされません。例えば、CSOCALL はサポートされなくなりました。マイグレーション・ツールでは、CSOCALL を **remoteCall** に変換します。ただし、EGL **remoteCall** に対して指定する必要がある属性は、CSOCALL の属性とは異なります。
 - VisualAge Generator の **remoteComType** 値の一部は、EGL ではサポートされません。例えば、DCE、DCESECURE、および APPCIMS はサポートされなくなりました。マイグレーション・ツールは、これらのサポートされない値をそのまま変換するので、無効な EGL リンケージ・オプション・パーツが生じます。このため、「問題」ビューにエラーが表示されます。このエラーを覚え書として、リンケージ・オプション・パーツを変更し、EGL に対して使用するオプションを指定する必要があります。
 - CICS へのリモート呼び出しを使用する場合は、以下のように **remoteComType** の値に基づいて追加のプロパティを指定する必要があります。
 - **remoteComType** = "CICSECI" の使用に変更する場合は、**ctgPort** および **ctgLocation** の各プロパティを追加する必要があります。

- **remoteComType** = "CICSSSL" を使用するように変更する場合は、**ctgKeyStore** および **ctgKeyStorePassword** の各プロパティを追加する必要があります。さらに、VAGen リンケージ・テーブルに **ctgPort** および **ctgLocation** の各プロパティをまだ組み込んでいない場合は、EGL の **remoteComType** = "CICSSSL" のためにこれらのプロパティを組み込む必要があります。
- **remoteComType** = "CICSJ2C" の使用に変更する場合は、**pgmName**、**conversionTable**、**remotePgmType**、**luwControl**、**remoteBind**、**location**、および **parmForm** の各プロパティを追加する必要があります。

選択した通信プロトコルにかかわらず、CICS Transaction Gateway インフラストラクチャーのセットアップおよび構成が必要です。CICS Client 製品を使用した CICS への直接呼び出しはサポートされなくなりました。

- APPCIMS の代わりに **remoteComType** = "IMSTCP" を使用するように変更する場合は、必要な追加プロパティの設定支援について、オンライン・ヘルプを参照してください。指定が必要な値が IMSTCP では違う意味を持つため、既存のプロパティについてもオンライン・ヘルプを確認してください。
- **conversionTable** = "BINARY" は、EGL ではサポートされません。マイグレーション・ツールは、この値を完全に現状のまま変換するので、EGL リンケージ・パーツ内にプレースホルダーができますが、この値は変更する必要があります。
- **callLink** エントリーの追加が必要になる場合があります。EGL では、次の状態で **callLink** エlementが必要です。
 - 生成された Java プログラムがネイティブの C++ または VAGen 生成のプログラムを呼び出す場合、そのプログラムが同じワークステーション上で実行中であっても、常にリモート呼び出しになります。**callLink** エントリーが必要です。
 - 呼び出し先プログラムに関して VAGen 生成オプション `/system=JAVAWRAPPER` を使用した場合は、**javaWrapper** プロパティを YES に設定した EGL **callLink** エントリーを作成する必要があります。エントリーがない場合、EGL は Java ラッパーを生成しません。
 - Java を生成し、呼び出し先プログラム名が EGL 予約語と競合する場合は、EGL **callLink** エントリーを作成して、**alias** プロパティを呼び出し先プログラムの実際の名前に設定する必要があります。
 - CICS に対して生成し、呼び出し先プログラムが PL/I プログラムなどの特別なリンケージを必要とする場合。例えば、PL/I プログラムを呼び出す場合、その呼び出しが VAGen プログラムまたは非 VAGen プログラムのどちらに対するものであるかには関係なく、VAGen デフォルト・リンケージは CICS LINK および COMMPTR です。EGL では、EGL プログラムへの呼び出しのデフォルト・リンケージは DYNAMIC および COMMPTR、非 EGL プログラムへの呼び出しのデフォルト・リンケージは CICS LINK および COMMPTR です。この場合、**call** ステートメントで **isExternal** = yes を指定しない場合は、**callLink** エントリーを作成して **linkType** = "CICS LINK"、**parmForm** = "COMMPTR"、および **pgmType** = "EXTERNALLYDEFINED" を設定し、PL/I プログラムに正しいリンケージが使用されるようにする必要があります。

- **fileLink** エlementでは、**conversionTable** = "BINARY" は、EGL ではサポートされません。マイグレーション・ツールは、この値を完全に現状のまま変換するので、EGL リンケージ・パーツ内にプレースホルダーができますが、この値は変更する必要があります。
- **asynchLink** エlement (VAGen crtlink) では、**conversionTable** = "BINARY" は、EGL ではサポートされません。マイグレーション・ツールは、この値を完全に現状のまま変換するので、EGL リンケージ・パーツ内にプレースホルダーができますが、この値は変更する必要があります。
- EGL の **transferToProgram** エlementは、VAGen の dxfrlink エントリーに相当します。Java を生成し、VAGen XFER ステートメントを使用する場合は、EGL の **transferToProgram** エントリーの追加が必要になることがあります。この新しいリンケージ・エントリーについては、オンライン・ヘルプを参照してください。

EGL リソース関連パーツの検討

マイグレーション・ツールは、VAGen リソース関連パーツを EGL リソース関連パーツに変換します。ただし、一部の VAGen オプションには同等な EGL オプションがありません。また、EGL には新しいリソース関連オプションがいくつかあり、これらの設定が必要になることがあります。これらの変更が原因で、「問題」ビューにエラーが表示される場合があります。マイグレーション・プロセスの結果として「問題」ビューに表示される共通メッセージの解決法については、519 ページの『付録 E. 「問題」ビューの IWN.xxx メッセージ』を参照してください。また、ご使用の環境に応じてサポートされるリソース関連オプションについて詳しくは、オンライン・ヘルプを参照してください。ステージ 1 から 3 までのマイグレーション、または単一ファイル・マイグレーションのいずれを使用した場合にも、次のタスクを実行する必要があります。

- マイグレーション・ログと「問題」ビューにあるメッセージを検討し、解決します。問題を解決するには、テキスト・エディターが必要になることがあります。
- VisualAge Generator のファイル・タイプの一部は、EGL ではサポートされません。例えば、BTRIEVE と MFCOBOL はサポートされなくなりました。マイグレーション・ツールは、これらのサポートされないオプションを完全に現状のまま変換するので、リソース関連パーツ内にプレースホルダーができます。このため、「問題」ビューにエラーが表示されます。このエラーを覚え書として、リソース関連パーツを変更し、EGL に対して使用するオプションを指定する必要があります。選択した EGL ファイル・タイプ・オプションによっては、リソース関連エントリーに関して追加のプロパティの設定が必要になることがあります。
- ご使用の環境に応じて、**formFeedOnClose** および **text** の各プロパティの設定が必要であるかどうかを判別するには、該当のオンライン・ヘルプを確認してください。VisualAge Generator の場合、同等なオプション (それぞれ /noff と /text) は、ワークステーション環境用のランタイム・リソース関連ファイル内でのみ指定できます。このため、これらのオプションはマイグレーション・ツールによって設定されません。これは、マイグレーション・ツールがリソース関連パーツのみを処理するからです。

テンプレートとして使用するバインド制御パーツの設定

注: このセクションは、z/OS 環境用に COBOL を生成する場合にのみ適用されます。

VisualAge Generator は、バインド制御テンプレートを使用してデフォルトのバインド制御コマンドを作成します。デフォルトの VAGen テンプレートは DB2 プランをバインドしますが、ユーザーがテンプレートを変更してパッケージをバインドするようにしたり、組織の標準に準拠するための変更を加えたりする場合があります。VAGen テンプレートは、ワークスペースの外部で EFK2MBDx.tpl という名前のファイルに保管されます。ここで、x は任意の文字です。バインド制御パーツは、特定プログラムに対して特殊バインドを行う必要がある場合のみ必須です。表 67 には、ランタイム環境およびデータベース・アクセスをベースにした VAGen バインド制御テンプレートが示されています。

表 67. VAGen バインド・コマンド・テンプレート

環境およびデータベース・アクセス	VisualAge Generator バインド・テンプレート
MVS CICS - および DB2	EFK2MBDA
MVS バッチ - および DL/I と DB2	EFK2MBDA
MVS バッチ - および DB2 のみ	EFK2MBDD
IMS/VS - および DL/I と DB2 作業データベース	EFK2MBDC
IMS/VS - および DL/I と DB2 (DB2 作業データベースつき)	EFK2MBDB
IMS/VS - および DL/I と DB2 (DB2 作業データベースなし)	EFK2MBDA
IMS BMP - および DL/I と DB2	EFK2MBDA

EGL は、外部定義のバインド制御テンプレートを使用しません。代わりに、EGL は内部テンプレートまたはバインド制御パーツを使用します。パッケージをバインドする場合は、すべてのバインドに使用するテンプレートを含む EGL バインド制御パーツを作成し、このパーツをワークスペースに保管することによって、VisualAge Generator テンプレートと同様な効果を得ることができます。

注: ここで説明する手法は、プランをバインドする場合には使用できません。プランをバインドする場合は、256 ページの『プログラム固有のバインド制御パーツの設定』を参照してください。

それぞれのプログラムごとにパッケージをバインドするように VAGen バインド制御テンプレートを変更した場合は、そのテンプレートを調整して EGL バインド制御パーツとして使用できます。以下の例のような、VAGen バインド制御テンプレートがあるとします。

```
DSN SYSTEM(%MYDB2SUBSYSTEM%)
BIND PACKAGE(%MYCOLLECTIONNAME%) -
  MEMBER(%EZEMBR%) -
  .
  .
  .
```


前の例の中で、MYDB2SUBSYSTEM と MYCOLLECTIONNAME は VAGen 生成オプションに設定したシンボリック・パラメーターで、EZEMBR は現在生成しているプログラムの名前に自動的に設定されます。

パッケージをバインドする場合、作成する必要がある EGL バインド制御パーツは、VAGen テンプレートと類似していますが、EZEMBR シンボリック・パラメーターに 3 つの行を追加し、1 つの変更を行う必要があります。対応する EGL バインド制御パーツは、以下の例のようになります。追加行および変更行は太字で強調表示されています。

```
TSOLIB ACTIVATE DA('%DSNLOAD%')
ALLOC FI(DBRMLIB) SHR DA('%EZEPID%.%SYSTEM%.DBRMLIB' +
'%ELA%.SELADBRM')
DSN SYSTEM(%MYDB2SUBSYSTEM%)
BIND PACKAGE(%MYCOLLECTIONNAME%) -
    MEMBER(%EZEALIAS%) -
    .
    .
    .
```

DSNLOAD、EZEPID、および ELA の意味は、すべて VisualAge Generator 内での意味と同じです。バインド制御パーツ内で生成するプログラムのランタイム名が必要な場合、EGL では EZEMBR の代わりに EZEALIAS が使用されます。EGL では、SYSTEM は EZEENV に置き換わります。ご使用の EGL ビルド・サーバー上では異なるデータ・セット命名規則を使用している場合は、バインド制御パーツの先頭 3 行を変更する必要があります。組織の命名規則に基づいてどのような 3 行を追加する必要があるか判断するには、EGL ビルド・サーバーをインストールして構成した担当者に問い合わせてください。また、VisualAge Generator 内で該当する値を設定していなかった場合は、EGL ビルド記述子オプションを変更して、**projectID** ビルド記述子オプション、および DSNLOAD と ELA の両シンボリック・パラメーターを設定する必要があります。シンボリック・パラメーターの名前の変更については、458 ページの『シンボリック・パラメーター』を参照してください。また、EGL バインド制御パーツに対するテンプレートの使用、および EGL シンボリック・パラメーターの値の設定について詳しくは、オンライン・ヘルプを参照してください。

テンプレートとして使用する EGL バインド制御パーツを作成するほかに、ビルド記述子パーツを変更して、バインド制御パーツを指定する **bind** ビルド記述子オプションを組み込む必要があります。変更が必要なビルド記述子パーツの数を最小限にするため、既存の共通ビルド記述子パーツのいずれかに **bind** ビルド記述子オプションを追加することを検討してください。

注: 任意のターゲット環境について、必ず VAGen バインド制御テンプレートの比較を行ってください。テンプレートが違う場合、相違点をサポートするために追加シンボリック・パラメーターを追加することができます。そのようにしない場合、別のターゲット環境用のテンプレートとして必要になる別の EGL バインド制御パーツを指し示すように、プログラム・ベースで別の **bind** ビルド記述子オプションの設定が必要となることがあります。

プログラム固有のバインド制御パーツの設定

注: このセクションは、z/OS 環境用に COBOL を生成する場合にのみ適用されます。

VisualAge Generator 内でプランをバインドする場合、通常はそれぞれのプログラムに異なるバインド・コマンドを使用する必要があります。この場合は、プログラム固有のバインド・コマンドを使用して、そのプログラムのプランを同じ実行単位内にある他のプログラムすべてにバインドする必要があります。このための標準的な方法としては、`xxxxx.BND` という名前のバインド制御パーツを作成します (ここで、`xxxxx` はプログラムの名前)。その後、VAGen 生成オプション `/BIND=BND` を設定して、プログラム固有のバインド・コマンドを検索する際に VisualAge Generator が使用する接尾部を指定します。まれな状況で、プログラムの 1 つが必要としているものがテンプレートの指定内容と異なるときにパッケージをバインドする場合にも、`.BND` 接尾部を使用できます。

EGL の **bind** ビルド記述子オプションには、接尾部を指定できません。代わりに、**bind** ビルド記述子オプションには、特定のバインド制御パーツの名前を指定する必要があります。EGL の場合、**bind** ビルド記述子オプションを指定しないと、EGL はプログラムと同じ名前のバインド制御パーツを検索します。一般に、最も簡単な手法としては、パッケージをバインドして、254 ページの『テンプレートとして使用するバインド制御パーツの設定』に説明する手順のとおりに行います。ただし、プランをバインドする場合、またはプログラムの 1 つがバインド制御パーツ・テンプレートの指定内容以外のものを必要としている場合には、プログラム固有のバインド制御パーツを作成できます。

デフォルトの `.BND` 接尾部を使用した VAGen プログラム固有のバインド制御パーツが存在する場合、マイグレーション・ツールは `.BND` 接尾部を自動的に除去し、EGL バインド制御パーツに必要な 3 つのステートメントを追加します。命名規則が `programName.BND` であり、常にプログラム固有のバインド・コマンド・パーツを使用していた場合、このプログラムに対しては EGL の **bind** ビルド記述子オプションを指定する必要はありません。ただし、EGL の **bind** ビルド記述子を使用して大部分のプログラムに対してテンプレートとして使用するバインド制御パーツを指定し、1 つのプログラムに対してはプログラム固有のバインド制御パーツを提供する必要がある場合は、この特定プログラム用にビルド記述子パーツを作成し、プログラム固有のバインド制御パーツを指し示すように **bind** ビルド記述子オプションを設定する必要があります。そのようにしないと、EGL は、通常の **bind** ビルド記述子オプションの指定に従って、テンプレートであるバインド制御パーツを選択します。

リンク・エディット・コマンドの検討

注: このセクションは、z/OS または VSE 環境用に COBOL を生成する場合にのみ適用されます。

VisualAge Generator は、ターゲット環境とデータベース・アクセスに基づいてデフォルトのリンク・エディット・コマンドを提供します。ただし、プログラムによっては、特定のリンク・エディット・コマンドを必要とする場合もあります。(例えば、MVS バッチ環境用に PL/I プログラムをリンクインする目的で。) このための

標準的な方法は、xxxxx.LKG という名前のリンク・エディット・パーツを作成することです (ここで、xxxxx はプログラムの名前)。その後、VAGen 生成オプション /LINKEDIT=LKG を設定して、プログラム固有のリンク・エディット・コマンドを検索するときに VisualAge Generator に使用させたい接尾部を指定します。

EGL の **linkEdit** ビルド記述子オプションには、接尾部を指定できません。代わりに、**linkEdit** ビルド記述子オプションに、特定のリンク・エディット・パーツの名前を指定する必要があります。EGL の場合、**linkEdit** ビルド記述子オプションが指定されないと、EGL はプログラムと同じ名前のリンク・エディット・パーツを検索します。EGL がプログラムと同名のリンク・エディット・パーツを検出できない場合は、VisualAge Generator と同様の方法で、ターゲット環境とデータベース・アクセスに基づいて EGL がデフォルトのリンク・エディット・コマンドを作成します。このため、**linkEdit** ビルド記述子オプションを指定する必要があるのは、プログラムとは異なる名前を指定してリンク・エディット・パーツを作成する場合に限ります。幾つかの COBOL 環境で同じプログラムを生成する場合に、これを行わなければならない可能性があります。

デフォルトの .LKG 接尾部を使用する VAGen プログラム固有のリンク・エディット・パーツが存在する場合、マイグレーション・ツールは自動的に .LKG 接尾部を除去します。命名規則が *programName.LKG* である場合、このプログラムに対して EGL の **linkEdit** ビルド記述子オプションを指定する必要はありません。EGL は、デフォルトのリンク・エディット・コマンドを作成する前に、まずプログラム固有のパーツを検索します。

ご使用の VGWebTransactions の検討

マイグレーション済み VGWebTransaction プログラムを検討する場合には、次の点について考慮してください。

- VGWebTransaction プログラムをデバッグする前に、すべての VGWebTransaction programs および VGUIRecord を生成する必要があります。
- EGL 生成の Bean をデプロイまたは使用する場合は、以下のタスクを実行する必要があります。
 - 新規の WAR ファイルに組み込む予定の VGUIRecords をすべて再生成します。
 - 対応するすべての VGWebTransaction プログラムを再生成します。
 - VGWebTransaction プログラムが **call**、**transfer**、または **show** の各ステートメント (VAGen CALL、DXFR、または XFER ステートメント) を使用して呼び出すかまたは転送する、すべての相手先プログラムをマイグレーションおよび生成します。

プログラム・リンクまたはハイパー・リンクで参照されるプログラムは、マイグレーションおよび生成の必要はありません。

- EGL パッケージ名が VAGen パッケージ名と異なる場合、JSP およびプロパティ・ファイルを更新する必要があります。以下のいずれかの理由により、パッケージ名が変更されている可能性があります。
 - パッケージが、EGL 予約語と競合するため、ステージ 1 のマイグレーション・ツール名前変更規則を使用した。

- パッケージを EGL に統合するため、ステージ 1 マイグレーション・ツール名
前変更規則を使用したか、ステージ 1 ツールを変更した。
- VAGen ソース・コードのパッケージ名と異なる可能性があるランタイム・パ
ッケージ名を指定するために、VAGen /packagename 生成オプションを使用し
た。EGL では、ランタイム・パッケージ名は通常、EGL ソースのパッケージ
名と同じです。

EGL で使用するよう変更した JSP のパッケージ名を更新する場合に役立つホワ
イト・ペーパーについては、19 ページの『参照』を参照してください。

- EGL Web トランザクションをデプロイする方法について詳しくは、オンライ
ン・ヘルプを参照してください。以下のタスクを必ず実行してください。
 - プロジェクトの src フォルダにあるデフォルトの gw.properties ファイルを
検討して変更します。 **hptEntryPage** および **hptEntryApp** の各値は必ず設定
してください。この情報は、ご使用の VisualAge Generator システム内の対応
する gw.properties ファイルからコピーできます。 VisualAge Generator の
gw.properties ファイルに加えた変更によっては、追加オプションの設定が必要
になる場合があります。オプション **hptDisableRMIDManager** が VisualAge
Generator フィックスパックに追加されています。このオプションを初めて使
用する場合は、EGL オンライン・ヘルプを参照し、値の設定の参考にしてくだ
さい。
 - プロジェクトの src フォルダにあるデフォルトの csogw.properties ファイル
を検討して変更します。 **serverLinkage** のエントリを更新し、パッケージ名
の変更を反映するように **javaProperty** 情報の変更が必要になる場合がありま
す。このとき、どのアプリケーションがどのサーバーにあるのかを指定する情
報を必ず組み込んでください。この情報は、ご使用の VisualAge Generator シ
ステム内の対応する csogw.properties ファイルからコピーできます。
VisualAge Generator の csogw.properties ファイルに加えた変更によっては、追
加オプションの設定が必要になる場合があります。
 - プロジェクトの WebContent フォルダにあるデフォルトの
Vagen1EntryPage.jsp を検討して変更します。 **hptAppId** の OPTION 情報を更
新し、リストの各プログラム用に表示する VGWebTransaction プログラムの名
前および関連するテキストを必ず含むようにしてください。この情報は、ご使
用の VisualAge Generator システム内の対応する JSP ファイルからコピーでき
ます。
 - プロジェクトを生成します。
 - 以下のステップに従って、エンタープライズ・アプリケーション・リソース
(EAR) プロジェクトを作成します。
 1. 「プロジェクト・エクスプローラー」ビューの「ワークベンチ」ウィンド
ウから、「新規」->「その他」->「J2EE」->「エンタープライズ・アプリ
ケーション・プロジェクト」をクリックする。
 2. EAR プロジェクトの「名前」を入力する。
 3. 「次へ」をクリックします。
 4. EAR プロジェクトにインクルードするプロジェクトを選択する。
 5. 「終了」をクリックします。
 - Web アプリケーション・サーバーを定義する。
 - EAR プロジェクトをサーバーに追加する。

- プロジェクトの WebContent フォルダにある EGLWebStartup.jsp を右クリックし、「実行」->「サーバーで実行」をクリックしてアプリケーションを実行する。

デバッグの準備

デバッグを準備するには、以下のタスクを実行する必要があります。

- 「ワークベンチ」ウィンドウから、「ウィンドウ」->「設定」->「EGL」->「デバッグ」をクリックします。ご使用の環境に応じて行う必要がある設定 (あれば) を判別するには、オンライン・ヘルプを参照してください。例えば、次のようになります。
- 「**EBCDIC で実行 (Run in EBCDIC)**」モードの VAGen テスト設定を使用した場合、ホスト・システムの EBCDIC コード・ページに EGL デバッグ設定「**文字エンコード**」を設定する必要があります。
- また、「**プログラムの先頭行で停止**」および「**ホット・スワップを使用可能にする (Enable hotswapping)**」も選択できます。
- VAGen テスト設定「**バイパスするライブラリー内のプログラム (Programs in Library to Bypass)**」を使用した場合は、「EGL」->「デバッグ」->「**振り舞いマッピングのデバッグ**」設定ページに情報を設定する必要があります。詳しくは、オンライン・ヘルプを参照してください。
- さらに、「EGL」->「**デフォルトのビルド記述子**」の設定を検討します。ワークスペース全体に対してデフォルトの「**デバッグ・ビルド記述子**」を設定できます。また、プロジェクト、EGL ソース・フォルダー、パッケージ、またはファイルに対して、デフォルトの「**デバッグ・ビルド記述子**」を設定することもできます。
- 生成した EGL プログラムまたは非 EGL プログラムをリモート CICS システム上でデバッガーから呼び出す場合は、CICS Transaction Gateway インフラストラクチャーを設定して構成する必要があります。CICS Client 製品を使用した CICS への直接呼び出しはサポートされなくなりました。
- DL/I データベース I/O を使用するプログラムに EGL デバッガーを使用する場合、またはリモート IMS システム上のプログラムを呼び出す場合は、複数のファイルを構成する必要があります。詳しくは、「*EGL プログラマー・ガイド*」を参照してください。
- z/OS でリモート VSAM ファイルを使用している場合は、Distributed File Manager (DFM) をインストールして使用する必要があります。DFM のインストール方法、およびデバッグ用のリソース関連エントリーの指定方法について詳しくは、「*EGL Generation Guide*」を参照してください。

zSeries に対する EGL サーバー製品のインストール

注: このセクションは、z/OS 環境用に COBOL を生成する場合にのみ適用されます。

z/OS の場合、EGL サーバーは独立した SMP/E ゾーンにインストールし、MVS、VSE、および VM (VAGen サーバー製品) の VisualAge Generator Server とは異なるターゲット・ライブラリーを指定する必要があります。

VAGen サーバー製品ロード・モジュールのいずれかを LPA に配置した場合は、マイグレーションが完了する前に、それを EGL サーバー製品ロード・モジュールに置き換える必要があります。予測不能の問題が発生する可能性があるため、この変更を実行する場合は、VAGen サーバー製品からのモジュールと EGL サーバー製品からのモジュールが混在しないようにする必要があります。そのため、LPA にロード・モジュールを最初に配置したときに VAGen SELALMD ロード・ライブラリーから VAGen サーバー製品ロード・モジュールを除去した場合は、ロード・モジュールのセットの整合性を保つために以下のステップに従ってください。

1. VAGen の各モジュールを VAGen SELALMD ロード・ライブラリーに戻します。
2. LPA から VAGen ロード・モジュールを除去します。
3. LPA に EGL サーバー製品のロード・モジュールを追加します。
4. EGL SELALMD ロード・ライブラリーから EGL サーバー製品のロード・モジュールを除去します。

EGL は準備プロセスにビルド・サーバーを使用しますが、EGL サーバー製品には VAGen サーバー製品と同じ JCL プロシーチャーが含まれています。VAGen プログラムの生成を継続する予定であるが、EGL ランタイム・サーバーを使用する場合は、EGL 準備 JCL プロシーチャーを調整して (または既存の VAGen 準備 JCL プロシーチャーを再調整して)、EGL ランタイム・サーバー・ライブラリーを指し示すようにする必要があります。

EGL サーバー製品は、VAGen 生成のプログラムとも互換性があります。そのため、ソース・コードを EGL にマイグレーションする前に、EGL サーバー製品にマイグレーションできます。例えば、IMS または CICS 領域は、以下のようにしだいに大きくなります。

- VAGen 生成のプログラムがある VAGen サーバー・ロード・ライブラリーおよびアプリケーション・ロード・ライブラリーを含む実動領域。実動領域と同等のテスト領域。
- VAGen 生成のプログラムがある VAGen サーバー・ロード・ライブラリーおよびアプリケーション・ロード・ライブラリーを含む実動領域。VAGen 生成のプログラムがある EGL サーバー・ロード・ライブラリーおよびアプリケーション・ロード・ライブラリーを含むテスト領域。これにより、既存の VAGen プログラムが以前と同様に実行できるようになります。オプションで、EGL 生成のプログラムがある EGL サーバー・ロード・ライブラリーおよびアプリケーション・ロード・ライブラリーを含む第 2 テスト領域。これにより、パイロット・マイグレーション時にプログラムをテストできます。
- VAGen 生成のプログラムがある EGL サーバー・ロード・ライブラリーおよびアプリケーション・ロード・ライブラリーを含む実動領域。VAGen 生成のプログラムがある EGL サーバー・ロード・ライブラリーおよびアプリケーション・ロード・ライブラリーを含むテスト領域。これにより、パイロット・マイグレーション時に VisualAge Generator でのプログラムの開発およびマイグレーションを継続できます。EGL 生成のプログラムがある EGL サーバー・ロード・ライブラリーおよびアプリケーション・ロード・ライブラリーを含む第 2 テスト領域。これにより、パイロット・マイグレーション時に EGL プログラムをテストできます。

- VAGen 生成および EGL 生成の各プログラムが混在した EGL サーバー・ロード・ライブラリーを含む実動領域。 VAGen 生成および EGL 生成の各プログラムが混在した EGL サーバー・ロード・ライブラリーを含むテスト領域。テスト領域には、実動領域よりも多数の EGL 生成のプログラムがあります。同時にすべてのソース・コードを EGL にマイグレーションする場合は、この構成の領域をスキップできます。
- EGL 生成のプログラムがある EGL サーバー・ロード・ライブラリーを含む実動領域。実動領域と同等のテスト領域。

VSE に対する EGL サーバー製品のインストール

注: このセクションは、VSE 環境用に COBOL を生成する場合にのみ適用されます。

VSE の場合、IBM Rational COBOL Runtime for z/VSE をインストールする必要があります。詳しくは、プログラム・ディレクトリーを参照してください。

VAGen 準備テンプレートおよびプロシーチャーの EGL ビルド・スクリプトへの変換

注: このセクションは、COBOL を生成する場合にのみ適用されます。

VisualAge Generator では、COBOL 生成の場合に、準備プロセスを制御するために準備テンプレートが使用されます。使用されるテンプレートは、生成されるパーツ、ランタイム環境、および (プログラム・パーツについては) データベース・アクセスのタイプによって決まります。生成オプション /TEMPLATES は、準備テンプレートが格納されるドライブおよびディレクトリーを指定します。テンプレートは、以下のように環境によってさまざまです。

- MVS ランタイム環境の場合、DB2 プリコンパイル、CICS 変換、COBOL コンパイル、リンク・エディット、およびバインドの実行に必要な JCL を生成するために、準備テンプレートが使用されます。準備テンプレートにより、準備プロセスの実際のステップを提供する JCL プロシーチャーが呼び出されます。
- VSE ランタイム環境の場合、DB2 プリコンパイル、CICS 変換、COBOL コンパイル、およびリンク・エディットの実行に必要な JCL を生成するために、準備テンプレートが使用されます。準備テンプレートにより、準備プロセスの実際のステップを提供する JCL プロシーチャーが呼び出されます。
- OS/400 ランタイム環境の場合、準備テンプレートはコンパイルおよびバインドを実行する制御言語 (CL) の生成に使用されます。

EGL では、準備プロセスは以下のように処理されます。

- zSeries および iSeries の場合、準備は EGL ビルド・サーバーによって処理されます。EGL ビルド・スクリプトにより、それまで VAGen の準備テンプレートおよび準備プロシーチャーに含まれていた情報がマージされます。
- VSE の場合、準備プロセスは引き続き準備テンプレートおよび準備プロシーチャーによって処理されます。詳しくは、VSE 用の生成フィーチャーの資料を参照してください。

zSeries の場合は、ご使用の VAGen 準備テンプレートおよびプロシージャーを検討し、それをカスタマイズするかどうか決定します。カスタマイズする場合、EGL ビルド・スクリプトを変更してデバッグするには、以下のステップを実行します。

1. ビルド記述子オプション **prep** を NO に設定します。
2. ご使用のランタイム環境用にプログラムを生成します。生成された準備ファイルを検討し、このタイプのプログラム、データベース・アクセス、およびランタイム環境に使用されているビルド・スクリプトの名前を判別します。
3. z/OS ホストで使用する予定のデータ・セットに生成の出力を手動でアップロードします。
4. そのプログラム用の準備 JCL を作成します。開始点としてご使用の VAGen 準備テンプレートおよびプロシージャーを使用し、EGL データ・セットを指定するようにそれを変更できます。また、準備テンプレートおよびプロシージャーを EGL で提供されているビルド・スクリプトと比較して、EGL で必要な追加のステップやライブラリーなどがあるかどうか判別します。
5. 正しく変更されていることが確認されるまで、準備 JCL をテストします。
6. 準備 JCL を疑似 JCL に変換し、ビルド・スクリプトで使用できるようにします。詳しくは、「*IBM Rational COBOL Runtime for zSeries ガイド*」を参照してください。
7. ビルド記述子オプション **prep** を YES に設定します。
8. ご使用のランタイム環境用にそのプログラムを再度生成します。生成の出力が、ビルド・サーバーを使用して自動的にアップロードされ準備されます。
9. それぞれのランタイム環境でデータベースの各タイプにアクセスするプログラムごとに、このプロセスを繰り返します。各ランタイム環境では別々のビルド・スクリプトを使用するため、必ずすべてのランタイム環境用に FormGroups (テキストおよび印刷書式の両方) および DataTables を生成してください。

zSeries の場合には、VAGen テンプレートおよびプロシージャー、および対応する EGL ビルド・スクリプトのリストについて、461 ページの『準備テンプレートおよびプロシージャー』も参照してください。VisualAge Generator は、MVS ランタイム環境用にバインド制御テンプレートも使用します。MVS バインド制御テンプレートの変換方法について詳しくは、254 ページの『テンプレートとして使用するバインド制御パーツの設定』を参照してください。

iSeries の場合、ビルド・スクリプトのカスタマイズ方法については、「*Rational Business Developer EGL Server Guide for IBM i*」を参照してください。

VAGen ランタイム・テンプレートの変換

注: このセクションは、COBOL を生成する場合にのみ適用されます。

VisualAge Generator では、ランタイム・テンプレートが以下のコードの生成に使用されます。

- MVS バッチおよび IMS BMP ランタイム環境用のサンプル・ランタイム JCL。
- VSE バッチ環境用のサンプル・ランタイム JCL。
- iSeries 環境用のサンプル制御言語。

生成オプション /TEMPLATES は、ランタイム・テンプレートが格納されるドライブおよびディレクトリーを指定します。使用されるテンプレートは、プログラムのタイプ、ランタイム環境、およびファイルまたはデータベース・アクセスのタイプによって決まります。以下のタイプのランタイム JCL テンプレートが、MVS バッチ、IMS BMP、および VSE に使用されます。

- サンプル・ランタイム JCL の主要部分を作成する実行 JCL テンプレート。
- DD ステートメントを作成する、ファイルおよびデータベースの割り振りテンプレート。これらの DD ステートメントは、ランタイム環境、およびリソース関連情報によって指定されたファイル実装に基づいて、DL/I データベース、シリアル、索引付き、または相対の各ファイル用のサンプル JCL に組み込まれます。
- コメントを生成するファイルおよびデータベースの割り振りプレースホルダー・テンプレート。これらのコメントは、プログラムが操作を行うために追加の DD ステートメントが必要であるが、それを生成するために必要な情報がない場合に、サンプル JCL に組み込まれます。例えば、ProgramA が ProgramB を呼び出す場合、ProgramA を生成する時点では、ProgramB でどのような DD ステートメントが必要であるのかを判別できないことがあります。ファイルおよびデータベースの割り振りプレースホルダー・テンプレートは、ProgramA のサンプル・ランタイム JCL にコメントを組み込み、それが ProgramB を呼び出すことを示します。

同様に、OS/400 の場合にも、いくつかのランタイム CL テンプレートがあります。

また、EGL にも、ランタイム・テンプレートがあります。ビルド記述子オプション **templateDir** は、テンプレートがあるディレクトリーを指定します。デフォルトのディレクトリーは、共用する製品インストール・ディレクトリー内の以下のサブディレクトリーです。

```
plugins\com.ibm.etools.egl.generators.cobol_version\MVStemplates
plugins\com.ibm.etools.egl.generators.cobol_version\VSEtemplates
plugins\com.ibm.etools.egl.generators.cobol_version\iSeriesTemplates
```

version は、インストールした COBOL 生成の最新バージョン・レベルです。

EGL ランタイム JCL または CL のテンプレートの調整が必要な場合は、以下の点について考慮してください。

- 製品インストール・ディレクトリーの外側にユーザー独自のディレクトリーを作成します。これにより、カスタマイズしたテンプレートが上書きされる心配がなくなり、EGL 保守のインストールが簡単になります。
- すべての開発者がアクセスできるように、このディレクトリーを共用ドライブに配置することを検討します。これにより、すべての開発者のワークステーションに新規のテンプレートを配布する必要がなくなり、テンプレートの変更が容易になります。
- VAGen ランタイム・テンプレートおよび対応する EGL ランタイム・テンプレートのリストについては、以下の情報を参照してください。
 - zSeries および iSeries の場合は、463 ページの『ランタイム・テンプレート』を参照してください。

- VSE 情報については、「*Rational Business Developer V7.5 Generation for z/VSE feature Reference Manual*」(SC19-2539-00) を参照してください。

ご使用の VAGen ランタイム・テンプレートを開始点として使用し、各テンプレートを対応する EGL テンプレートと比較して、どのような調整が必要であるのかを判別します。VAGen シンボリック・パラメーターは、458 ページの『シンボリック・パラメーター』に示されているように、必ず対応する EGL シンボリック・パラメーターに変更してください。

VAGen 予約語ファイルの変換

注: このセクションは、COBOL を生成する場合にのみ適用されます。

VisualAge Generator では、生成オプション /RESVWORD によってオプションの予約語ファイルのドライブ、ディレクトリ、およびファイル名が指定されます。このファイルには、生成される COBOL プログラム内のパーツ名またはフィールド名として使用できない、すべての COBOL、SQL、および CICS の予約語が含まれています。デフォルトの予約語ファイルは、VisualAge Generator に付属しています。生成オプション /RESVWORD を指定した場合は、おそらくリストに追加の語を追加しています。予約語リストを変更していない場合は、このセクションをスキップできます。

EGL では、標準の COBOL、SQL、および CICS の予約語は、COBOL 生成プログラムで事前定義されています。EGL の予約語ファイルには、ユーザーがリストに追加した予約語のみが含まれます。

EGL 予約語リストへの追加が本当に必要であるかどうかを検討してください。必要である場合は、必要な追加の単語のリストのファイルをワークステーションに作成します。次に、EGL ビルド記述子パーツを変更して **reservedWord** ビルド記述子オプションを組み込み、このオプションで予約語ファイルを指定します。予約語ファイルを各開発者のワークステーションに配布するのではなく、共用ドライブに配置して、全員がその単一のファイルにアクセスできるようにすることができます。この手法により、予約語ファイルの変更が容易になります。

COBOL 生成を行う場合の生成とテスト

COBOL の生成を準備するには、以下のタスクを実行します。

- この時点で、ソース・コード・リポジトリにある EGL プロジェクトとパッケージの別のバージョンを作成できます。これは、生成の前に手動で行った変更を反映した、コードのもう 1 つのベースラインになります。
- ご使用の環境の EGL ランタイム・サーバーに最新の PTF がすべてインストールされていることを確認します。
- z/OS および iSeries の場合、EGL ビルド・サーバーに最新の PTF がすべてインストールされていることを確認します。また、それがホスト環境に構成されていることも確認します。VisualAge Generator では、ワークステーション上の準備テンプレートを変更することにより、z/OS および iSeries の準備プロセスに対するカスタマイズが実行されていました。EGL の場合、このカスタマイズはホスト・マシン上で行われます。詳しくは、261 ページの『VAGen 準備テンプレートおよびプロシーチャーの EGL ビルド・スクリプトへの変換』を参照してください。

- VSE の場合、準備 JCL テンプレートおよび準備 JCL プロシーチャーをカスタマイズします。
- EGL ビルド・サーバーのインストールと構成を行った担当者にお問い合わせください。生成と準備の出力を格納するホスト・データ・セットの命名規則に変更があるかどうか確認してください。例えば、VisualAge Generator では、MVS バッチ用に生成する場合、データ・セットのデフォルト名は `xxxx.MVSBATCH.yyyy` です (ここで、`xxxx` は `/projectid` 生成オプションに指定した高位修飾子であり、`yyyy` はコードのタイプです)。 (例: COBOL ソースの場合は `EZESRC`。) EGL の場合は、ターゲット環境名が変更されているため、対応するデフォルト・データ・セット名は `xxxx.ZOSBATCH.yyyy` です。このため、ホスト上でデータ・セットのグループを新規に定義する必要が生じることがあります。
- プログラムおよび DataTable を生成します。プログラムを生成する際には、次のビルド記述子オプションを使用します。

- `genFormGroup="YES"`
- `genHelpFormGroup="YES"`
- `genDataTables="NO"`

これにより、FormGroup を使用するプログラムとともに FormGroup を生成できるようになりますが、DataTable を使用するプログラムの数に関係なく、各 DataTable の生成は一度のみ可能です。生成中に検出された検証エラーは、すべて解決してください。

注: どのような場合に EGL プログラムの生成または VAGen プログラムの再リンクが必要になるのかについて詳しくは、565 ページの『付録 I. VisualAge Generator および EGL インターオペラビリティ』を参照してください。

- VAGen および EGL は、COBOL のレコードおよび項目を別々の名前で作成します。EGL の名前のほうが、読みやす COBOL プログラムが生成されますが、ご使用の SQL 製品によっては、生成された COBOL プログラムが SQL ステートメント・プリコンパイラーの制限を超過する可能性があります。以下に、プリコンパイラーの制限の例を示します。
 - 処理行の最大数。すべての SQL ステートメントは、この制限より前のプログラムに配置する必要があります。COBOL 生成では、手続き部のなるべく前に SQL ステートメントが配置されます。ただし、多数の SQL 関数がある場合や、レコードまたはフォームに多数のデータ項目がある場合に、プログラムがこの制限に直面することがあります。
 - 固有のホスト変数の最大数。NULL が許可された各ホスト変数には、最大値までカウントが割り振られた標識変数があります。デフォルトでは、VAGen から EGL へのマイグレーション・ツールは、SQL レコードの各フィールドについて `isSQLNullable` プロパティを YES に設定し、VAGen の振る舞いを保持します。
 - SQL ステートメントの行または文字の最大数。

SQL プリコンパイラーの制限のいずれかを超過する場合は、プログラムに以下の 1 つ以上の変更を加える必要があります。

- SQL テーブルの一部の列が NOT NULL として定義されている場合は、EGL SQL レコード定義の対応するフィールドからプロパティ `isSQLNullable = yes` を除去します。これにより、固有ホスト変数の数が削減され、SQL ステートメント

トメントの文字数および行数、およびプログラムの合計行数を減少させることになります。この手法は、作業量を最小に抑える大きな効果があり、ランタイム・パフォーマンスが向上する可能性もあります。

- デフォルトの SQL ステートメントの使用方法を検討します。デフォルト・ステートメントが、実際に必要であるよりも多くの列を検索している場合、必要な列だけを指定するようにステートメントを変更してください。
- SQL レコードの名前を短縮します。
- SQL ステートメントを複数のステートメントに分割します。例えば、1 つの **get** ステートメントを複数の **get** ステートメントに変更して、各ステートメントで列のサブセットを取得します。
- プログラムを複数のプログラムに分割します。
- 生成されたコードをテストします。
- この時点で、ソース・コード・リポジトリにある EGL プロジェクトとパッケージの別のバージョンを作成できます。これは、生成およびテスト中に検出された問題に対処するために行った変更を反映した、コードのもう 1 つのベースラインになります。

Java 生成を行う場合の生成とテスト

Java の生成を準備するには、以下のタスクを実行します。

- この時点で、ソース・コード・リポジトリにある EGL プロジェクトとパッケージの別のバージョンを作成できます。これは、生成の前に手動で行った変更を反映した、コードのもう 1 つのベースラインになります。
- プログラムおよび DataTable を生成します。プログラムを生成する際には、次のビルド記述子オプションを使用します。

- **genFormGroup="YES"**
- **genHelpFormGroup="YES"**
- **genDataTables="NO"**

これにより、FormGroup を使用するプログラムとともに FormGroup を生成できるようになりますが、DataTable を使用するプログラムの数に関係なく、各 DataTable の生成は一度のみ可能です。生成中に検出された検証エラーは、すべて解決してください。

注: どのような場合に EGL プログラムの生成または VAGen プログラムの再リンクが必要になるのかについて詳しくは、565 ページの『付録 I. VisualAge Generator および EGL インターオペラビリティ』を参照してください。

- VAGen 製品のメッセージ・テキストを変更した場合は、EGL メッセージ・テキストに対して同様な変更を行うことができます。
- 生成されたコードをテストします。
- この時点で、ソース・コード・リポジトリにある EGL プロジェクトとパッケージの別のバージョンを作成できます。これは、生成およびテスト中に検出された問題に対処するために行った変更を反映した、コードのもう 1 つのベースラインになります。

標準の検討

現行のコーディング標準を検討して、作成される新規コードに使用する新しい標準を設定する必要があることがあります。例えば、COBOL の生成を行う場合は、次のような標準を検討してください。

- パーツ名の中で、ハイフンの代わりに下線を使用する。COBOL プログラムは、名前に下線を使用することを許容しません。ただし、COBOL 生成によって下線が自動的にハイフンに変更されるので、生成された COBOL は引き続き読み取り可能です。下線をハイフンに変更した後、重複するパーツ名または変数名が存在する場合に限り、別名が割り当てられます。
- 生成される COBOL コードを読みやすくするために、生成によって別名が割り当てられるような名前の使用を避けます。このためには、次の命名規則を使用します。
 - レコード名と関数名は 18 文字以下にする必要があります。これは、VisualAge Generator 内での制限と同じです。
 - データ項目名は 27 文字以下にする必要があります。これは、COBOL 生成に関する VAGen 推奨制限である 30 文字、および VisualAge Generator の最大文字数である 32 文字よりわずかに少ない文字数です。
- プログラム名と DataTable 名は、8 文字にすることができます。

ソース・コードの二重メンテナンスの計画

VAGen ソース・コードは、EGL にマイグレーションできます。ただし、EGL ソース・ファイルは VisualAge Generator にマイグレーションして戻すことはできません。すべての VAGen ソース・コードを同時にマイグレーションするのではなく、VAGen および EGL コードの両方で必要となるパーツを用意する場合は、以下のいずれかの方法で共通パーツを変更する必要があります。

- VAGen および EGL の両方のバージョンのパーツを変更します。
- VAGen を変更した後、そのパーツ用に外部ソース・フォーマット・ファイルをエクスポートします。変更内容によっては、パーツ間のマイグレーションを実行可能にするため、変更されたパーツで使用するパーツ、および変更されたパーツを使用するパーツを外部ソース・フォーマット・ファイルに組み込む必要があります。外部ソース・ファイルを EGL の新規ファイルにインポートします。次に、新規にマイグレーションしたパーツの変更をオリジナルの EGL パーツと比較し、その変更を EGL ワークスペース内の正しいロケーションに移動します。

二重メンテナンス問題の最も簡単な解決策は、以下の方法によって問題を完全に回避することです。

- EGL へのマイグレーションの間は、VAGen の開発およびメンテナンスを凍結します。
- 現在進行中のすべての VAGen プログラムを生成してテストし、それらを実動に移行します。これにより、実動バージョンのソース・コードのみをマイグレーションできます。

同時にすべてをマイグレーションできない場合は、変更の対象を少数のパーツに限定し、その後にパーツの VAGen および EGL の両バージョンを変更します。この両方への変更は、VAGen のパーツを変更してそれを再度 EGL にマイグレーションするよりも簡単に行えることがあります。これは、パーツ間のマイグレーションに

必要なすべての関連パーツを特定する必要がなく、新規にマイグレーションしたパーツを EGL ワークスペース内の正しいロケーションに移動する必要がないためです。

VisualAge Generator 互換モードの使用の中止

VisualAge Generator 互換モードでは、VAGen プログラムの振る舞いを容易に保持できるように、さまざまな振る舞いをサポートしています。VisualAge Generator 互換モードをオフにする必要はありません。ただし、特に、使用する互換性の振る舞いが少ない場合は、VisualAge Generator 互換モードの使用を中止できます。次のリストでは、VisualAge Generator 互換モードをオンにした場合の EGL の振る舞いについて説明し、この振る舞いを使用する必要がなくなるようにするために役立つ情報を示します。

- EGL は、パーツ名の中でハイフン (-) と各国語文字 @ および # の使用を許容します。VisualAge Generator 互換モードの使用を中止する予定の場合は、ステージ 2 マイグレーション時に使用する**名前変更ユーザー出口**を作成する必要があります。**名前変更ユーザー出口**を作成して、ハイフン、@、および # が使用されないようにします。例えば、VAGen パーツ名に下線を使用することがない場合は、ハイフンを下線に変更してパーツ名を変更する**名前変更ユーザー出口**を作成できます。ステージ 2 マイグレーションでの**名前変更ユーザー出口**の指定方法について詳しくは、211 ページの『VAGen マイグレーションの設定』を参照してください。VisualAge Generator 互換モードをオフにすると、ハイフン、@、または # を含むすべてのパーツ名または変数名に関して、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。名前を変更することによって、この問題を訂正できます。出口ルーチンの作成例については、19 ページの『参照』にリストされている、**名前変更ユーザー出口**の使用に関するホワイト・ペーパーを参照してください。
- EGL では、プリミティブ・データ型 NUMC および PACF の使用が許可されます。NUMC は NUM と類似していますが、NUMC では正符号インディケータとして C を使用する点が異なります。PACF は DECIMAL (VAGen PACK) と類似していますが、PACF では正符号インディケータとして F を使用する点が異なります。VisualAge Generator 互換モードをオフにすると、NUMC または PACF のプリミティブ型を指定するすべてのデータ項目定義または変数宣言に関して、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。この問題は、プリミティブ型を NUM または DECIMAL にそれぞれ変更し、新規の正符号正符号インディケータに必要な、関連するプログラム変更を行うことにより、訂正できます。NUMC または PACF フィールドの現在の使用方法によっては、データベース、ファイル、または DataTables 内のデータの変更も必要になる場合があります。
- EGL は、1 次元の構造化フィールド配列の添え字をデフォルトで 1 に設定します。1 次元の構造化フィールド配列は、VAGen の配列、あるいはレコード、マップ、またはテーブル内で複数回出現する項目の EGL 置換表現です。VisualAge Generator 互換モードをオフにすると、添え字が必要となった構造化フィールド配列を指定するすべてのステートメントについて、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。この問題を訂正するには、添え字 1 を明示的に指定するようにステートメントを変更する必要があります。
- EGL では、DataTable の **use** 宣言で **deleteAfterUse** プロパティーが許可されます。プロパティー **deleteAfterUse** は、VAGen のプロパティー **Keep After Use** の

置換表現です。 VisualAge Generator 互換モードをオフにすると、**deleteAfterUse** を指定する **use** 宣言を組み込んだ各プログラムに対して、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。この問題は、プロパティ **deleteAfterUse** を除去することにより訂正できます。このプロパティを除去すると、EGL は、Keep After Use = yes を指定した VAGen テーブルと同様にテーブルを処理します。

ご使用の実動プロダクションが、既に VisualAge Generator バージョン 4.5 フィックスパック 4 以降を使用して生成されている場合は、EGL のプロパティ **deleteAfterUse** を除去することによる影響はありません。システム共通プロダクトまたは以前のバージョンの VisualAge Generator からマイグレーションしている場合は、プロパティ **deleteAfterUse** を除去するように変更するすべてのプログラムを十分にテストする必要があります。

テーブルのマイグレーション設定「**deleteAfterUse** を組み込まない (Do not include deleteAfterUse)」を選択すると、マイグレーション・ツールは自動的にプロパティ **deleteAfterUse** を省略し、影響を受けるプログラムおよびテーブルに関する警告メッセージを出します。

- EGL は、**display printForm** ステートメントを **print printForm** ステートメントと同じ方法で実装します。マイグレーション・セットにマップを組み込むことにより、**display printForm** の使用を最小限にできます。これにより、マイグレーション・ツールは、テキスト・マップの場合は **display** ステートメントに、プリンター・マップの場合は **print** ステートメントに正しくマイグレーションできます。 VisualAge Generator 互換モードをオフにすると、各 **display printForm** ステートメントごとに、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。この問題は、ステートメントを **print** ステートメントに変更することにより訂正できます。
- EGL では、値が割り当てられていないフィールドを画面に表示するときのみ、書式フィールドの **value** プロパティを使用します。 **value** プロパティは、ストレージ内の書式フィールドの初期値を設定しません。マイグレーション・ツールは、マップのマイグレーション時に **value** プロパティを組み込みます。 VisualAge Generator 互換モードをオフにすると、EGL は **value** プロパティを使用してストレージ内の書式フィールドの初期値を設定します。「問題」ビューにエラーが示される場合と示されない場合があります。例えば、VisualAge Generator では、DISPLAY または CONVERSE 入出力オプションの前にプログラムが数値マップ変数フィールドにデータを移動しない場合でも、マップ・エディターのプレビュー・モードの値を指定したりユーザーに出力を提供したりできるように、数値マップ変数フィールドの初期値が「MM/DD/YYYY」になっていることがあります。このような場合、VisualAge Generator 互換モードをオフにすると、値が NUM プリミティブ型と互換ではないため、「問題」ビューにメッセージが示され、ランタイム・エラーも発生することがあります。ただし、初期値が 5 のような数値である場合は「問題」ビューにメッセージは表示されませんが、プログラムは VisualAge Generator とは異なる振る舞いをする可能性があります。 VisualAge Generator 互換モードをオフにした場合は、「問題」ビューに表示されるメッセージを訂正することに加えて、EGL 書式の **value** プロパティを検索し、振る舞いの変化を回避するために必要なプログラムの変更を特定する必要があります。この変更には、書式フィールドの **value** プロパティの除去や変更などがあります。

- EGL では、SQL の WHERE 文節および EGL の **prepare** ステートメント内のホスト変数参照の場合を除き、精度を 1 つ増やすことによって DECIMAL フィールド (VAGen PACK フィールド) の偶数精度をサポートしています。VisualAge Generator 互換モードをオフにすると、EGL 検査は「問題」ビューにエラー・メッセージを表示しません。ただし、プログラムは VisualAge Generator の場合と異なる動作をする可能性があります。具体的には、偶数精度の DECIMAL フィールドが、データベースに格納されているデータを表すには小さすぎる可能性があります。一般的に、VisualAge Generator 互換モードを安全にオフにできるかどうかを判断するためには、各データ項目パーツまたは変数を慎重に評価する必要があります。VisualAge Generator では、「参照」ツールを使用してテキスト・ストリング *evensql = Y* (= 記号の前後にブランクを 1 つ入れる) を検索することにより、すべてのデータ項目およびレコード・パーツを検索できます。この検索により、偶数精度を指定した項目またはレコードがあるかどうか判別できます。EGL では、*decimal(2, decimal(4, decimal(18* を検索することによって、偶数精度の DECIMAL フィールドを検索できます。偶数精度の DECIMAL フィールドを使用していない場合は、VisualAge Generator 互換モードを安全にオフにできます。偶数精度の DECIMAL フィールドを使用していた場合は、次に大きな奇数精度への変更が SQL アクセスのパフォーマンスに及ぼす影響を検討する必要があります。EGL ホスト変数が SQL 列定義の精度と正確に一致している場合に、SQL は DECIMAL フィールドに関して良好なパフォーマンスを発揮します。

マイグレーション設定「項目または変数の *evensql=y* を受け入れない」を選択すると、マイグレーション・ツールは自動的に奇数精度 (項目が最大長である場合は 18) を使用し、影響を受けるデータ項目パーツや非共用レコード項目に関して警告メッセージを出します。

- EGL は、システム変数 **converseVar.segmentedMode** の使用を許可します。これは、メインの CICS トランザクション・プログラム内で、実行時にセグメント化モードと非セグメント化モードの切り替えを可能にする、VAGen EZESEGM の置換表現です。VisualAge Generator 互換モードをオフにすると、**converseVar.segmentedMode** を使用する各ステートメントごとに、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。EZESEGM が使用されることはまれなので、多くの場合、「問題」ビューにエラーは示されません。エラーがある場合、**converseVar.segmentedMode** の使用を中止するには、セグメント化モードと非セグメント化モードの切り替えが必要ないように、プログラムの再構築が必要になる場合があります。
- EGL は、システム関数 **vgLib.getVAGSysType()** の使用を許可します。マイグレーション・ツールは、*customerPrefixEZESYS* という名前の変数を宣言し、それぞれのプログラム内でこの変数を **vgLib.getVAGSysType** の結果に初期化します。*customerPrefix* は、ステージ 2 マイグレーション時に指定した名前変更接頭部です。**vgLib.getVAGSysType** は、IF、WHILE、または TEST 以外のステートメントのマイグレーション時に使用する、オリジナルの VAGen EZESYS システム値 (例: MVSCICS または OS400) を提供します。VisualAge Generator 互換モードをオフにすると、*customerPrefixEZESYS* を初期化するステートメントの各プログラムごとに、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。プログラムを変更して、*customerPrefixEZESYS* 宣言と初期化ステートメントを除去することによって、この問題を訂正できます。プログラムの保存時には、*customerPrefixEZESYS* を使用する各ステートメントごとに、EGL 検査により

「問題」ビューに追加のエラー・メッセージが表示されます。このエラーを訂正するには、新しい EGL のシステム値を提供する EGL **sysVar.systemType** システム変数を使用するようにステートメントを変更できます。 *customerPrefixEZESYS* の現在の使用方法によっては、データベース、ファイル、または DataTable の値を、以前の VAGen システム値から新しい EGL システム値に変更することが必要になる場合もあります。 142 ページの『EZESYS』 および 400 ページの EZESYS の状態条件を参照してください。

マイグレーション設定「以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)」を選択すると、マイグレーション・ツールは自動的に各プログラムの変数宣言および初期化ステートメントを省略します。 EGL 検査は、*customerPrefixEZESYS* を使用するすべてのステートメントについて「問題」ビューにエラー・メッセージを表示します。

- EGL は、システム関数 **vgLib.connectionService()** の使用を許可します。これは、指定した引数とランタイム環境に応じてさまざまな SQL 接続サービスを提供する、VAGen EZECONCT システム関数の置換表現です。 VisualAge Generator 互換モードをオフにすると、**vgLib.connectionService()** を使用する各ステートメントごとに、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。このエラーは、新しい EGL 特殊システム関数 (例: **sysLib.connect()**、**sysLib.disconnect()**、**sysLib.disconnectAll()**、または **sysLib.queryDatabase()**) のいずれかを使用するように変更することによって訂正できます。使用する EGL システム関数は、VAGen EZECONCT システム関数に対して指定した引数、およびランタイム環境によって異なります。また、**vgVar.sqlIsolationLevel** は、EGL システム関数の選択、またはシステム関数に指定する必要がある引数に影響する可能性があるため、これが使用されているかどうか確認する必要があります。
- EGL は、システム変数 **vgVar.sqlIsolationLevel** の使用を許可します。これは VAGen EZESQISL の置換表現です。この変数は、システム共通プロダクトの旧リリースと VSE ランタイム環境用の VisualAge Generator で SQL 分離レベルを制御するために使用され、また VisualAge Generator 4.5 では ODBC データベースへのアクセスに使用されます。 VisualAge Generator 互換モードをオフにすると、**vgVar.sqlIsolationLevel** を使用する各ステートメントごとに、EGL 検査により「問題」ビューにエラー・メッセージが表示されます。 EZESQISL が使用されることはまれなので、多くの場合、「問題」ビューにエラーは示されません。エラーがある場合は、**vgVar.sqlIsolationLevel** がプログラム・ロジックの制御に使用されていないと、この変数を完全に除去できます。また、一部のプログラム・ロジックの制御に **vgVar.sqlIsolationLevel** が使用されている場合は、プログラム内で宣言した新規変数によって **vgVar.sqlIsolationLevel** を置き換えることができます。システム関数 **vgLib.connectionService()** の振る舞いは **vgVar.sqlIsolationLevel** の値によって決まる場合があるので、このシステム関数が使用されているかどうかを必ず確認してください。

VisualAge Generator 互換モード設定をオフにする場合は、それぞれのビルド記述子パーツから **vagCompatibility = "YES"** を必ず除去してください。マイグレーション設定「互換モードを設定しない (Do not set compatibility mode)」を選択すると、マイグレーション・ツールは自動的に各ビルド記述子パーツから **vagCompatibility = "YES"** を省略します。

第 6 部 言語と実行時に関する違い

VisualAge Generator と EGL の間には、さまざまな言語の違い、および実行時の違いがあります。

第 10 章 言語と実行時に関する違い

言語に関する違い

EGL が VisualAge Generator Developer を完全に置き換えない部分については、10 ページの『EGL にマイグレーションできるかどうかの判別』を参照してください。

EGL 言語に正確にマイグレーションできない VAGen 言語要素、またはマイグレーション方式については、79 ページの『第 3 章 未確定状態の処理』を参照してください。

それぞれの VAGen 言語要素を EGL にマイグレーションする方法の詳細については、303 ページの『付録 B. VisualAge Generator と EGL の言語要素の関係』を参照してください。

実行時の違い

ソース・コードを EGL にマイグレーションした後、コードを生成し、十分なテストを行って、コードが VisualAge Generator の場合と同じように実行されることを確認する必要があります。実行時の具体的な違いは、ターゲット環境およびプログラムの処理内容によって異なります。これらの違いについては、以下のセクションで説明されています。

- 一般的な違い
- SQL サポートに関する違い
- DL/I サポートに関する違い
- デバッグに関する違い
- 生成される COBOL に関する違い
- 生成される Java に関する違い
- ホスト環境とワークステーション環境の違い
- 分散 CICS 環境とネイティブ・ワークステーション環境の違い
- 生成される C++ と生成される Java の違い

一般的な違い

次に示す実行時の振る舞いの違いは、発生してもマイグレーション・ログや「問題」ビューにメッセージが示されない可能性があります。デバッグ時、または生成された Java コードまたは COBOL コードの実行時に、問題が起こることがあります。

- テキスト・プログラムと印刷書式には、以下の違いが存在します。
 - 書式フィールドの長さが不足していて、数字とフォーマット設定情報 (符号、小数点、通貨記号、および数字分離記号) がすべて収まらない場合は、ランタイム・エラーが発生します。

- デフォルト以外の充てん文字は、プログラムが SET *formItem* FULL ステートメントを発行しない場合でも常に受け入れられます。
- 書式にある配列は、配列の先頭要素の検証プロパティとフォーマット設定プロパティを常に使用します。このため、配列の要素によってこれらのプロパティが異なることを許容していた VisualAge Generator とは少し異なる振る舞いが生じる場合があります。詳しくは、102 ページの『マップ配列と属性』を参照してください。
- row=0, column=0 のフィールドがマップに含まれていた場合は、対応する書式を使用するプログラムをテストして、外観や振る舞いの違いを必ず確認してください。詳しくは、105 ページの『row=0, column=0 にあるフィールド』を参照してください。
- ワークステーション・プラットフォームの場合、デバッグを含み、以下のキー・マッピングが textForms に使用されます。

表 68. textForms のキー・マッピング

3270 キー	VAGen マッピング	Windows での EGL マッピング	Linux および AIX での EGL マッピング
PF1-PF12	F1-F12	F1-F12	F1-F12
PF13-PF24	Alt+F1-Alt+F12	Shift+F1-Shift+F12	Ctrl+S を押してから F1-F12 を押します
PA1-PA3	Ctrl+F1-Ctrl+F3	Ctrl+F1-Ctrl+F3	Ctrl+A を押してから F1-F3 を押します

注:

- + は同時に 2 つのキーを押す必要があることを示します。
- Linux および AIX では、Ctrl+S および Ctrl+A はトグルとして機能します。キーの組み合わせを間違えて押した場合、それらのキーを再度押すとオフになります。Ctrl+S を押してから F1-F12 以外のキーを押しても効果はありません。同様に、Ctrl+A を押してから F1-F3 以外のキーを押しても効果はありません。
- VAGen REDEFINED レコードであるレコードがプログラムのマイグレーション時に使用不可の場合、マイグレーション・ツールはデータ宣言に EGL **redefines** プロパティを組み込みません。この結果、VisualAge Generator の場合は単一の領域に 2 つの定義がありましたが、その代わりにレコード域が 2 つに分割されます。このため、未初期化のデータや無効データが原因で、異常終了などのエラーが発生する可能性があります。詳しくは、86 ページの『再定義レコード』を参照してください。
- EGL では、VisualAge Generator より多くの状況でハード入出力エラーが起こります。
 - VisualAge Generator の場合、非 SQL レコードの UNQ はソフト・エラーなので、HRD 入出力エラー状態は設定されません。EGL では、**unique** はハード・エラーであるため **hardIOError** テストの結果も真になります。詳しくは、138 ページの『入出力エラー値 UNQ および DUP』を参照してください。
 - iSeries の場合、VAGen 入出力エラー値 LOK は EGL **deadlock** 入出力エラー状態にマイグレーションされます。VisualAge Generator の場合、LOK はソフ

ト・エラーなので、HRD 入出力エラー状態は設定されません。EGL では、**deadlock** はハード・エラーであるため **hardIOError** テストの結果も真になります。詳しくは、141 ページの『入出力エラー値 LOK』を参照してください。

- 関数のマイグレーション時に入出力エラー・ルーチンが使用不可ならば、マイグレーション・ツールは入出力エラー・ルーチンが **main** 関数でないと想定し、関数呼び出しに変換します。VisualAge Generator では、入出力エラー・ルーチンが **main** 関数である場合に、エラーが実行時に発生すると、VisualAge Generator は関数の現行の実行スタックを消去し、入出力エラー・ルーチンとして指定された **main** 関数のみを含む新規スタックを開始します。これにより、実行スタック用のストレージも消去されます。EGL では、マイグレーション・ツールが関数呼び出しへの変換を行うので、エラーが実行時に発生すると、EGL は **main** 関数を現行の実行スタックに追加します。現行の実行スタックを消去して、**main** 関数のみの新規スタックを開始することはありません。このため、関数にローカル・ストレージやパラメーター・リストがある場合は、無限ループが発生したり、大量のリソースが使用されたりする可能性があります。詳しくは、115 ページの『入出力エラー・ルーチン』を参照してください。
- **DataTable** 内のフィールドを単一行の内容とともに使用して書式上のレコードまたは配列内の構造化フィールドを初期化すると、VisualAge Generator とは別の初期化結果になります。VisualAge Generator では、この場合、ソースはスカラーとして処理され、ターゲット配列はスカラー・ソースによって完全に初期化されます。EGL では、ターゲット配列の最初の要素のみが初期化されます。このため、ターゲット配列内の未初期化のデータや無効データが原因で、異常終了などのエラーが発生する可能性があります。詳しくは、130 ページの『単一行テーブルをソースとして含む MOVEA』を参照してください。

SQL サポートに関する違い

SQL サポートに関する違いは、以下の状況においてプログラムの動作に影響を与える可能性があります。

- COBOL を生成する場合、ホスト上の DB2 とデバッガーによって使用される JDBC との違いが、EGL でデバッグを行う際にプログラムの動作に影響を与える可能性があります。
- 以前に C++ を生成して今回 Java を生成する場合は、DB2 または ODBC と Java 生成によって使用される JDBC との違いが、EGL でデバッグを行う際およびネイティブ・ワークステーション環境でプログラムを実行する際の両方においてプログラムの動作に影響を与える可能性があります。

SQL 動作では以下の違いがあります。

- ODBC は EGL ではサポートされません。DB2 以外の SQL データベース・マネージャーを使用する場合は、ご使用のデータベース・マネージャー用の JDBC ドライバーを入手する必要があります。
- JDBC は 2 フェーズ・コミットをサポートしません。このため、次の違いがあります。
 - コミットとロールバックに関して、SQL マネージャーと MQ series マネージャーの呼び出しは別個に行われます。このため、問題が発生した場合に一方の

リソースでコミットまたはロールバックが行われ、他方のリソースで対応するコミットまたはロールバックが行われない可能性があります。

- EZECONCT (EGL **vgLib.connectionService()**)。VisualAge Generator の場合は、作業単位引数として R オプションを指定することにより、現行接続を終了せずに接続を別のデータベースに切り替えることができます。これにより、同じ作業単位内で複数のデータベースを更新することができます。EGL では、R オプションも、作業単位引数の D1C、D2A、D2C、および D2E の各オプションも、すべて、D1E を指定した場合と同様に処理されます。D1E は 1 フェーズ・コミットですが、データベース接続を自動的に解放しません。データベースを切断するには、DISC、DCURRENT、または DALL オプションを明示的に要求する必要があります。詳しくは、**vgLib.connectionService()** のオンライン・ヘルプを参照してください。
- JDBC では、常に動的 SQL が実行されます。(VAGen および EGL の両方において) 生成された COBOL および (VAGen において) 生成された C++ は、VAGen Execution time statement build オプション (EGL **prepare** ステートメント) またはテーブル名のホスト変数を使用する場合を除いて、静的 SQL を使用します。このため、次の違いがあります。
 - 動的モードでは、single row select は **sysVar.sqlData.sqlCode** を -811 に設定せずに戻される複数の行になります。
- JDBC は、DB2 とは異なる方法でコード・ページ変換を処理します。データベースがホストに置かれており、「FOR BIT DATA」として定義された CHAR、DBCHAR、または MBCHAR SQL 列が組み込まれている場合、DB2 はどのようなコード・ページ変換も行いません。しかし、JDBC はデータを変換します。このような場合は、EGL SQL レコード定義を変更して、「FOR BIT DATA」として定義されている SQL 列に対応するフィールド用に **asBytes = yes** プロパティを追加します。
- VisualAge Generator の場合は、DATE、TIME、または TIMESTAMP データを含む SQL 列用に CHAR フィールドを使用します。ホスト上で DB2 によって使用される SQL 日付/時刻形式は JDBC によって戻される形式とは異なることがあります。ホスト環境を対象としたプログラムをデバッグするか、またはホスト DB2 データベースを使用する Java プログラムを生成している場合、インターフェースは JDBC を介して提供されます。このため、以下の手法のいずれかを使用して必要な日付/時刻形式を取得する必要があります。
 - **手法 1:** この手法では、DB2 バージョン 9、フィックスパック 4 またはそれ以降を使用します。接続 URL を設定して、以下の DB2 プロパティを組み込みます。
 - **dateFormat**
 - **timeFormat**
 - **timestampFormat**
 - **手法 2:** この手法では、DB2 バージョン 8 を使用します。デフォルトの汎用ドライバーの代わりに DB2 APP ドライバーを使用します。APP ドライバーを使用して日付、時刻、およびタイム・スタンプの形式を制御するには、以下の手順を実行します。

1. APP ドライバーを使用するには、以下の情報を設定します。

表 69.

フィールド	値
ドライバー名	com.ibm.db2.jdbc.app.DB2Driver
ドライバーの場所	db2java.zip
サンプル接続 URL	jdbc:db2:MYHOSTDB

2. 日付形式を設定するには、「DB2 コマンド・プロンプト」ウィンドウで以下のコマンドを入力します。

```
db2 update cli cfg for section COMMON using DateTimeStringFormat xxx
```

ここで、*xxx* には、使用する DB2 日付形式に基づいて EUR、ISO、 USA などを指定します。

3. 以下のコマンドを入力して設定を確認します。

```
db2 get cli cfg
```

4. EGL デベロッパー製品を開始します。

- **手法 3:** この手法では EGL のみを使用します。SQL DATE、TIME、または TIMESTAMP 列を含む CHAR フィールド用に **sqlDataCode** プロパティを設定します。**sqlDataCode** プロパティに、以下の値を指定します。

表 70.

データ型	sqlDataCode
DATE	385
TIME	389
TIMESTAMP	393

さらに、以下のシステム変数も設定して DB2 内の日付、時刻、およびタイム・スタンプ形式に対応させる必要があります。

- **strLib.defaultDateFormat**
- **strLib.defaultTimeFormat**
- **strLib.defaultTimestampFormat**

デフォルトの日付/時刻形式は、プログラム内で直接に、または以下のビルド記述子オプションを設定して初期化することができます。

- **defaultDateFormat**
- **defaultTimeFormat**
- **defaultTimeStampFormat**

詳しくは、「EGL 言語リファレンス」および「EGL 生成ガイド」を参照してください。

- JDBC を使用する場合は SQL に関連する EZE ワード内の他の変更点は、以下のとおりです。
 - EZESQISL (EGL **vgVar.sqlIsolationLevel**)。VisualAge Generator では、値 1 はカーソル固定の要求を表します。EGL では、値 1 を指定すると順序付け可能トランザクションが使用されます。
 - EZESQRRM (EGL **sysVar.sqlData.sqlerrmc**) はサポートされません。

- EZESQWN6 (EGL `sysVar.sqlData.sqlwarn[7]`) はサポートされません。
- EZESQLCA (EGL `sysVar.sqlData.sqlca`) フィールドには制限があります。これらのフィールドには、EZESQRRM と EZESQWN6 の値を含めることはできません。

DL/I サポートに関する違い

DL/I サポートには、以下の違いが生じます。

- VisualAge Generator は、IMS または DL/I サポートを生成する際に常に CBLTDLI を使用します。CBLTDLI の場合は、IMS または DL/I PSB に PCB 用の PCB 名を組み込む必要はありません。EGL 生成のプログラムでは、プログラム用に指定された **callInterface** プロパティに応じて CBLTDLI または AIBTDLI のいずれかが使用されます。マイグレーション・ツールは、CBLTDLI を使用するようにプログラムを変換します。これは、VisualAge Generator が CBLTDLI を使用するためです。ただし、**callInterface** プロパティの設定にかかわらず、EGL デバッグ機能では常に AIBTDLI が使用されます。AIBTDLI では、IMS または DL/I PSB 内の PCB を表す PCB 名が必要になります。このため、EGL デバッグ機能を使用するには、IMS または DL/I PSB 内のすべてのデータベース PCB を表すに PCB 名を追加する必要があります。各 PCB マクロ上のラベルを指定するか、または各 PCB マクロ上の PCBNAME パラメーターを指定するかのいずれかの方法によって、PCB 名を追加することができます。EGL デバッグ機能は DL/I データベースのデバッグのみをサポートするため、データベース PCB 用の PCB 名を追加するのみで済みます。プログラムを変更して AIBTDLI を使用する場合は、IMS または DL/I PCB 内のすべての PCB 用に PCB 名を組み込んで、EGL PSB 内の対応する PCB 名を指定する必要があります。

マイグレーション・ツールは、**pcbName** プロパティを VAGen PSB のデータベース名に設定することにより、デバッガーによって使用される EGL PSB の準備を試行します。IMS または DL/I PSB に PCB 名を追加する場合は、VAGen PSB から得られたデータベース名を開始点として使用することにより、EGL PSB への変更を最小限にするができます。ただし、VAGen PSB では PSB 内の複数の PCB 用に同じデータベース名があることは許容されます。これは、生成された COBOL 内でこれらの名前が使用されないためです。IMS および DL/I PSB では、PCB 名を固有にする必要があります。このため、マイグレーション・ツールによって提供された PCB 名が受け入れ不可である場合があります。PSB 内に同じデータベース名を持つ複数の PCB が存在する場合、マイグレーション・ツールはエラー・メッセージを出します。

以下の例は、VAGen PSB、マイグレーション・ツールによって生成された対応する EGL PSB、および PCB 名を追加する前と後の両方における IMS PSB を示します。VAGen PSB は、同じデータベース名 DDDDDDDDD に番号 3 および 4 が付けられた PCB を持ちます。マイグレーション済み EGL PSB は、**pcbName** プロパティの値としてデータベース名を使用します。IMS PSB は両方の PCB に対して DDDDDDDDD を使用することができないため、別の名前 DDDDDDDDX を使用し、EGL PSB 内の対応する PCB を更新して正しい名前を指定する必要があります。

PSB の例

注:

- PCB 名は太字です
- データベース PCB 用の PCB 名のみを追加して、EGL デバッガーを使用する必要があります

VAGen PSB

PCBs

1	TP	(代替入出力 PCB)
2	TP	(明示された代替入出力 PCB)
3	DB	DatabaseName = DDDDDDDD , <i>otherPCBInformation</i>
4	DB	DatabaseName = DDDDDDDD , <i>otherPCBInformation</i>
5	DB	DatabaseName = OTHERDB , <i>otherPCBInformation</i>
6	DB	DatabaseName = ELAWORK
7	GSAM	DatabaseName = GGGGGGGG

マイグレーション・ツールによって作成された EGL PSBRecord

```
iopcb IO_PCBRecord { @PCB { pcbType = PCBKind.TP } };
pcb0 IO_PCBRecord { redefines = iopcb};
ELAALT ALT_PCBRecord { @PCB {pcbType = PCBKind.TP } };
pcb1 ALT_PCBRecord { redefines = ELAALT };
ELAEXP ALT_PCBRecord { @PCB {pcbType = PCBKind.TP } };
pcb2 ALT_PCBRecord { redefines = ELAEXP };
DDDDDDDD dbPCB DB_PCBRecord
{ @PCB { pcbType = PCBKind.DB, pcbName = "DDDDDDDD",
      otherPCBInformation
} }
pcb3 DB_PCBRecord { redefines = DDDDDDDD_dbPCB };
DDDDDDDD_4 dbPCB DB_PCBRecord
{ @PCB { pcbType = PCBKind.DB, pcbName = "DDDDDDDD",
      // pcbName in the above line must be changed to DDDDDDDX
      // to match the IMS PSB
      otherPCBInformation
} }
pcb4 DB_PCBRecord { redefines = DDDDDDDD_4_dbPCB };
OTHERDB_dbPCB DB_PCBRecord
{ @PCB { pcbType = PCBKind.DB, pcbName = "OTHERDB",
      otherPCBInformation
} }
pcb5 DB_PCBRecord { redefines = OTHERDB_dbPCB };
ELAWORK DB_PCBRecord
{ @PCB { pcbType = PCBKind.DB } }
pcb6 DB_PCBRecord { redefines = ELAWORK };
GGGGGGGG_gsamPCB GSAM_PCBRecord
{ @PCB { pcbType = PCBKind.GSAM, pcbName = "GGGGGGGG" } } ;
pcb7 GSAM_PCBRecord { redefines = GGGGGGGG_gsamPCB };
```

元の IMS PSB

```
TITLE 'PSB FOR PROCESSING SAMPLE DATABASES'
PRINT NOGEN
PCB TYPE=TP,MODIFY=YES
PCB TYPE=TP,MODIFY=YES,EXPRESS=YES
PCB TYPE=DB,DBDNAME=XXXXXXX,otherPCBInformation
PCB TYPE=DB,DBDNAME=XXXXXXX,otherPCBInformation
PCB TYPE=DB,DBDNAME=XXXXXXX,otherPCBInformation
```

```

PCB      WORKDB=ELAWORK
PCB      TYPE=GSAM,DBDNAME=YYYYYYYY,otherPCBInformation
PSBGEN   LANG=ASSEM,CMPAT=YES,otherPSBInformation
END

```

変更された IMS PSB

```

                                TITLE 'PSB FOR PROCESSING SAMPLE DATABASES'
                                PRINT NOGEN
ELAALT  PCB      TYPE=TP,MODIFY=YES
ELAEXP  PCB      TYPE=TP,MODIFY=YES,EXPRESS=YES
DDDDDDDD PCB      TYPE=DB,DBDNAME=XXXXXXXX,otherPCBInformation
DDDDDDDX PCB      TYPE=DB,DBDNAME=XXXXXXXX,otherPCBInformation
OTHERDB PCB      TYPE=DB,DBDNAME=XXXXXXXX,otherPCBInformation
ELAWORK PCB      WORKDB=ELAWORK
GGGGGGGG PCB      TYPE=GSAM,DBDNAME=YYYYYYYY,otherPCBInformation
                                PSBGEN LANG=ASSEM,CMPAT=YES,otherPSBInformation
                                END

```

- 一般に、IMS および DL/I プログラムのテストの際は、VAGen および EGL プログラムの両方に同じ制限が適用されます。
 - 以下の処理が EGL デバッグによってサポートされます。
 - IMS/VS 環境内で実行中のプログラムへの呼び出し
 - EGL 入出力ステートメントを通しての DL/I データベース入出力の使用
 - 以下の処理は EGL デバッグによってサポートされません。
 - データベースが z/OS CICS または VSE 上にある場合の、DL/I データベース I/O の使用。
 - IMS メッセージ・キューまたは GSAM ファイルに関連付けられたシリアル・ファイル。デバッグでは、代わりに **seqws** を使用します。
 - AUDIT (EGL **sysLib.audit()**) または CREATX (EGL **VGLib.startTransaction()**) などのシステム関数
 - 以下のどちらかのアクションを実行する IMS または DL/I 呼び出しを行うための **vgLib.VGTDLI()**、**dliLib.EGLTDLI()**、または **dliLib.AIBTDLI()** システム関数。
 - 入出力 PCB、TP PCB、GSAM PCB、または FastPath データベースへのアクセス
 - CHKP、GSCD、PCB、TERM、または XRST 呼び出し

デバッグに関する違い

デバッグに関する違いが、テストに影響を及ぼす可能性があります。COBOL 環境で生成を行う場合、デバッグが提供する次の領域のサポートは、生成される COBOL のサポートと異なるので、特にこれらの違いに注意する必要があります。

- テキスト書式には、以下の違いが生じます。
 - テキスト書式では明滅はサポートされません。
 - **isDecimalDigit** プロパティは、文字フィールドに対してのみサポートされます。ハードウェア属性としてではなく、ソフトウェア編集としてインプリメントされます。数値フィールドもソフトウェア編集を使用します。詳しくは、101 ページの『マップ・フィールドと数値ハードウェア属性』を参照してください。

- フローティング・テキスト書式の **screenSizes** プロパティに対して指定された値の 1 つが、FormGroup の **@ScreenFloatingArea** プロパティに対して指定された **screenSize** エントリーの 1 つに一致する必要があります。同様に、フローティング印刷書式の **formSize** プロパティに対して指定された値の 1 つが、FormGroup の **@PrintFloatingArea** プロパティに対して指定された **pageSize** エントリーの 1 つに一致する必要があります。
- 代替索引レコードが定義されている索引レコードの場合、DUP 入出力エラー値の設定は VisualAge Generator とは異なります。VisualAge Generator の場合、**SET record SCAN** に続いて **SCAN** または **SCANBACK** 入出力オプションを指定しても、**SET record SCAN** ステートメントに対して DUP 入出力エラー値が設定されません。DUP 入出力エラー値は、最後に検索された重複キー・レコードを除く、それぞれの重複キー・レコードに対して設定されます。Java 生成の場合は、**set record position** に続いて **get next** または **get previous** ステートメントを指定すると、検索された最初の重複キー・レコード上ではなく **set record position** 上に **duplicate** 状態が設定されます。その他の重複キー・レコードに対しては、VisualAge Generator の場合と同様に **duplicate** 状態が設定されます。EGL の **duplicate** 状態は、最初と最後の重複キー・レコードを除くすべてのレコードに対して設定されます。索引レコードと代替索引レコードについて、また、これらのレコードを **set record position**、**get next**、および **get previous** とともに使用方法の詳細については、オンライン・ヘルプを参照してください。
- (VisualAge Generator における) 対話式テスト機能では、DB2 または ODBC が使用されます。(VisualAge Generator および EGL の両方において) 生成された COBOL では、DB2 が使用されます。(VisualAge Generator において) 生成された C++ では、DB2 または ODBC が使用されます。EGL デバッグおよび Java 生成では、JDBC が使用されます。COBOL を生成する場合、または以前に C++ を生成した場合は、これにより EGL デバッガーを使用してプログラムをデバッグする場合に違いが生じます。詳しくは、277 ページの『SQL サポートに関する違い』を参照してください。
- (VisualAge Generator における) 対話式テスト機能では、CBLTDLI が使用されます。EGL デバッグでは AIBTDLI が使用されます。詳しくは、280 ページの『DL/I サポートに関する違い』を参照してください。
- 無効なデータを含む NUM または NUMC データの処理は、VisualAge Generator の場合とは異なります。VisualAge Generator では、「無効数値データで停止 (**Stop on invalid numeric data**)」設定をクリアすると、NUM および NUMC フィールドにあるブランクなどの無効なデータが、ITF によって許容されます。NUM または NUMC フィールドにブランクが含まれている場合、そのフィールドは値が 0 の場合と同様に処理されます。例えば、そのフィールドが 0 と比較された場合、比較テストの結果は真になります。NUM または NUMC フィールドにブランク以外の無効データが含まれている場合、比較テストの結果は偽になります。つまり、これは、COBOL 生成オプション **/SPZERO** の使用をシミュレートするもので、ブランクが NUM および NUMC フィールドでは許容され、それらのフィールドに 0 が含まれる場合と同様に処理されることを意味します。NUM または NUMC フィールドで他の無効データを使用すると、異常終了などのランタイム・エラーになります。**/SPZERO** 生成オプションを使用しない場合、NUM または NUMC フィールドにおけるブランクなどの無効データがあると、ランタイム・エラーが発生します。**/SPZERO** 生成オプションは、BIN、PACK、および PACF フィールドには影響を与えません。EGL では、**spacesZero** ビルド

記述子オプションにより、生成済みの COBOL プログラムに対して /SPZERO と同じサポートが提供されます。EGL デバッガーには、VAGen の「無効数値データで停止 (Stop on invalid numeric data)」に対応する設定はありません。代わりに、EGL デバッガーおよび Java 生成の両方には、**spacesZero** ビルド記述子オプションが用意されています。このため、COBOL 生成において **spacesZero** を使用する場合は、このオプションをデバッグ・ビルド記述子パーツに追加する必要があります。

生成される COBOL に関する違い

生成される COBOL コードには、次の違いが生じます。

- z/OS の場合、EGL で生成された COBOL テキストと基本プログラムは、VisualAge Generator プログラムと完全に互換性があります。次のどちらの場合も、VAGen プログラムの再生成または再コンパイルは必要ありません。
 - VAGen プログラムが、EGL への転送手段として **CALL**、**DXFR**、または **XFER** を使用しています。
 - EGL プログラムは、VAGen プログラムへの転送手段として **call**、**transfer**、または **show** を使用します。

EGL プログラムと VAGen プログラムの間で行われる呼び出しまたは転送に関する制限は、2 つの VAGen プログラム間で呼び出しまたは転送を行う場合の制限と同様です。例えば、VAGen の呼び出し先プログラムは、他のプログラムへの転送に **DXFR** ステートメントまたは **XFER** ステートメントを使用できません。同様に、EGL 呼び出し先プログラムでは、**transfer to program**、**transfer to transaction**、および **show** を使用して他のプログラムに転送することはできません。

- z/OS CICS の場合は、特別な状況により、実行単位内ですべてのプログラムを再リンクして IBM Rational COBOL Runtime for zSeries を使用しなければならないことがあります。詳しくは、565 ページの『z/OS CICS における VisualAge Generator および EGL のインターオペラビリティ』を参照してください。
- iSeries の場合、565 ページの『iSeries における VisualAge Generator および EGL のインターオペラビリティ』で説明されている規則に従ってプログラムを生成する必要があります。
- 書式には以下の違いが生じます。
 - **isDecimalDigit** プロパティは、文字フィールドに対してのみサポートされます。ハードウェア属性としてではなく、ソフトウェア編集としてインプリメントされます。数値フィールドもソフトウェア編集を使用します。詳しくは、101 ページの『マップ・フィールドと数値ハードウェア属性』を参照してください。

生成される Java に関する違い

生成される Java コードには、次の違いが生じます。

- VAGen で生成された Java プログラムでは、**vgj.properties** ファイルを使用してランタイム環境が制御されます。EGL で生成された Java プログラムでは、**genProperties** ビルド記述子オプションの値に応じて、**programName.properties** ま

たは **rununit.properties** のいずれかが使用されます。VisualAge Generator と EGL ランタイム・プロパティとの間の対応については、468 ページの『vgj.properties』を参照してください。

- EGL で生成された Java プログラムは、VisualAge Generator プログラムと完全に互換性があるわけではありません。

以下のプログラム対話がサポートされます。

- EGL プログラムは、リモート呼び出しを使用して、VAGen で生成された Java または C++ プログラムを呼び出すことができます。同様に、VAGen で生成された Java または C++ プログラムは、リモート呼び出しを使用して EGL プログラムを呼び出すことができます。
- EGL VGWebTransaction プログラムは、**show VGUIRecord** ステートメントを使用して Web Transaction プログラムに間接的に転送することができます。同様に、VAGen Web トランザクション・プログラムは、UI record ステートメントを含む XFER を使用して、EGL VGWebTransaction プログラムに間接的に転送することができます。

以下のプログラム対話はサポートされません。

- EGL プログラムは、ローカル呼び出しを使用して VAGen 生成済み Java および C++ プログラムを呼び出すことはできません。同様に、VAGen 生成済み Java または C++ プログラムは、ローカル呼び出しを使用して EGL プログラムを呼び出すことはできません。
- EGL プログラムは、**transfer** または **show TextForm** ステートメントを使用して VAGen プログラムに転送することはできません。同様に、VAGen プログラムは、DXFR または XFER ステートメントを使用して EGL プログラムに転送することはできません。

ホスト環境とワークステーション環境の違い

ホスト環境 (CICS など) での生成からワークステーション環境 (ネイティブ AIX など) での生成に変更する場合は、以下の違いを考慮する必要があります。

- ホスト環境の照合シーケンスは EBCDIC です。Java 生成の照合シーケンスは UNICODE です。このため、コードの以下のエリアを検討する必要があります。
 - 適合テーブルの範囲
 - キーの high-value または low-value としてコード化された特定の値
 - 2 つのファイルが一致するキーの比較
- ユーザーは以下の情報を知っておく必要があります。
 - ファンクション・キーのマッピング。通常、ワークステーション・キーボードには、PF13 - PF24 および PA1 - PA3 のキーがないためです。キーボード・マッピングについては、276 ページの表 68を参照してください。
 - DBCS モードから、または DBCS モードへの切り替えを行うときに、シフトイン文字およびシフトアウト文字を入力する必要がないこと。
- 1 つの呼び出しで AIX ネイティブから CICS に渡されるすべてのレコードの全長は 32567 を超えることはできません。これは、CICS の制限事項です。以前に CICS の生成を行い、デフォルトの parmform=COMMPTR を使用した場合は、す

すべてのレコードの全長がこの制限を超える可能性があります。以前に `parmform=COMMDATA` を使用した場合は、すべてのレコードの全長はこの制限内になります。

- 以下のセクションの情報も検討してください。
 - 『分散 CICS 環境とネイティブ・ワークステーション環境の違い』
 - 289 ページの『生成される C++ と生成される Java の違い』

分散 CICS 環境とネイティブ・ワークステーション環境の違い

生成された EGL コードをワークステーション環境で実行するには、Transaction Series (TX Series または CICS) の環境で実行する代わりに、ネイティブ・プロセスとして実行するように変更する必要があります。次に、CICS 環境からネイティブ環境への移行に伴う相違点または変更点を示します。このリストは VAGen 用語を使用していますが、対応する EGL 言語要素内では変更を加える必要があります。対応する EGL 言語要素を判別するには、303 ページの『付録 B. VisualAge Generator と EGL の言語要素の関係』を参照してください。

- 以下の一般的な違いが存在します。
 - 複数のユーザーが 1 つのサーバー上の同じアドレス・スペースで実行することはできません。ユーザーはそれぞれのワークステーションで実行します。
 - クライアントの作業単位はサポートされません。
 - C++ の生成から Java の生成に変更されました。必ず 289 ページの『生成される C++ と生成される Java の違い』に関するセクションを参照してください。
 - CICS からネイティブ環境にマイグレーションする際には、パフォーマンスとスケーラビリティを必ずテストしてください。
 - CICS とネイティブ環境の間では、通信プロトコルが異なります。使用するプロトコルを決定してから、それに応じて EGL リンケージ・オプション・パーツとリソース関連パーツを変更する必要があります。
 - 分散 CICS 用の VAGen 生成済みプログラムでは、環境変数を使用してランタイム環境が制御されます。EGL で生成された Java プログラムでは、**genProperties** ビルド記述子オプションの値に応じて `programName.properties` または `rununit.properties` のいずれかが使用されます。VisualAge Generator 環境変数と EGL ランタイム・プロパティとの間の対応については、466 ページの『ランタイム環境変数』を参照してください。
- 次の CICS 固有の特殊機能語とサービス・ルーチンは、ネイティブ環境ではサポートされません。
 - CICS ジャーナル・エントリー書き込み用の AUDIT (EGL `sysLib.audit()`)。AUDIT という名前の非 EGL プログラムをユーザーが独自に作成して、ネイティブ環境用の同様な情報をファイルに書き込むことができます。
 - 一時ストレージ・キュー削除用の EZEPURGE (EGL `sysLib.purge()`)。
。 `sysLib.purge()` への参照を除去する必要があります。代わりに、`sysVar.systemType` を確認して z/OS 環境の CICS で実行する場合にのみ

`sysLib.purge()` を使用することもできます。この手法を使用する場合は、必ず `EGL eliminateSystemDependentCode` ビルド記述子オプションを「はい」に設定してください。

- リモート・ファイルおよびリモート・プログラムのロケーション、または `CREATX` (`EGL sysLib.startTransaction()`) を使用してリモート・トランザクションを開始するロケーションを設定するための `EZELOC` (`EGL sysVar.remoteSystemID`)。
- CICS 固有のリソース関連は、ネイティブ環境ではサポートされません。EGL ネイティブ環境でサポートされるオプションを使用するように、リソース関連パーツを変更する必要があります。ネイティブ環境で生成を行う場合にサポートされない、CICS 固有のリソース関連は次のとおりです。
 - CICS スプール・ファイル。
 - 一時データ・キュー (トリガー・レベル 1 の一時データ・キューを含む)。
 - 一時ストレージ・キュー。
 - ローカル VSAM ファイル (AIX 環境の場合を除く)。
- 次に示す CICS 提供の機能は、ネイティブ環境ではサポートされません。
 - セキュリティー・サービス。
 - データベース接続と保存。
 - CICS ファイル管理 (リカバリー可能ファイルの使用など)。
 - 実際のセグメンテーションのサポート。
 - プログラム管理。
- プログラム間を転送する場合、以下の違いが生じます。
 - Web トランザクション以外のメインプログラムの場合、CICS の `XFER` ステートメントは次のトランザクション ID に転送されます。ネイティブ環境の場合、`XFER` ステートメントはプログラム名に転送されます。このため、すべての `EGL transfer to transaction` ステートメントと `show` ステートメントを変更して、プログラム名を指定する必要があります。
 - CICS 環境では、非 VAGen プログラムへの `XFER` または `DXFR` がサポートされます。ネイティブ環境の場合、`transfer` および `show` は非 EGL プログラムについてはサポートされません。
- 以下のコミットおよびロールバックの違いが生じます。
 - CICS は 2 フェーズ・コミットをサポートします。ネイティブ環境は、単一フェーズ・コミットのみをサポートします。
 - CICS に対しては、ファイルをリカバリー可能ファイルとして定義できます。ネイティブ環境の場合は、このように定義することはできません。
 - メッセージ・キューは、CICS 環境では他のリソースと同時にコミットまたはロールバックされます。ネイティブ環境では、単一フェーズ・コミットのみがサポートされるので、コミットとロールバック中に問題が発生した場合は、メッセージ・キューが SQL リソースと同時にコミットまたはロールバックされない可能性があります。
- 以下の `CALL CREATX` (`EGL sysLib.startTransaction()`) に関する違いが生じます。
 - `CALL CREATX` は、CICS 環境で別のトランザクションを開始し、パラメーター `prid` と `recip` を受け入れます。EGL `sysLib.startTransaction` は、ネイティ

ブ環境で別のプログラムを開始し、パラメーター *prid* と *recip* を無視します。ネイティブ環境用の生成を行う場合は、少なくとも

sysLib.startTransaction を変更してプログラム名を指定する必要があります。

- CICS は、ローカルとリモートの両方の CALL CREATX をサポートします。EGL **sysLib.startTransaction()** は、ローカル・プログラムの開始のみをサポートします。
- EZECONCT (EGL **vgLib.connectionService()**) を使用した SQL 接続サービス。VisualAge Generator の CICS 環境の場合、EZECONCT はパスワードを無視します。EGL のネイティブ環境の場合、**vgLib.connectionService()** ではパスワードが使用されます。Java 生成の変更に伴うその他の違いについては、289 ページの『生成される C++ と生成される Java の違い』を参照してください。
- EZE 特殊データ・ワードには、以下の違いが生じます。
 - EZEAPP (EGL **sysVar.transferName**)。VisualAge Generator の CICS 環境の場合に、XFER ステートメントに EZEAPP を使用するときは、EZEAPP の内容は開始される新規トランザクションの名前です。EGL のネイティブ環境において、**sysVar.transferName** が **transfer to transaction** または **show** ステートメントとともに使用される場合、**sysVar.transferName** には開始される新規プログラムの名前が含まれます。
 - EZEDEST (EGL **recordName.resourceAssociation**)。VisualAge Generator の CICS 環境の場合、EZEDEST の内容はプログラムの実行中にレコードに関連付けられるシステム・リソース名です。EGL のネイティブ環境の場合も、**recordName.resourceAssociation** にはレコードに関連付けられたシステム・リソース名が含まれます。ただし、情報のフォーマットはランタイム環境とファイル・タイプによって異なります。ここでは、ランタイム環境とファイル・タイプの両方を変更するため、**recordName.resourceAssociation** が使用されている箇所をすべて検討して、プログラムから提供される情報がご使用のネイティブ環境およびファイル・タイプに適していることを確認してください。
 - EZEDESTP (EGL **converseVar.printerAssociation**)。VisualAge Generator の CICS 環境の場合、EZEDESTP の内容は印刷ファイルに関連付けられる宛先です。EGL のネイティブ環境の場合、**converseVar.printerAssociation** には印刷ファイルに関連付けられたファイル名も含まれます。ただし、情報のフォーマットはランタイム環境とファイル・タイプによって異なります。ここでは、ランタイム環境とファイル・タイプの両方を変更するため、**converseVar.printerAssociation** が使用されている箇所をすべて検討して、プログラムから提供される情報がご使用のネイティブ環境およびファイル・タイプに適していることを確認してください。
 - EZEELTERM (EGL **sysVar.terminalID**)。VisualAge Generator の CICS 環境の場合、EZEELTERM の内容は CICS 端末 ID で、これは EZEUSR と同等です。EGL のネイティブ環境の場合、**sysVar.terminalID** は **user.name** Java 仮想マシンのシステム・プロパティから初期化されます。このプロパティが検索できない場合、**sysVar.terminalID** はブランクになります。
 - EZERCODE (EGL **sysVar.returnValue**)。VisualAge Generator の CICS 環境の場合、EZERCODE の値はシステムまたは呼び出し側プログラムに戻されません。EGL のネイティブ環境の場合、EZERCODE の値は無視されます。
 - EZERT8 (EGL **sysVar.errorCode**)。VisualAge Generator の CICS 環境の場合、EZERT8 は次のどちらかの形式になります。

- *RSnnnnnn*。ここで *nnnnnn* は、ファイル・アクセスと発生した問題に基づく VAGen 戻りコードです。
- *nnnnnnnn*。ここで、最初の 2 文字は、CICS EXEC インターフェース・ブロックにある EIBFN の先頭バイトの 16 進表記です。残りの 6 文字は、CICS EXEC インターフェース・ブロックにある EIBRCODE のバイト 0 から 2 の 16 進表記です。

EGL のネイティブ環境の場合、戻りコード情報はファイル・タイプによって異なります。**sysVar.errorCode** を使用している箇所を検討して、確認する値がご使用の環境およびファイル・タイプに適していることを確認する必要があります。

- **EZESEGTR (EGL sysVar.transactionID)**。VisualAge Generator の CICS 環境の場合、EZESEGTR は現行トランザクション ID に初期設定され、また CONVERSE の後に有効になる新規トランザクション ID の設定にも使用されます。EZESEGTR は、プログラム・ロジックの制御に使用できます。EGL のネイティブ環境の場合、EZESEGTR は無視され、プログラム・ロジックの制御には使用できません。
- **EZEUSR (EGL sysVar.sessionID)**。VisualAge Generator の CICS 環境の場合、EZEUSR の内容は CICS 端末 ID で、これは EZELTERM と同等です。EGL のネイティブ環境の場合、**sysVar.sessionID** は **user.name** Java 仮想マシンのシステム・プロパティから初期化されます。このプロパティが検索できない場合、**sysVar.sessionID** はブランクになります。
- **EZEUSRID (EGL sysVar.userID)**。VisualAge Generator の CICS 環境の場合、ユーザーがシステムにサインオンしていれば、EZEUSRID の内容は CICS ユーザー ID です。そうでなければ、内容はブランクです。EGL のネイティブ環境の場合、**sysVar.userID** は **user.name** Java 仮想マシンのシステム・プロパティから初期化されます。このプロパティが検索できない場合、**sysVar.userID** はブランクになります。

生成される C++ と生成される Java の違い

C++ の生成から、Java の生成に変更する場合は、次の違いが生じます。

- VAGen で生成された C++ プログラムでは、環境変数を使用してランタイム環境が制御されます。EGL で生成された Java プログラムでは、**genProperties** ビルド記述子オプションの値に応じて **programName.properties** または **rununit.properties** のいずれかが使用されます。VisualAge Generator 環境変数と EGL ランタイム・プロパティとの間の対応については、466 ページの『ランタイム環境変数』を参照してください。
- 生成される Java は、VAGen によって生成される C++ プログラムと相互運用できません。Java 用に生成された EGL プログラムは、C++ 用に生成された VAGen プログラムとの間で相互に転送を行うことができません。Java 用に生成された EGL プログラムは、C++ 用に生成された VAGen の呼び出し先バッチ・プログラムを呼び出すことができます。
- 生成された Java プログラムからネイティブ C++ または VAGen で生成された C++ への呼び出しは、プログラムが同じワークステーション上で実行されている場合でも、常にリモート呼び出しになります。したがって、この状況ではリンク・オプション・エレメントを追加する必要があります。

- VAGen によって生成される C++ プログラム内では、バイナリー・データは Intel フォーマット (逆方向のバイト・オーダー) で格納されます。EGL によって生成される Java プログラム内では、バイナリー・データは Intel フォーマットでは格納されません。生成される Java プログラムと、生成される C++ プログラムの間で呼び出し時に受け渡されるバイナリー・データは、EGL 変換テーブルによって変換されます。ただし、生成される Java プログラム内でファイルを使用する場合は、その前に、C++ プログラムによって書き込まれたファイル内のバイナリー・データを変換するプログラムを作成する必要があります。
- 生成された Java プログラムの場合、**transfer** または **show** ステートメントに対して以下のネーム解決規則が適用されます。
 - **transfer** または **show** ステートメントがパッケージ名とプログラム名の両方を明示的に指定している場合は、このプログラムが使用されます (例: `transfer to program mypackage.program1`)。
 - **transfer** または **show** ステートメントがプログラム名のみを明示的に指定している場合は、以下のうち最初に適用される基準がプログラム名の解決に使用されます。
 - **transfer** または **show** ステートメントを含むファイルが、パッケージとプログラムを明示的にインポートする (例: `import mypackage.program1` および `transfer to program program1`)。
 - ビルド記述子オプション **programPackageName** により、転送先プログラムを含むパッケージが示されている。
 - **transfer** または **show** ステートメントを含むファイルが、転送先プログラムを含むパッケージをインポートする (例: `import mypackage.*` および `transfer to program program1`)。
 - 生成時に使用されるリンケージ・オプション・パーツに、転送先プログラムとそのプログラムを含むパッケージを指定する転送元プログラムの **transferToProgram** または **transferToTransaction** エントリーが含まれている。**show** ステートメントの場合、転送リンク・エントリーは **transferToTransaction** エントリーの形式になっている必要があります。
 - 転送先プログラムが、**transfer** または **show** ステートメントと同じパッケージ内にある。
 - **transfer** または **show** ステートメントが **sysVar.transferName** を使用し、転送先プログラムが別のパッケージにある場合は、リンケージ・オプション・パーツまたは **programPackageName** ビルド記述子オプションを使用して、転送先プログラムを含むパッケージを指定する必要があります。
- 以下の一般的な違いが存在します。
 - VisualAge Generator によって生成された C++ コードを使用する場合、リソース関連は実行時に処理されます。EGL の場合は、生成時にリソース関連情報を指定して、この情報をプロパティ・ファイルに生成するオプションがあります。**resourceAssociations** ビルド記述子オプションを設定して、リソース関連パーツを指示します。
 - **genProperties** ビルド記述子オプションを、GLOBAL または PROGRAM に設定します。これらのビルド記述子オプションについて詳しくは、オンライン・ヘルプを参照してください。
- テキスト書式に以下の違いが生じます。

- テキスト書式では明滅はサポートされません。
- **isDecimalDigit** プロパティは、文字フィールドに対してのみサポートされます。ハードウェア属性としてではなく、ソフトウェア編集としてインプリメントされます。数値フィールドもソフトウェア編集を使用します。詳しくは、101 ページの『マップ・フィールドと数値ハードウェア属性』を参照してください。
- フローティング・テキスト書式の **screenSizes** プロパティに対して指定された値の 1 つが、FormGroup の **@ScreenFloatingArea** プロパティに対して指定された **screenSize** エントリーの 1 つに一致する必要があります。同様に、フローティング印刷書式の **formSize** プロパティに対して指定された値の 1 つが、FormGroup の **@PrintFloatingArea** プロパティに対して指定された **pageSize** エントリーの 1 つに一致する必要があります。
- 代替索引レコードが定義されている索引レコードの場合、DUP 入出力エラー値の設定は VisualAge Generator とは異なります。VisualAge Generator の場合、**SET record SCAN** に続いて **SCAN** または **SCANBACK** 入出力オプションを指定しても、**SET record SCAN** ステートメントに対して DUP 入出力エラー値が設定されません。DUP 入出力エラー値は、最後に検索された重複キー・レコードを除く、それぞれの重複キー・レコードに対して設定されます。Java 生成の場合は、**set record position** に続いて **get next** または **get previous** ステートメントを指定すると、検索された最初の重複キー・レコード上ではなく **set record position** 上に **duplicate** 状態が設定されます。その他の重複キー・レコードに対しては、VisualAge Generator の場合と同様に **duplicate** 状態が設定されます。EGL の **duplicate** 状態は、最初と最後の重複キー・レコードを除くすべてのレコードに対して設定されます。索引レコードと代替索引レコードについて、また、これらのレコードを **set record position**、**get next**、および **get previous** とともに使用する方法的詳細については、オンライン・ヘルプを参照してください。
- SQL に以下の違いが生じます。
 - VisualAge Generator の場合、生成済み C++ では DB2 または ODBC のいずれかが使用されます。EGL デバッグおよび Java 生成では、JDBC が使用されます。これにより、デバッグおよびランタイムの両方の場合で違いが生じます。詳しくは、277 ページの『SQL サポートに関する違い』を参照してください。
- EZE 特殊データ・ワードには、以下の違いが生じます。
 - **EZECONVT** (EGL **sysVar.callConversionTable**)。VisualAge Generator の C++ 生成の場合、変換テーブル名の書式は **ELAxxyyy** です。ここで、**xx** はシステムを示し、**yyy** は言語を示します。EGL の Java 生成の場合、EGL が提供する変換テーブル名の書式は **CSOBxxxx** です。ここで、**CSO** は固定の接頭部で、**B** はターゲット・システムのバイト順序を示し、**xxxx** はターゲット・システムのコード・ページを示します。**B** の有効な値は、Unix システムの場合は **X**、Intel システムの場合は **I**、EBCDIC システムの場合は **E** です。EGL が **ELA** テーブル名を **CSO** テーブル名に自動的に変換するため、コードの変更は必要ありません。ただし、ユーザー独自の **CSO** 変換テーブルを作成する機能は EGL にはありません。
 - **EZERCODE** (EGL **sysVar.returnValue**)。VisualAge Generator の C++ 生成の場合、**EZERCODE** はシステムまたは呼び出し側プログラムに戻されます。プロ

グラムが異常終了した場合は、EZERCODE には値でなく VAGen 戻りコードが戻されます。EGL の Java 生成の場合、EZERCODE は無視されます。

第 7 部 付録

付録 A. 予約語

VisualAge Generator マイグレーション・ツールで拡張された予約語

VisualAge Generator マイグレーション・ツールは、EGL の予約語と競合するデータ項目、レコード、PSB、マップ、および関数の名前を変更します。また、その他のパーツが EGL の予約語と競合している場合には、エラー・メッセージを出します。実際には、マイグレーション・ツールは名前の競合を回避するために、拡張された予約語のリストを基にして、パーツの名前を変更するか、エラー・メッセージを出します。拡張された予約語のリストには以下の語が含まれています。

- EGL 予約語 (例えば、**program**、**sql**、YES など)
- EGL パーツのステレオタイプ (例えば、SQLRecord、BasicProgram、TextForm など)
- EGL プロパティー (例えば、**column**、**fieldLen** など)
- EGL 例外 (例えば、AnyException、FileIOException など)
- EGL 列挙名 (例えば、**colorKind**、**signKind** など)
- EGL システム・ライブラリー名およびシステム変数名 (例えば、**sysLib**、**sysVar** など)
- EGL 予約レコード (例えば、IO_PCBRECORD、DB_PCBRECORD など)
- SQL 予約語 (例えば、authorization、doubleprecision など)
- マイグレーション・ツールが特定の目的に使用する追加語 (例えば、プログラムの PSBRecord の変数名 psb など)

EGL 予約語

EGL には大量の予約語があります。予約語をパーツ名として使用することはできません。パーツ名が EGL 予約語である場合、マイグレーション・ツールは関数、データ項目、レコード、およびマップの名前を変更します。マイグレーション・ツールは、テーブル、マップ・グループ、またはプログラムの名前を変更しません。以下の語が EGL で予約されています。

表 71. EGL 予約語

文字	予約語
A	absolute, add, all, and, any, as
B	bigInt, bin, bind, blob, boolean, by, byName, byPosition
C	call, case, char, clob, close, const, continue, converse, current
D	DataItem, DataTable, date, dbChar, decimal, decrement, delete, display, dli
E	else, embed, end, escape, execute, exit, extends
F	false, field, first, float, for, forEach, form, FormGroup, forUpdate, forward, freeSql, from, function
G	get, goto, group
H	handler, hex, hold
I	if, implements, import, in, inOut, inParent, insert, int, interface, interval, into, is, isa

表 71. EGL 予約語 (続き)

文字	予約語
L	label、languageBundle、last、library、like
M	matches、mbChar、money、move
N	new、next、nil、no、not、nullable、num、number、numc
O	of、onEvent、onException、open、openUI、or、otherwise、out
P	pacf、package、passing、prepare、previous、print、private、program
R	record、ref、relative、replace、return、returning、returns
S	scroll、self、service、set、show、singleRow、smallFloat、smallInt、sql、sqlCondition、stack、static、string
T	this、time、timeStamp、to、transaction、transfer、true、try、type
U	unicode、update、url、use、using、usingKeys、usingPCB
W	when、where、while、with、wrap
Y	yes

注: EGL パーツ名は、EZE、 # シンボル、または @ シンボルで始めることはできません。

EGL 列挙型ワード

VisualAge Generator には、それぞれのプロパティごとに有効な値の固定リストがあります。EGL は、対応するプロパティの有効な値を指定する列挙型リストを提供しています。例えば、EGL は **align** プロパティの有効な値を **AlignKind** という名前の EGL 列挙型に格納します。また、EGL の場合はプロパティ値に変数と式を利用できます。EGL がプロパティ値を解決する際に、EGL はまずその名前の変数を検索します。その名前の変数がない場合、EGL はプロパティ値を検証するために、そのプロパティに対応する列挙型を検索します。例えば、書式の中で変数フィールドのプロパティが **align=center** である場合、EGL はまず書式の中から **center** という名前の変数を検索します。書式に **center** という名前の変数がない場合、EGL は **AlignKind.center** を使用します。表 72 に、EGL 列挙型の名前、列挙型の有効な値のリスト、および変数名と列挙型の値の衝突を引き起こす可能性があるプロパティを使用するパーツを示します。

また、EGL ライブラリーの定数と変数を使用して設定できるプロパティがいくつかあります。表 73 に、EGL ライブラリーの定数と変数を使用する EGL プロパティを参照用にリストします。ただし、マイグレーション・ツールが使用する EGL ライブラリーの変数と定数は、表の中でアスタリスク (*) によって示されているもののみです。

次のいずれかの手法を使用して、コードを新しく作成する際に競合が起こらないようにすることができます。

- 名前の衝突を起こす可能性がある変数を定義するときに、有効な値のリストのいずれかと一致する変数名を使用しない。
- プロパティの列挙値を常に修飾する (例えば、常に **align = AlignKind.center** を指定する)。

- 競合が生じたときに、プロパティの列挙値を修飾する (例えば、特定の書式に `center` という名前のフィールドが存在するときに、**align = AlignKind.center** を指定する)。

マイグレーション・ツールは、予想される使用頻度、マイグレーション・ツールが実際に使用するプロパティ、およびマイグレーションのパフォーマンスに関する考慮事項に基づいて、これら 3 つの技法を組み合わせ使用します。パーツと変数の名前を変更する必要性を最小限にするために、マイグレーション・ツールは次の処理を行います。

- YES と NO は EGL 予約語なので、これらの名前が付いたパーツまたは変数を常に名前変更します。
- PFn という名前のパーツまたは変数を常に名前変更します (ここで、*n* は 1 から 24 までです)。実質上、マイグレーション・ツールはこれらの値を予約語として扱います。これにより、マイグレーションのパフォーマンスに影響を与えることなく、プログラムと書式の **helpKey** プロパティと **validationBypassKeys** プロパティの見やすさが向上します。
- FormGroup 内で、マイグレーション・ツールは次の処理を行います。
 - **deviceType** プロパティ値を常に完全に修飾します。これにより、FormGroup 内のすべての書式にあるすべてのフィールド名を検討する必要がなくなるので、パフォーマンスが向上します。
 - **helpKey** プロパティまたは **validationBypassKeys** プロパティは使用しません。これは、マップ・グループ用の同等な VAGen プロパティが存在しないからです。
- 書式内では、書式内の 1 つ以上のフィールド名が、書式で使える列挙値、EGL 定数、または EGL 変数のいずれかと同じ名前である場合に限り、マイグレーション・ツールは書式のプロパティ値を修飾します。フォームのいずれかのフィールドに関して競合が起こった場合、マイグレーション・ツールは、対応する列挙型またはライブラリー名のフォーム内にあるすべてのフィールドを対象に、すべてのプロパティ値を完全に修飾します。
- VGUI レコード内では、マイグレーション・ツールは、レコードの 1 つ以上のフィールド名が、VGUI レコードで使える列挙値または EGL 変数のいずれかと同じ名前である場合にのみ、プロパティ値を修飾します。レコードのいずれかのフィールドに関して競合が起こった場合、マイグレーション・ツールは、対応する列挙型またはライブラリー名を使用して、そのレコード内のすべてのフィールドについてすべてのプロパティ値を完全に修飾します。
- PSBRecord 内で、マイグレーション・ツールは常に **pcbType** プロパティを完全修飾します。
- プログラム内で、マイグレーション・ツールは常に **callInterface** プロパティを完全修飾します。
- 他のパーツ型の中ではパーツ名と列挙値が競合する可能性はないので、マイグレーション・ツールはプロパティ値の修飾を行いません。

注: 298 ページの表 72 と 299 ページの表 73 は参照用です。このため、マイグレーション・ツールによって作成されることのない Library、ConsoleUI、JSFHandler、Service、および Interface パーツがテーブルにインクルードされます。

表 72. EGL 列挙型

EGL 列挙型	変数名の制約となる有効な値のリスト	衝突を起こす可能性があるパーツ
AlignKind	center、left、none、right	Form
CallingConventionKind	I4GL (lib は今後の使用のために予約)	Library
CaseFormatKind	defaultCase、lower、upper	ConsoleUI
ColorKind	black、blue、cyan、defaultColor、green、magenta、red、white、yellow 注: black は ConsoleUI の場合のみ有効です。	Form ConsoleUI
ConvertDirection	local、remote	適用不可
DataSource	databaseConnection、reportData、sqlStatement	Report
DeviceTypeKind	doubleByte、singleByte	FormGroup
DisplayUseKind	button、hyperlink、input、output、secret、table	JSFHandler とともに使用されるレコード
DLICallInterfaceKind	AIBTDLI、CBLTDLI	Program
EventKind	afterDelete、afterField、afterInsert、afterOpenUI、afterRow、beforeDelete、beforeField、beforeInsert、beforeOpenUI、beforeRow、menuAction、onKey	ConsoleUI
ExportFormat	html、pdf、text、xml	Report
HighlightKind	blink、defaultHighlight、noHighlight、reverse、underline	Form ConsoleUI
IndexOrientationKind	across、down	Form
IntensityKind	bold、defaultHighlight、dim、invisible、normalIntensity	Form ConsoleUI
LineWrapKind	character、compress、word	ConsoleUI
OrderingKind	byKey、byInsertion、none	Dictionary
OutlineKind	bottom、left、right、top、box、noOutline	Form
PCBKind	TP、DB、GSAM	PSBRecord
PfKeyKind	pfn ここで (1 <= n <=24)	Form FormGroup Program
ProtectKind	noProtect、protect、skipProtect	Form
ScopeKind	application、request、session	JSFHandler とともに使用されるレコード
SelectTypeKind	index、value	JSFHandler とともに使用されるレコード
SignKind	leading、none、parens、trailing	Form VGUI レコード JSFHandler とともに使用されるレコード
UITypeKind	hidden、input、inputOutput、none、output、programLink、submit、submitBypass、uiForm	VGUI レコード

表 72. EGL 列挙型 (続き)

EGL 列挙型	変数名の制約となる有効な値のリスト	衝突を起こす可能性があるパーツ
WindowAttributeKind	color、commentLine、errorLine、 formLine、highlight、intensity、 menuLine、messageLine、promptLine	ConsoleUI

表 73. sysLib 定数および sysVar 変数を使用する EGL 列挙型

EGL プロパティ	EGL 定数および変数 (* = マイグレーション・ツールによって使用される値)	衝突を起こす可能性があるパーツ
dateFormat	<ul style="list-style-type: none"> • strLib.defaultDateFormat* • strLib.eurDateFormat • strLib.isoDateFormat • strLib.jisDateFormat • strLib.usaDateFormat • vgVar.systemGregorianCalendar* • vgVar.systemJulianCalendar* 注: マイグレーション・ツールは、DataItem および VGUI レコード・パーツでは defaultDateFormat のみを使用します。	ConsoleUI Form、 VGUI レコード
fillCharacter	<ul style="list-style-type: none"> • strLib.nullFill* 	Form
timeFormat	<ul style="list-style-type: none"> • strLib.defaultTimeFormat • strLib.eurTimeFormat • strLib.isoTimeFormat • strLib.jisTimeFormat • strLib.usaTimeFormat 	JSFHandler とともに使用されるレコード
timeStampFormat	<ul style="list-style-type: none"> • strLib.db2TimeStampFormat • strLib.odbcTimeStampFormat 	ConsoleUI

SQL 予約語

EGL では SQL 文節の中での使用が許可されていない SQL 予約語が多数存在します。パーツ名が SQL 予約語である場合、マイグレーション・ツールは関数、データ項目、レコード、PSB、およびマップの名前を変更します。マイグレーション・ツールは、テーブル、マップ・グループ、またはプログラムの名前を変更しません。以下の語が SQL で予約されています。

文字	予約語
A	absolute、action、add、alias、all、allocate、alter、and、any、are、as、asc、assertion、at、authorization、avg
B	begin、between、bigint、binaryLargeObject、bit、bit_length、blob、boolean、both、by
C	call、cascade、cascaded、case、cast、catalog、char、char_length、character、character_length、characterLargeObject、characterVarying、charLargeObject、charVarying、check、clob、close、coalesce、collate、collation、column、comment、commit、connect、connection、constraint、constraints、continue、convert、copy、corresponding、count、create、cross、current、current_date、current_time、current_timestamp、current_user、cursor

文字	予約語
D	data、database、date、dateTime、day、deallocate、dec、decimal、declare、default、deferrable、deferred、delete、desc、describe、diagnostics、disconnect、distinct、domain、double、doublePrecision、drop
E	else、end、endExec、escape、except、exception、exec、execute、exists、explain、external、extract
F	false、fetch、first、float、for、foreign、found、from、full
G	get、getCurrentConnection、global、go、goto、grant、graphic、group
H	having、hour
I	identity、image、immediate、in、index、indicator、initially、inner、input、insensitive、insert、int、integer、intersect、into、is、isolation
J	join
K	key
L	language、last、leading、left、level、like、local、long、longint、lower、ltrim
M	match、max、min、minute、module、month
N	national、nationalCharacter、nationalCharacterLargeObject、nationalCharacterVarying、nationalCharLargeObject、nationalCharVarying、natural、nchar、ncharVarying、nclob、next、no、not、null、nullIf、number、numeric
O	octet_length、of、on、only、open、option、or、order、outer、output、overlaps
P	pad、partial、position、prepare、preserve、primary、prior、privileges、procedure、public
R	raw、read、real、references、relative、restrict、revoke、right、rollback、rows、rtrim、runtimeStatistics
S	schema、scroll、second、section、select、session、session_user、set、signal、size、smallint、some、space、sql、sqlcode、sqlcondition、sqlerror、sqlstate、substr、substring、sum、system_user
T	table、tablespace、temporary、terminate、then、time、timestamp、timezone_hour、timezone_minute、tinyint、to、trailing、transaction、translate、translation、trim、true
U	uncatalog、union、unique、unknown、update、upper、usage、user、using
V	values、varbinary、varchar、varchar2、varGaphic、varying、view
W	when、whenever、where、with、work、write
Y	year
Z	zone

特殊な処理を必要とする SQL 予約語

次の SQL 予約語を SQL 表名または列名として使用する場合、EGL では特殊な処理が必要です。

call、from、group、having、insert、order、select、set、union、update、values、where

これらの SQL 予約語を使用するには、次の手法を使用します。

- SQLRecord 内の項目の **column** プロパティを指定するには、次のように指定します。

```
column = "\"reservedWord\""
```

例えば、次のようになります。

```
column = "\"FROM\""
```

- SQLRecord の **tableNames** プロパティを指定するには、次のように指定します。

```
tableNames = [{"reservedWord2\""}]
```

例えば、次のようになります。

```
tableNames = [{"ORDER\""}]
```

- レコードの **defaultSelectCondition** の中で SQL 列名として予約語のいずれかを使用するには、次のように指定します。

```
defaultSelectCondition = #sqlCondition{ "reservedWord" = ... }
```

例えば、次のようになります。

```
defaultSelectCondition = #sqlCondition{ "FROM" = ... }
```

- SQL 入出力ステートメントの中で SQL 列名として予約語のいずれかを使用するには、次のように指定します。

```
... #sql{ select "reservedWord" from "reservedWord2" .... } ...
```

例えば、次のようになります。

```
... #sql{ select "FROM" from "ORDER" .... } ...
```

Java 予約語

Java には、パッケージ名として使用できない予約語があります。Java の生成を行う場合は、次の名前の使用は避けてください。

文字	予約語
A	abstract
B	boolean、break、byte
C	case、catch、char、class、const、continue
D	default、do、double
E	else、extends
F	false、final、finally、float、for
G	goto
I	if、implements、import、instanceof、int、interface
L	long
N	native、new、null
P	package、private、protected、public
R	return
S	short、static、super、switch、synchronized
T	this、throw、throws、transient、true
V	void、volatile
W	while

付録 B. VisualAge Generator と EGL の言語要素の関係

この付録の表には、次の 3 つの欄があります。

- VisualAge Generator 4.5 -- この欄は、VAGen 言語要素を示しています。パーツ型に関連した部分では、表の編成と使用される用語は VAGen ユーザー・インターフェースに対応しています。ステートメント、EZE ワード、およびサービス・ルーチンの表は、ステートメント、EZE ワード、またはサービス・ルーチンのタイプ別に編成されています。
- 마이그레이션・ツールによって作成される EGL -- この欄は、EGL 言語要素を示しています。この欄は마이그레이션に必要な情報のみを示しており、完全な EGL 構文を示すことは目的としていません。一部の EGL 言語要素に対しては、他のプロパティ、値、およびオプションを使用できる場合があります。例えば、EGL **set** ステートメントには、VisualAge Generator では使用できない追加のオプションがあります。この付録の表にリストされている **set** ステートメントのオプションは、VAGen 言語要素に対応するもののみです。この欄は、EGL 資料にある詳細情報を見つけるためのガイドとして使用してください。
- 마이그레이션・ツールに関する考慮事項 -- この欄には、마이그레이션・ツールが VisualAge Generator から EGL への変換を処理する方法に関する追加情報があります。また、必要に応じて、関連パーツを使用した마이그레이ション、関連パーツを使用しない마이그레이ション、および VAGen 言語要素の마이그레이ション時に起こりうる問題について詳しく説明する、未確定状態に関するセクションの参照先も示しています。

それぞれのパーツ型ごとに、セクションの最初にある表の 1 行目に次の情報があります。

- VisualAge Generator 4.5 - VisualAge Generator のパーツ型に対して、さまざまなウィンドウで指定できる情報の概要。
- EGL - 対応する EGL パーツ型の EGL 構文の全体 (마이그레이ション・ツールが使用する構文)。これとは異なる構文を使用できる場合があります。例えば、VAGen テーブルを마이그레이ションする際に、마이그레이ション・ツールは DataTable の内容を DataTable 構造の後に常に配置するので、『テーブル』のセクションにはこの構文が示されています。EGL 構文上は、DataTable の内容を DataTable 構造の前に配置することもできます。

表の中では、次の構文が使用されます。

- | - いくつかのオプションからの選択。選択の順序は、VisualAge Generator 4.5 と EGL の両方の欄で同じです。
- 黒丸付きリスト - オプションまたは値の長いリストからの選択。選択の順序は、VisualAge Generator 4.5 と EGL の両方の欄で同じです。
- 斜体 - VisualAge Generator から마이그레이ションするときに마이그레이ション・ツールが入力する値、または EGL ステートメントを新規に作成するときにユーザーが入力する値。
- 太字 - 示されたとおりに指定する必要がある EGL キーワードと記号。
- { } - 0 回から n 回繰り返すことができる情報を囲みます。

- { } - EGL プロパティ・リストを囲みます。プロパティは常にコンマで区切られます。
- [] - オプションの情報を囲みます。
- [] - EGL の値のリストを囲みます。値は常にコンマで区切られます。

一般的な構文規則

VisualAge Generator と EGL の全体的な構文には、いくつかの違いがあります。

表 74. 一般的な構文規則

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>コメントは次のフォーマットで指定されます。</p> <ul style="list-style-type: none"> • プログラム、テーブル、およびレコードの場合は、プロログ。 • 項目および関数の場合は、記述。 • 関数内のコメントは、次の記号によって示されます。 <ul style="list-style-type: none"> – セミコロン (;)。同じ行でセミコロンの後にあるものは、すべてコメントとして扱われます。 – /*。同じ行で /* の後にあるものは、すべてコメントとして扱われます。 	<p>コメントは次のフォーマットで指定されます。</p> <ul style="list-style-type: none"> • // は行コメントを示します。同じ行で // の後にあるものは、すべてコメントとして扱われます。コメントは 1 行のみです。 • /* コメント */。/* の後にあるものは、次の */ まですべてコメントとして扱われます。複数行にわたるコメントを入力できます。 	<p>マイグレーション・ツールは、以下の方法で変換を行います。</p> <ul style="list-style-type: none"> • プロログとパーツ記述は、EGL の // 行コメントに変換されます。 • レコード、テーブル、マップ、関数ローカル・ストレージ、関数仮パラメーター、または関数からの戻り値などのフィールドとして使用される項目の記述は、EGL // 行コメントに変換されます。 • 関数内のコメントは、/* コメント */ に変換されます。 • マイグレーション・ツールによって追加された通知コメントは EGL // 行コメントの形式になります。
<p>小数点は、ロケールに応じてピリオドまたはコンマのどちらかを使用できます。</p>	<p>開発時の小数点は、常にピリオドです。生成時および実行時には、実行時のロケールと decimalSymbol および symbolSeparator ビルド記述子オプションに応じてピリオドまたはコンマのどちらかが使用されます。</p>	<p>デフォルトで小数点にコンマを使用するロケールの場合、またはマイグレーション構文の設定「小数点位のコンマを小数点に変換 (Convert decimal comma to decimal point)」を選択した場合は、マイグレーション時にマイグレーション・ツールがコンマをピリオドに変換します。</p>
<p>プロパティは、専用のエディターで、チェック・ボックス、ドロップダウン・リストなどを使用して入力します。</p>	<p>プロパティはテキスト・エディターで入力し、コンマで区切る必要があります。</p> <p>DataItem パーツ用のソース・アシスタントやフォーム・パーツ用の EGL フォーム・エディターなど、専用のエディターがいくつか存在します。</p>	<p>特別な考慮事項なし。</p>
<p>値のリストは、専用のエディターで入力します。例えば、SQL レコードの表名は「SQL 行プロパティ」ウィンドウに入力します。</p>	<p>値のリストは大括弧 [] で囲む必要があります。</p>	<p>特別な考慮事項なし。</p>

表 74. 一般的な構文規則 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
他のパーツを参照するプロパティ値は、専用のエディターで入力します。例えば、マップ変数フィールドの編集ルーチンは、「変数フィールドのプロパティ (Variable Field Properties)」ウィンドウの「編集 (Edits)」ページに入力します。	現行パーツの命名スコープ内になければならないその他のパーツまたは変数を参照するプロパティ値は、引用符で囲みません (例えば、SQL レコード内の keyItems プロパティなど)。現行パーツの命名スコープ外にある可能性もあるその他のパーツまたは変数を参照するプロパティ値は、二重引用符で囲む必要があります (例えば、相対レコードの recordNumItem プロパティなど)。	特別な考慮事項なし。
ステートメント内の名前の解決は、コンテキストに依存し、ステートメント・タイプに基づいて行われます。	ステートメント内の名前の解決は、ステートメント・タイプに関係なく、常に同じ規則に従って行われます。	特別な考慮事項なし。ネーム・レゾリューションに違いがあると、ほとんどの場合、「問題」ビューに EGL 検査メッセージが表示されます。メッセージが表示された場合は、537 ページの『メッセージの参照情報 - ネーム解決規則および名前の修飾規則』を参照してください。

データ項目

データ項目に関するセクションは、次の表で構成されています。

- データ項目 - 一般構文、データ型、長さ、小数部、および説明 (306 ページの表 75)
- デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 一般情報 (308 ページの表 76)
- デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 一般編集 (308 ページの表 77)
- デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 数値の編集 (311 ページの表 78)
- デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - エラー・メッセージ (312 ページの表 79)
- ユーザー・インターフェース・プロパティ - ラベルとヘルプ (313 ページの表 80)

注: EGL の場合、DataItem パーツの編集とメッセージのプロパティは、1 セットのみ存在します。マイグレーション・ツールはデータ項目のマップ・プロパティと UI プロパティをマージします。

表 75. データ項目 - 一般構文、データ型、長さ、小数部、および説明

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VAGen データ項目パーツ:</p> <ul style="list-style-type: none"> • itemName • 基本情報: <ul style="list-style-type: none"> – データ型 – 長さ – 小数部 – 説明 • デフォルトのマップ・プロパティ • ユーザー・インターフェース (UI) プロパティ 	<p>EGL 構文の例:</p> <pre>// Description DataItem itemName dataType(lengthInformation) { [{formattingProperties}] [{validationProperties}] [{JSFHandlerFieldProperties}] } end</pre>	<p>マイグレーション・ツールは、VAGen のデータ型、長さ、および小数部を使用して、EGL の <i>dataType</i> と <i>lengthInformation</i> を決定します。</p> <p>マイグレーション・ツールは、VAGen のデフォルトのマップ・プロパティと UI プロパティをマージして、単一の EGL フォーマット設定プロパティ、検証プロパティ、および JSFHandler フィールド・プロパティのセットにします。</p>
<p>文字項目の型:</p> <ul style="list-style-type: none"> • Char • Hex • DBCS • Mixed • Unicode (VisualAge for Java のみ) <p>長さは文字数です。レコード・エディター内ではバイト数も表示できます。</p>	<p>対応する文字項目の型:</p> <ul style="list-style-type: none"> • CHAR • HEX • DBCHAR • MBCHAR • UNICODE <p>長さは文字数です。</p>	<p>マイグレーション・ツールは、文字データ項目を対応する型と長さに変換します。</p>
<p>数字 (ブーン 10 進数) の型:</p> <ul style="list-style-type: none"> • Num • Numc <p>長さは総桁数で、最大は 18 です。小数部は、小数点の右側の桁数です。レコード・エディター内ではバイト数も表示できます。</p>	<p>対応する数値型:</p> <ul style="list-style-type: none"> • NUM • NUMC <p>精度は総桁数です。スケールは、小数点の右側の桁数です。</p> <p>NUM フィールドの最大精度は、デバッグおよび Java 生成の場合は 32、COBOL 生成の場合は 31 です。NUMC フィールドの最大精度は 18 です。</p>	<p>マイグレーション・ツールは、対応する型、精度、およびスケールへの変換を行います。小数部が 0 の場合、マイグレーション・ツールはスケールを省略します。</p>

表 75. データ項目 - 一般構文、データ型、長さ、小数部、および説明 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>パック 10 進数の型:</p> <ul style="list-style-type: none"> • Pacf • Pack <p>長さは総桁数で、最大は 18 です。小数部は、小数点の右側の桁数です。Pacf の長さは、奇数または 18 であることが必要です。Pack の長さは、偶数でも奇数でも構いません。長さ 18 の場合を除き、偶数の長さはデータ項目定義に記録されますが、テストおよび生成時、およびデータ項目エディターとレコード・エディター内では、次に大きな奇数の長さとして扱われます。SQL レコード・エディターのみが偶数の長さを表示し、SQL レコードのみがテストと生成の際に偶数の長さをサポートします。偶数の長さは、SQL WHERE 文節と、Execution Time Statement Build オプションを使用する SQL 関数内でのみ使用されます。レコード・エディター内ではバイト数も表示できます。</p>	<p>対応する数値型:</p> <ul style="list-style-type: none"> • PACF • DECIMAL <p>精度は総桁数です。スケールは、小数点の右側の桁数です。</p> <p>PACF フィールドの最大精度は 18 です。DECIMAL フィールドの最大精度は、デバッグおよび Java 生成の場合は 32、COBOL 生成の場合は 31 です。</p> <p>PACF の長さは、奇数または 18 になっている必要があります。DECIMAL の長さは、偶数でも奇数でも構いません。DataItem パーツ定義とすべてのレコード・タイプに関して、偶数の長さがサポートされます。</p> <p>テストおよび生成時に、VisualAge Generator 互換モードを使用している場合、EGL は偶数の精度をもつ DECIMAL 項目に対して次の処理を行います。</p> <ul style="list-style-type: none"> • すべてのレコードの精度を 1 だけ増やします。 • EGL は、SQL WHERE 文節または PREPARE ステートメント内で偶数の精度の一時変数を使用します。 	<p>マイグレーション・ツールは、対応する型、精度、およびスケールへの変換を行います。小数部が 0 の場合、マイグレーション・ツールはスケールを省略します。Pack 項目の場合、DataItem パーツ定義に偶数の長さが記録されていれば、マイグレーション・ツールは、デフォルトでその長さを偶数の長さとしてマイグレーションします。</p> <p>マイグレーション設定で「項目または変数の evensql=y を受け入れない」を選択すると、マイグレーション・ツールは Pack 項目用の奇数精度 (項目が最大長である場合は 18) を自動的に使用し、影響を受けるデータ項目パーツや非共用レコード項目に関する警告メッセージを出します。</p>
<p>バイナリー項目の型:</p> <ul style="list-style-type: none"> • Bin、長さ 4、小数部なし • Bin、長さ 9、小数部なし • Bin、長さ 18、小数部なし • Bin、長さ 4、9、または 18、小数部あり 	<p>対応するバイナリー形式:</p> <ul style="list-style-type: none"> • SMALLINT (精度またはスケールなし) • INT (精度またはスケールなし) • BIGINT (精度またはスケールなし) • BIN (精度またはスケールあり) 	<p>マイグレーション・ツールは、小数部の長さと数に基づいて、バイナリー・データ項目を対応する型に変換します。BIN 型は、小数部 (スケール) が指定されている場合에만使用されます。</p>
説明	適用不可	マイグレーション・ツールは、項目の説明をコメントに変換し、DataItem 定義の前に配置します。

表 76. デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 一般情報

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>データ項目には、デフォルト・マップ・プロパティとユーザー・インターフェース (UI) プロパティの両方を指定できます。以下の種類のプロパティがサポートされています。</p> <ul style="list-style-type: none"> フォーマット設定編集 検証編集 エラー・メッセージ <p>UI プロパティには、ラベルとヘルプ・テキストも含まれます。</p> <p>VisualAge Generator 内で一部のプロパティを明示的に設定すると、他のプロパティも設定されます。例えば、数字分離記号を設定すると、充てん文字、入力必須 (input required)、位置調整、通貨記号、および符号も明示的に設定されます。</p>	<p>DataItem パーツには以下の種類のプロパティを設定できます。</p> <ul style="list-style-type: none"> フォーマット設定プロパティ 検証プロパティ JSFHandler フィールド・プロパティ <p>一部のプロパティのカテゴリが、VisualAge Generator から変更されています。例えば、エラー・メッセージは検証プロパティと一緒にグループにあります。JSFHandler フィールド・プロパティは、UI ラベルとヘルプ・テキストをインクルードしています。</p> <p>以下の表の EGL の列は、EGL プロパティのカテゴリを示します。</p>	<p>マイグレーション・ツールは、デフォルトのマップ・プロパティと UI プロパティをマージし、UI プロパティを優先します。検証編集とそれに関連したエラー・メッセージは、対してマイグレーションされます。マイグレーション・ツールは、VisualAge Generator 内で明示的に設定されたプロパティのみをマイグレーションしません。EGL プロパティのデフォルト値は、ツールによって自動的に挿入されません。詳細と起こりうる問題については、81 ページの『共用編集とメッセージ』にある、マップと UI 編集のマージに関する情報を参照してください。</p> <p>また、詳細と起こりうる問題については、83 ページの『共用データ項目のマップ編集ルーチン』にある、共用データ項目のマップ項目編集に関する情報も参照してください。</p>

表 77. デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 一般編集

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>編集タイプ (UI のみ) - 値</p> <ul style="list-style-type: none"> None Boolean Date Time 	<p>EGL は、複数のプロパティをサポートします。</p> <ul style="list-style-type: none"> 適用不可 isBoolean = yes dateFormat = defaultDateFormat timeFormat = "HH:mm:ss" <p>(フォーマット設定プロパティ)</p>	特別な考慮事項なし。
編集関数 (UI のみ)	validatorFunction (検証プロパティ)	特別な考慮事項なし。
編集テーブル (UI のみ)	validatorDataTable (検証プロパティ)	特別な考慮事項なし。
Web 上での編集関数の実行 (UI のみ)	runValidatorFromProgram	EGL プロパティは、VAGen プロパティの逆です。マイグレーション・ツールは yes を no に、no を yes に変換します。

表 77. デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 一般編集 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
編集ルーチン (マップのみ)	validatorFunction または validatorDataTable (検証プロパティ)	<p>UI 編集関数と編集テーブルが指定されていなければ、マイグレーション・ツールは以下の方法で EGL プロパティを設定します。</p> <ul style="list-style-type: none"> • マップ編集ルーチンが EZEC10 または EZEC11 であれば、validatorFunction プロパティを設定します。 • 編集ルーチンが関数であれば、validatorFunction プロパティを設定します。 • 編集ルーチンがテーブルであれば、validatorDataTable プロパティを設定します。 <p>UI 編集関数または編集テーブルが指定されている場合、マイグレーション・ツールはマップ編集ルーチンをマイグレーションしません。</p> <p>マイグレーション時に編集ルーチンが使用できない場合は、特別な考慮事項が適用されます。詳細と起こりうる問題については、83 ページの『共用データ項目のマップ編集ルーチン』にある、マップ編集ルーチンに関する情報を参照してください。</p>
<p>位置調整 - 左揃え 右揃え なし (マップのみ)</p> <p>注:</p> <ul style="list-style-type: none"> • マップ項目のデフォルトは、数値フィールドの場合は右揃え、その他のフィールドの場合は左揃えです。 • UI 項目の場合、位置調整はサポートされません。 	<p>align = left right none (フォーマット設定プロパティ)</p> <p>注:</p> <ul style="list-style-type: none"> • 書式フィールドのデフォルトは、数値フィールドの場合は右揃え、その他のフィールドの場合は左揃えです。 • JSFHandler フィールドと VGUI レコード・フィールドについては、align はサポートされていません。 	特別な考慮事項なし。

表 77. デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 一般編集 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>日付編集マスク (マップのみ)</p> <p>以下の値が有効です。</p> <ul style="list-style-type: none"> • SYSGREGRN • SYSJULIAN • dateEditPattern 	<p>dateFormat = <i>value</i></p> <p>以下の値が有効です。</p> <ul style="list-style-type: none"> • systemGregorianCalendar • systemJulianCalendar • "dateEditPattern" <p>(フォーマット設定プロパティ)</p> <p>注: <i>dateEditPattern</i> の中で、マイグレーション・ツールは次の EGL 表記への変換を行います。</p> <ul style="list-style-type: none"> • 年を示す yy または yyyy。 • 月を示す MM。 • 日を示す dd。 • 年の通算日を示す DDD。 	<p>UI 編集タイプで「日付」が指定されていない場合、マイグレーション・ツールは VisualAge Generator 内で指定されている日付編集マスク (存在する場合) に基づいて dateFormat を設定します。UI 編集タイプが「日付」を指定している場合、マイグレーション・ツールはマップの日付編集マスクをマイグレーションしません。</p>
最小入力	minimumInput (検証プロパティ)	特別な考慮事項なし。
<p>充てん文字</p> <p>注:</p> <ul style="list-style-type: none"> • UI レコード内で使用される項目のデフォルト充てん文字は、文字、混合、および数値の各フィールドの場合はブランクです。16 進フィールドの場合、デフォルト充てん文字はゼロです。DBCS および Unicode のフィールドの場合は、ブランクが必須の充てん文字です。NULL は無効な充てん文字です。 • 文字、DBCS、または混合の各フィールドの場合、マップに使用される項目のデフォルト充てん文字は NULL です。デフォルト充てん文字は、数値フィールドの場合はブランク、16 進フィールドの場合はゼロです。 	<p>fillCharacter</p> <p>(フォーマット設定プロパティ)</p> <p>注:</p> <ul style="list-style-type: none"> • 特定のページ、書式、または VGUI レコード内でオーバーライドされない限り、JSFHandler フィールド、フォーム・フィールド、VGUI レコード・フィールドには同じデフォルト fillCharacter が使用されます。 • strLib.nullFill は、NULL 充てん文字を表す EGL 定数です。あるいは、"" (2 つの連続する二重引用符) を使用します。 • VGUI レコードでは、UNICODE フィールドで非ブランク文字を使用することが許可されています。 	<p>マイグレーション・ツールは、N を以下のいずれかの値に変換します。</p> <ul style="list-style-type: none"> • UI 充てん文字用の N • マップ・フィールド文字用の nullFill <p>EGL にはデフォルトの充てん文字が 1 つしかないため、特殊な考慮事項が適用されます。詳細と起こりうる問題については、85 ページの『共用データ項目の充てん文字』の未確定データ項目と充てん文字に関する情報を参照してください。</p>
大文字変換	upperCase	特別な考慮事項なし。
	(フォーマット設定プロパティ)	
16 進編集 (マップのみ)	isHexDigit (検証プロパティ)	特別な考慮事項なし。
入力必須	inputRequired (検証プロパティ)	特別な考慮事項なし。
SO/SI スペースの検査	needsSOSI (検証プロパティ)	特別な考慮事項なし。

表 78. デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 数値の編集

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>最小値と最大値</p> <p>注: 最小値または最大値のどちらかを指定する場合は、両方を指定する必要があります。</p>	<p>validValues = <code>[[minimumValue, maximumValue]]</code> (検証プロパティ)</p> <p>注: 複数の値の対、および単一値を validValues プロパティにリストできます。</p>	<p>マイグレーション・ツールは、最小値と最大値を EGL の validValues プロパティに結合します。</p>
<p>符号 - なし 先頭 末尾</p> <p>注:</p> <ul style="list-style-type: none"> UI レコード内の数値項目に対するデフォルト記号は leading です。 マップ上の数値項目に対するデフォルト記号は none です。 	<p>sign = <code>none leading trailing</code> (フォーマット設定プロパティ)</p> <p>注: 数値フィールドのデフォルトの sign は常に leading です。</p>	<p>数値フィールドについて、マイグレーション・ツールは次のうち最初に該当する基準に基づいてマイグレーションを行います。</p> <ul style="list-style-type: none"> UI 記号編集が指定されている場合、ツールは対応する sign 記号プロパティにマイグレーションします。 UI 編集タイプとして日付、時刻、またはブール値が指定されている場合、ツールは sign プロパティを none に設定します。 他の UI 記号編集が指定されている場合、ツールは sign プロパティを leading に設定します。 マップ記号編集が指定されている場合、ツールは対応する sign プロパティにマイグレーションします。 マップ記号編集が指定されていない場合、ツールは sign プロパティを none に設定します。
<p>通貨 (マップと UI 両方) 通貨記号 (UI のみ)</p>	<p>currency = <code>yes no</code> currencySymbol = <code>"symbol"</code> (フォーマット設定プロパティ)</p> <p>注:</p> <ul style="list-style-type: none"> currencySymbol は書式にも適用されます。 currency = <code>yes</code> で、currencySymbol が指定されていない場合、実行時に実際に使用される通貨記号は、VisualAge Generator の場合と同じように設定されます。 	<p>マイグレーション・ツールは、次のうち最初に該当する基準に基づいてマイグレーションを行います。</p> <ul style="list-style-type: none"> UI の通貨記号が指定されている場合、ツールは currency = <code>yes</code>, currencySymbol = <code>"symbol"</code> にマイグレーションします。 UI の通貨編集が <code>yes</code> または <code>no</code> に設定されている場合、ツールは currency プロパティをそれぞれ <code>yes</code> または <code>no</code> に設定します。 マップの通貨編集が <code>yes</code> または <code>no</code> に設定されている場合、ツールは currency プロパティをそれぞれ <code>yes</code> または <code>no</code> に設定します。
セパレーター	numericSeparator (フォーマット設定プロパティ)	特別な考慮事項なし。

表 78. デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - 数値の編集 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
ゼロ編集	zeroFormat (フォーマット設定プロパティ)	特別な考慮事項なし。

表 79. デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - エラー・メッセージ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
編集テーブル (UI のみ)	validatorDataTableMsgKey (検証プロパティ)	特別な考慮事項なし。
EZE 関数 (UI のみ)	validatorFunctionMsgKey (検証プロパティ)	特別な考慮事項なし。
編集ルーチン (マップのみ)	validatorDataTableMsgKey または validatorFunctionMsgKey (検証プロパティ)	<p>特別な考慮事項が適用されます。マイグレーション・ツールがマップ編集ルーチン・メッセージをマイグレーションするかどうかを決定する方法については、81 ページの『共用編集とメッセージ』を参照してください。 マイグレーション・ツールがマップ編集ルーチン・メッセージをマイグレーションする場合、ツールは以下の方法で EGL プロパティを設定します。</p> <ul style="list-style-type: none"> 編集ルーチンが EZEC10 または EZEC11 であれば、validatorFunctionMsgKey を設定します。 編集ルーチンがテーブルであれば、validatorDataTableMsgKey を設定します。 編集ルーチンが関数ならば、この状態では VisualAge Generator 内でメッセージは使用されないの、編集ルーチン・メッセージをマイグレーションしません。 <p>特別な考慮事項が適用されます。詳細と起こりうる問題については、83 ページの『共用データ項目のマップ編集ルーチン』にある、未確定データ項目とマップ編集ルーチンに関する情報を参照してください。</p>
最小入力	minimumInputMsgKey (検証プロパティ)	特別な考慮事項なし。
入力必須	inputRequiredMsgKey (検証プロパティ)	特別な考慮事項なし。
データ型	typeChkMsgKey (検証プロパティ)	特別な考慮事項なし。

表 79. デフォルトのマップ・プロパティおよびユーザー・インターフェース・プロパティ - エラー・メッセージ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
数値範囲	validValuesMsgKey (検証プロパティ)	特別な考慮事項なし。

表 80. ユーザー・インターフェース・プロパティ - ラベルとヘルプ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
UI ラベル	displayName (JSFHandler フィールド・プロパティ)	特別な考慮事項なし。
ヘルプ・テキスト	help (JSFHandler フィールド・プロパティ)	特別な考慮事項なし。

レコード

レコードに関するセクションは、次の表で構成されています。

- レコード - 一般構文、レコード・タイプ、プロパティ、およびプロログ (314 ページの表 81)
- レコード - ほとんどのレコード・タイプのレコード構造 (316 ページの表 82)
- レコード - SQL プロパティと SQL レコード構造 (318 ページの表 83)
- レコード - DL/I プロパティと DL/I レコード構造 (322 ページの表 84)
- レコード - UI レコード・プロパティと UI レコード構造 (326 ページの表 85)
- レコード - UI 項目プロパティ - 一般 (327 ページの表 86)
- レコード - UI 項目プロパティ - 編集 (327 ページの表 87)
- レコード - UI 項目プロパティ - エラー・メッセージ (329 ページの表 88)
- レコード - UI 項目プロパティ - ヘルプ (330 ページの表 89)
- レコード - UI 項目プロパティ - 実行依頼 (330 ページの表 90)
- レコード - UI 項目プロパティ - プログラム・リンク (330 ページの表 91)

注: マイグレーション・ツールは、VAGen レコードを常に EGL 構造化レコード・パーツに変換します。

表 81. レコード - 一般構文、レコード・タイプ、プロパティ、およびプロローグ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VAGen レコード・タイプ:</p> <ul style="list-style-type: none"> recordName 基本情報 <ul style="list-style-type: none"> レコード・タイプ レコード構造 (項目リスト) プロパティ (レコード・タイプによって異なる) プロローグ <p>注: 代替仕様レコードを指定するか、項目リストを組み込むことによって、レコード構造を指定できます。</p>	<p>EGL レコードの例:</p> <pre> /** Record=recordName** // prolog //***** Record recordName type recordType { [recordProperties] } recordStructure end // end recordName </pre> <p>注: embed キーワードを指定するか、項目リストを組み込むことによって、レコード構造を指定できます。</p>	<p>特別な考慮事項なし。</p>
<p>レコード・タイプ:</p> <ul style="list-style-type: none"> 作業用ストレージ 再定義 シリアル 索引付き 相対 メッセージ・キュー SQL 行 ユーザー・インターフェース DL/I セグメント 	<p>EGL レコード・タイプ:</p> <ul style="list-style-type: none"> BasicRecord BasicRecord SerialRecord IndexedRecord RelativeRecord MQRecord SQLRecord VGUIRecord DLISegment 	<p>マイグレーション・ツールは、再定義レコードを BasicRecord にマイグレーションします。ツールは、レコード定義にコメントを組み込んで、再定義されたレコードの名前を示します。再定義レコードに関しては、特別な考慮事項が適用されます。詳細と起こりうる問題については、86 ページの『再定義レコード』を参照してください。</p>
<p>作業用ストレージ・レコードのプロパティ:</p> <ul style="list-style-type: none"> 代替仕様 	<p>BasicRecord プロパティ:</p> <ul style="list-style-type: none"> embed キーワード 	<p>マイグレーション・ツールは、代替仕様を embed キーワードにマイグレーションします。</p>
<p>再定義レコードのプロパティ:</p> <ul style="list-style-type: none"> 再定義 <p>注: 再定義プロパティは、物理ストレージを提供する別のレコードの名前を指定します。現行レコードは、同じ物理ストレージに対して異なるデータ項目レイアウトを提供します。</p>	<p>BasicRecord プロパティ:</p> <ul style="list-style-type: none"> 適用不可再定義情報は、レコードを使用するプログラム内でのみ指定されます。同じレコードを別のレコードの再定義として使用することも、通常のレコードとして使用することもできます。 	<p>マイグレーション・ツールは、レコード定義にコメントを組み込んで、再定義されたレコードの名前を示します。</p> <p>マイグレーション・ツールはまた、レコードを使用するプログラム内で、そのレコードに関する declaration ステートメントに redefines プロパティを組み込みます。</p> <p>レコードがプログラム内でどのように使用されるか、およびマイグレーション時にレコードが使用可能かどうかによって、特別な考慮事項が適用されます。詳細と起こりうる問題については、86 ページの『再定義レコード』を参照してください。</p>

表 81. レコード - 一般構文、レコード・タイプ、プロパティ、およびプロローグ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
シリアル・レコードのプロパティ: <ul style="list-style-type: none"> ファイル名 代替仕様 可変長項目 出現箇所項目 	serialRecord プロパティ: <ul style="list-style-type: none"> fileName embed キーワード lengthItem numElementsItem 	マイグレーション・ツールは、代替仕様を embed キーワードにマイグレーションします。
索引付きレコードのプロパティ: <ul style="list-style-type: none"> ファイル名 レコード ID 代替仕様 可変長項目 出現箇所項目 	indexedRecord プロパティ: <ul style="list-style-type: none"> fileName keyItem embed キーワード lengthItem numElementsItem 	マイグレーション・ツールは、代替仕様を embed キーワードにマイグレーションします。
相対レコードのプロパティ: <ul style="list-style-type: none"> ファイル名 レコード ID 代替仕様 	relativeRecord プロパティ: <ul style="list-style-type: none"> fileName recordNumItem embed キーワード 	マイグレーション・ツールは、代替仕様を embed キーワードにマイグレーションします。
メッセージ・キュー・レコードのプロパティ: <ul style="list-style-type: none"> ファイル名 代替仕様 トランザクションにメッセージを組み込む 入力時にキューを排他使用で開く レコード長項目 出現箇所項目 キュー記述子レコード open オプション・レコード メッセージ記述子レコード get オプション・レコード put オプション・レコード 	mqRecord プロパティ: <ul style="list-style-type: none"> queueName embed キーワード includeMsgInTransaction openQueueExclusive lengthItem numElementsItem queueDescriptorRecord openOptionsRecord msgDescriptorRecord getOptionsRecord putOptionsRecord 	マイグレーション・ツールは、代替仕様を embed キーワードにマイグレーションします。
SQL 行レコードのプロパティ: <ul style="list-style-type: none"> 318 ページの表 83 を参照。 	SQL 行レコードのプロパティ: <ul style="list-style-type: none"> 318 ページの表 83 を参照。 	特別な考慮事項なし。
DL/I セグメント・レコード・プロパティ: <ul style="list-style-type: none"> 322 ページの表 84 を参照してください。 	DL/I セグメント・レコード・プロパティ: <ul style="list-style-type: none"> 322 ページの表 84 を参照してください。 	特別な考慮事項なし。
UI レコード・プロパティ: <ul style="list-style-type: none"> 326 ページの表 85 を参照してください。 	UI レコード・プロパティ: <ul style="list-style-type: none"> 326 ページの表 85 を参照してください。 	特別な考慮事項なし。

表 81. レコード - 一般構文、レコード・タイプ、プロパティ、およびプロローグ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
プロローグ	適用不可	マイグレーション・ツールは、プロローグをコメントに変換し、レコード定義の前に配置します。

表 82. レコード - ほとんどのレコード・タイプのレコード構造

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>レコード構造 - バリエーション 1: 代替仕様。RecordA が代替仕様 RecordB を指定している場合、RecordB は RecordA の項目をすべて提供します。RecordA 内に項目構造はありません。</p> <p>RecordB にレベル 77 項目がある場合、RecordA には RecordB のレベル 77 以外の項目のみがあります。</p>	<p>レコード構造 - バリエーション 1: EGL embed キーワードは、現行レコードのフィールド構造を提供するレコードを指定します。RecordA は RecordB を組み込みます。例えば、次のようになります。</p> <p>embed RecordB;</p>	<p>マイグレーション・ツールは、代替仕様を embed キーワードにマイグレーションします。</p> <p>作業用ストレージ・レコード内のレベル 77 項目には、特別な考慮事項が適用されます。詳細と起こりうる問題については、87 ページの『レコード内のレベル 77 項目』を参照してください。</p>

表 82. レコード - ほとんどのレコード・タイプのレコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>レコード構造 - バリエーション 2 (共用項目を指定):</p> <ul style="list-style-type: none"> • itemName • Occurs • 共用 • levelNumber は非表示ですが、レコード内のデータ項目階層に基づいています。 <p>注: 型、長さ、小数部、および説明は、レコード・エディターに表示されますが、レコードに格納されません。</p>	<p>レコード構造 - EGL 型定義を使用したバリエーション 2 の例:</p> <pre>levelNumber itemName itemName [occurs];</pre> <p>注: 型、長さ、小数部、および説明は、エディターには表示されません。</p>	<p>マイグレーション構文設定「共用データ項目をプリミティブ項目定義に変換」を選択した場合に、データ項目パーツが使用可能であれば、共用項目はマイグレーション・ツールによって EGL 変数に変換されます。この変数はデータ項目パーツに対して指定された型、長さ、および 10 進数に基づいたプリミティブ型定義を使用して定義されます。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。</p> <p>マイグレーション構文設定「選択済み共用データ項目をプリミティブ項目定義に変換」を選択解除した場合、またはデータ項目パーツが使用不可の場合は、共用項目はマイグレーション・ツールによって EGL 変数に変換されます。この変数は型定義を使用して定義されます。マイグレーション用の型定義は、常に項目名と同じです。</p> <p>occurs が 1 の場合、マイグレーション・ツールは occurs 情報を省略します。</p> <p>作業用ストレージ・レコード内のレベル 77 項目には、特別な考慮事項が適用されます。詳細と起こりうる問題については、87 ページの『レコード内のレベル 77 項目』を参照してください。</p>

表 82. レコード - ほとんどのレコード・タイプのレコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>レコード構造 - バリエーション 2 (非共用項目を指定):</p> <ul style="list-style-type: none"> • itemName • Occurs • 型 • 長さ • 小数部 • 非共用 • 説明 • levelNumber は隠されていますが、レコード内のデータ項目階層を基にしています。 <p>注: 型、長さ、小数部、および説明は、項目とともにレコードに格納されます。</p>	<p>レコード構造 - EGL プリミティブ型を使用したバリエーション 2 の例:</p> <pre>levelNumber itemName dataType(lengthInformation) [occurs]; // Description</pre> <p>注: 型、長さ、小数部、および説明は、エディターに表示されます。</p>	<p>マイグレーション・ツールは非共用項目を、プリミティブ型を使用して定義される EGL 変数に変換します。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。</p> <p>occurs が 1 の場合、マイグレーション・ツールは occurs 情報を省略します。</p> <p>作業用ストレージ・レコード内のレベル 77 項目には、特別な考慮事項が適用されます。詳細と起こりうる問題については、87 ページの『レコード内のレベル 77 項目』を参照してください。</p>

表 83. レコード - SQL プロパティと SQL レコード構造

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>SQL レコード・プロパティ:</p> <ul style="list-style-type: none"> • デフォルト・キー項目 • 代替仕様 • SQL 表: <ul style="list-style-type: none"> – ラベル – 名前 <p>注:</p> <ul style="list-style-type: none"> • レコードが代替仕様を指定しない場合、キー項目はレコード構造内で key=yes を指定している項目です。デフォルトのキー項目は無視されます。 • レコードが代替仕様を指定している場合、キー項目は、現行レコード内のデフォルトのキー項目と、代替仕様レコード内で key=yes を指定している項目をマージしたものです。キーは、レコード構造内での項目の出現順にマージされます。現行レコード内のデフォルトのキー項目が、代替仕様レコード内でも key=yes として指定されている場合、その項目はマージされたキーのリストに 1 回だけ組み込まれます。 • SQL 表名は、実際の表名 (通常の場合)、または実行時に置換される表名ホスト変数のどちらかです。表名ホスト変数はセミコロン (;) から始まります。 	<p>SQLRecord プロパティ:</p> <ul style="list-style-type: none"> • keyItems • embed キーワード • tableNames と tableNameVariables のいずれかまたは両方 <p>注:</p> <ul style="list-style-type: none"> • keyItems プロパティは、レコード内のキーすべてのリストです。レコード構造内の項目に対して、key=yes は指定されません。 • 表名がホスト変数でない場合は、tableNames プロパティが表名と表ラベルのリストです。表名が実行時に置換されるホスト変数である場合は、tableNameVariables プロパティは表名と表ラベルのリストです。tableNameVariables プロパティ内の表名は、セミコロンから始まりません。tableNames および tableNameVariables プロパティは、両方とも同じレコード定義の中で使用できます。 	<p>マイグレーション・ツールは、keyItems プロパティを以下のように作成します。</p> <ul style="list-style-type: none"> • VAGen 代替仕様が指定されていない場合、ツールはレコード構造にある key=yes を指定している項目すべてを使用しますが、VAGen のデフォルトのキー項目はインクルードしません。 • VAGen 代替仕様が指定されている場合、ツールは代替仕様レコードにある key=yes を指定している項目と、現行レコードのデフォルトのキー項目をマージします。キーは、レコード構造内での項目の出現順と同じ順序でリストされます。現行レコードのデフォルトのキー項目が、代替仕様レコードにあるキー項目のいずれかと同じである場合、その項目は keyItems プロパティに 1 回のみ組み込まれます。 <p>マイグレーション・ツールは、tableNames プロパティと tableNameVariables プロパティのリストを以下の方法で作成します。</p> <ul style="list-style-type: none"> • 表名がホスト変数でない場合は、tableNames が表名と表ラベルから作成されます。 • 表名がホスト変数である場合は、tableNameVariables が表名と表ラベルから作成されます。 <p>特別な考慮事項が適用されます。詳細と起こりうる問題については、89 ページの『代替仕様レコード』にある、SQL 代替仕様に関する情報を参照してください。</p>

表 83. レコード - SQL プロパティと SQL レコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>SQL デフォルト条件:</p> <ul style="list-style-type: none"> • whereClauseText <p>注:</p> <ul style="list-style-type: none"> • SQL デフォルト条件では WHERE 文節を指定できます (SQL 行レコード内で複数の表が使用されているときに結合条件で指定する場合など)。構文は SQL 構文です。 • !itemColumnName 変数が使用できます。この変数は、SQL 行レコード内の項目の名前を指定します。テストまたは生成時に、VisualAge Generator は対応する SQL 列名への置換を行います。 	<p>デフォルト選択条件の例</p> <pre>defaultSelectCondition = #sqlCondition{ whereClauseText }</pre> <p>注:</p> <ul style="list-style-type: none"> • defaultSelectCondition プロパティは、VisualAge Generator と同じ目的で使用されます。 • !itemColumnName 変数はサポートされません。実際の SQL 列名を使用する必要があります。 	<p>マイグレーション・ツールは、!itemColumnName 変数を対応する SQL 列名に変換します。</p> <p>特別な考慮事項が適用されます。詳細と起こりうる問題については、89 ページの『代替仕様レコード』にある、SQL 代替仕様に関する情報を参照してください。</p>
<p>レコード構造 - バリエーション 1: 代替仕様。RecordA が代替仕様 RecordB を指定している場合、RecordB は RecordA の項目をすべて提供します。RecordA 内に項目構造はありません。</p>	<p>レコード構造 - バリエーション 1: EGL embed キーワードは、現行レコードのフィールド構造を提供するレコードを指定します。RecordA は RecordB を組み込みます。例えば、次のようになります。</p> <pre>embed RecordB;</pre>	<p>マイグレーション・ツールは、代替仕様を embed キーワードにマイグレーションします。</p>

表 83. レコード - SQL プロパティと SQL レコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>レコード構造 - バリエーション 2 (共用項目を指定):</p> <ul style="list-style-type: none"> • itemName • 読み取り専用 • キー • SQL 列名 • SQL コード • 共用 <p>注:</p> <ul style="list-style-type: none"> • 型、長さ、小数部、および説明は、レコード・エディターに表示されますが、レコードに格納されません。 • レベル番号は、SQL レコード内では使用されません。 • バック・フィールドとバイナリー・フィールドの外部ソース形式には、SQL コードは組み込まれません。char、dbchar、または unicode のフィールドの外部ソース形式に SQL コードが組み込まれていない場合、フィールドは固定長フィールドとして処理されます。この状態は、VisualAge Generator の旧リリースからマイグレーションされたレコードが、VisualAge Generator 4.5 を使用して変更されなかった場合に起こります。 	<p>レコード構造 - EGL 型定義を使用したバリエーション 2 の例:</p> <pre>levelNumber itemName itemName { [sqlDataCode=sqlCodeNumber] [column="SQLColumnName"] [isReadOnly=yes] [isSQLNullable = yes] [sqlVariableLen = yes] };</pre> <p>注:</p> <ul style="list-style-type: none"> • 型、長さ、小数部、および説明は、エディターには表示されません。 • レベル番号は、SQL レコード内ではオプションです。レベル番号が指定されている場合、SQL レコードは構造化レコードです。レベル番号が指定されていない場合、SQL レコードは構造化レコードではありません。非構造化レコードでは、EGL データ型 BLOB、CLOB、および STRING を使用できます。ただし、非構造化レコードのその他の振る舞いは、VAGen の振る舞いと互換性がありません。 	<p>「共用データ項目をプリミティブ項目定義に変換」設定を選択した場合、データ項目が使用可能であれば、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • 共用項目を EGL 変数に変換します。この変数は、306 ページの表 75で説明しているように、データ項目パーツに関して指定された型、長さ、および小数部に基づいたプリミティブ定義を使用して定義されます。 • 16 進項目に関する sqlDataCode プロパティを組み合わせます。 • 項目が可変長であることが VAGen SQL データ・コードによって示されている場合は、CHAR、DBCHAR、または UNICODE のフィールドの sqlVariableLen プロパティを YES に設定します。項目が固定長であることが VAGen SQL データ・コードによって示されている場合、マイグレーション・ツールは sqlVariableLen プロパティを省略します。 <p>「共用データ項目をプリミティブ項目定義に変換」設定を選択解除した場合、またはデータ項目パーツが使用不可の場合は、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> • 型定義を使用して定義された EGL 変数に共用項目を変換します。マイグレーション用の型定義は、常に項目名と同じです。 • 項目が外部ソース形式に含まれていて、VAGen バイナリー・フィールドまたはバック・フィールドに関する値のうちのいずれかでなければ、sqlDataCode プロパティを組み合わせます。 • 項目が可変長であることが VAGen SQL データ・コードによって示されている場合は、sqlVariableLen プロパティを YES に設定します。項目が固定長であることが VAGen SQL データ・コードによって示されている場合、マイグレーション・ツールは sqlVariableLen プロパティを省略します。 <p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> • SQLRecord の EGL keyItems プロパティに、key=yes になっている項目をすべて組み込みます。 • レコードが構造化レコードになるように、SQLRecord 内の項目にレベル番号を常に追加します。これにより、VAGen の振る舞いが保持されます。

表 83. レコード - SQL プロパティと SQL レコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>レコード構造 - バリエーション 2 (非共用項目を指定):</p> <ul style="list-style-type: none"> itemName 型 長さ 小数部 読み取り専用 キー SQL 列名 SQL コード 非共用 説明 <p>注:</p> <ul style="list-style-type: none"> 型、長さ、小数部、および説明は、項目とともにレコードに格納されます。 レベル番号は、SQL レコード内では使用されません。 パック・フィールドとバイナリー・フィールドの外部ソース形式には、SQL コードは組み込まれません。char、dbchar、または unicode のフィールドの外部ソース形式に SQL コードが組み込まれていない場合、フィールドは固定長フィールドとして処理されます。この状態は、VisualAge Generator の旧リリースからマイグレーションされたレコードが、VisualAge Generator 4.5 を使用して変更されなかった場合に起こります。 	<p>レコード構造 - EGL プリミティブ型を使用したバリエーション 2 の例:</p> <pre>levelNumber itemName dataType(lengthInformation) // Description { [sqlDataCode=sqlCodeNumber] [column="SQLColumnName"] [isReadOnly=yes] [isSQLNullable = yes] [sqlVariableLen = yes] };</pre> <p>注:</p> <ul style="list-style-type: none"> 型、長さ、小数部、および説明は、エディターに表示されます。 レベル番号は、SQL レコード内ではオプションです。レベル番号が指定されている場合、SQL レコードは構造化レコードです。レベル番号が指定されていない場合、SQL レコードは構造化レコードではありません。非構造化レコードでは、EGL データ型 BLOB、CLOB、および STRING を使用できます。ただし、非構造化レコードのその他の振る舞いは、VAGen の振る舞いと互換性がありません。 	<p>マイグレーション・ツールは非共用項目を、プリミティブ型を使用して定義される EGL 変数に変換します。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同様です。</p> <p>マイグレーション・ツールは、16 進項目に関する sqlDataCode プロパティのみを組み込みます。</p> <p>項目が可変長であることが VAGen SQL データ・コードによって示されている場合、マイグレーション・ツールは CHAR、DBCHAR、および UNICODE データ項目の sqlVariableLen プロパティを YES に設定します。項目が固定長であることが VAGen SQL データ・コードによって示されている場合、マイグレーション・ツールは sqlVariableLen プロパティを省略します。</p> <p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> SQLRecord の EGL keyItems プロパティに、key=yes になっている項目をすべて組み込みます。 レコードが構造化レコードになるように、SQLRecord 内の項目にレベル番号を常に追加します。これにより、VAGen の振る舞いが保持されます。
<p>VAGen データ型 - Char</p> <ul style="list-style-type: none"> データ・コード - 453 データ・コード - 449 または 457 	<p>EGL データ型:</p> <ul style="list-style-type: none"> CHAR (sqlVariableLen は省略) CHAR (sqlVariableLen が YES に設定されているもの) 	特別な考慮事項なし。
<p>VAGen データ型 - DBCS</p> <ul style="list-style-type: none"> データ・コード - 469 データ・コード - 465 または 473 	<p>EGL データ型:</p> <ul style="list-style-type: none"> DBCHAR (sqlVariableLen は省略) DBCHAR (sqlVariableLen が YES に設定されているもの) 	特別な考慮事項なし。
<p>VAGen データ型 - Unicode</p> <ul style="list-style-type: none"> データ・コード - 469 データ・コード - 465 または 473 	<p>EGL データ型:</p> <ul style="list-style-type: none"> UNICODE (sqlVariableLen は省略) UNICODE (sqlVariableLen が YES に設定されているもの) 	特別な考慮事項なし。
<p>SQL 列名</p> <p>注: SQL 列名は必須です。</p>	<p>column = "SQLColumnName"</p> <p>注: column プロパティはオプションです。column プロパティを省略した場合は、デフォルトでフィールド名に設定されます。</p>	<p>マイグレーション・ツールは、「列名を省略」の設定に基づいてマイグレーションを行います。</p> <ul style="list-style-type: none"> この設定を選択した場合、ツールは次の処理を行います。 <ul style="list-style-type: none"> SQL 列名がフィールド名と同じであれば、column プロパティを省略します。 SQL 列名がフィールド名と異なっていれば、column プロパティを組み込みます。 この設定を選択解除した場合、ツールは column プロパティを常に組み込みます。

表 83. レコード - SQL プロパティと SQL レコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>対応するプロパティはありません。</p> <p>注: VisualAge Generator は、SQL 項目に常に NULL 標識変数を組み込みます。</p>	<p>isSQLNullable = yes no</p> <p>注: isSQLNullable のデフォルトは no です。</p>	<p>マイグレーション・ツールは、「isSQLNullable プロパティを省略」の設定に基づいてマイグレーションを行います。</p> <ul style="list-style-type: none"> この設定を選択した場合、ツールは isSQLNullable プロパティを組み込まず、このプロパティはデフォルトで no に設定されます。 この設定を選択解除した場合、ツールは isSQLNullable = yes プロパティを常に組み込みます。これにより、VAGen の振る舞いが保持されます。
<p>読み取り専用</p> <p>注: 「読み取り専用」は常に明示的に設定されます。SQL レコードに対して複数の表が指定されている場合、「読み取り専用」は常に yes にする必要があります。</p>	<p>isReadOnly = YES NO</p> <p>注: SQL レコードに対して指定されている表が複数ある場合、isReadOnly はデフォルトで YES に設定されます。SQL レコードに対して指定されている表が 1 つのみの場合、isReadOnly はデフォルトで NO に設定されます。</p>	<p>マイグレーション・ツールは、「isReadOnly プロパティを省略」の設定に基づいてマイグレーションを行います。</p> <ul style="list-style-type: none"> この設定を選択した場合は、レコードに関して単一の表が指定されていて、VAGen の「読み取り専用」プロパティが yes に設定されている場合に限り、ツールは isReadOnly プロパティを組み込みます。 この設定を選択解除した場合は、VAGen の「読み取り専用」プロパティが yes に設定されているれば、ツールは isReadOnly プロパティを組み込みます。

表 84. レコード - DL/I プロパティと DL/I レコード構造

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>DL/I レコード・プロパティ</p> <ul style="list-style-type: none"> キー項目 代替仕様 レコード長項目 <p>注: レコード名は DL/I PSB 内のセグメント名と同じでなければなりません。</p>	<p>DLISegment プロパティ:</p> <ul style="list-style-type: none"> keyItem embed キーワード lengthItem <p>注: EGL では DL/I PSB のセグメント名と異なるレコード名が許されます。この状態で、EGL segmentName プロパティは DL/I PSB で使用される名前を提供します。</p>	<p>マイグレーション・ツールは、EGL 予約語との競合、またはレコード名の先頭が # または @ シンボルであることが原因でレコード名を変更する場合、オリジナルのレコード名を提供するために、segmentName プロパティをインクルードします。</p>

表 84. レコード - *DL/I* プロパティと *DL/I* レコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>レコード構造 - バリエーション 1: 代替仕様。RecordA が代替仕様 RecordB を提供している場合、RecordB は RecordA の項目をすべて提供します。RecordA 内に項目構造はありません。</p> <p>注: 代替仕様のフィールド名は、DL/I PSB のフィールド名と同じでなければなりません。</p>	<p>レコード構造 - バリエーション 1: EGL embed キーワードは、現行レコードのフィールド構造を提供するレコードを指定します。RecordA は RecordB を組み込みます。例えば、次のようになります。</p> <p>embed RecordB;</p> <p>EGL では、レコードのフィールド名に、DL/I PSB のフィールド名と異なる名前を使用できます。組み込まれたレコードが DLISegment ではなく、フィールド名が異なる場合、dliFieldName プロパティを次のように設定してください。</p> <pre>embed RecordB { fieldInRecordB {dliFieldName="nameInPSB"} } ;</pre>	<p>マイグレーション・ツールは、代替仕様を embed キーワードにマイグレーションします。</p> <p>代替仕様レコードが DL/I セグメントでない場合、ツールは、EGL 予約語との競合、またはフィールド名の先頭が # または @ シンボルであることが原因で名前変更された項目の dliFieldName プロパティを指定変更します。</p> <p>特別な考慮事項が適用されます。詳しくは、89 ページの『代替仕様レコード』を参照してください。</p>

表 84. レコード - *DL/I* プロパティと *DL/I* レコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>レコード構造 - バリエーション 2 (共用項目を指定):</p> <ul style="list-style-type: none"> • itemName • Occurs • 共用 • levelNumber は隠されていますが、レコード内のデータ項目階層を基にしています。 <p>注:</p> <ul style="list-style-type: none"> • 型、長さ、小数部、および説明は、レコード・エディターに表示されますが、レコードに格納されません。 • フィールド名は、DL/I PSB のフィールド名と同じでなければなりません。 	<p>レコード構造 - EGL 型定義を使用したバリエーション 2 の例:</p> <pre>levelNumber itemName itemName [occurs] {dliFieldName="nameInPSB"};</pre> <p>注:</p> <ul style="list-style-type: none"> • 型、長さ、小数部、および説明は、エディターには表示されません。 • EGL では、レコードのフィールド名に、DL/I PSB のフィールド名と異なる名前を使用できます。 	<p>マイグレーション構文設定「共用データ項目をプリミティブ項目定義に変換」を選択した場合に、データ項目パーツが使用可能であれば、共用項目はマイグレーション・ツールによって EGL 変数に変換されます。この変数はデータ項目パーツに対して指定された型、長さ、および 10 進数に基づいたプリミティブ型定義を使用して定義されます。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。</p> <p>マイグレーション構文設定「選択済み共用データ項目をプリミティブ項目定義に変換」を選択解除した場合、またはデータ項目パーツが使用不可の場合は、共用項目はマイグレーション・ツールによって EGL 変数に変換されます。この変数は型定義を使用して定義されます。マイグレーション用の型定義は、常に項目名と同じです。</p> <p>occurs が 1 の場合、マイグレーション・ツールは occurs 情報を省略します。</p> <p>EGL 予約語との競合、またはレコード名の先頭が # または @ シンボルであることが原因で、項目が名前変更される場合、マイグレーション・ツールは dliFieldName プロパティをインクルードします。</p>

表 84. レコード - *DL/I* プロパティと *DL/I* レコード構造 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>レコード構造 - バリエーション 2 (非共用項目を指定):</p> <ul style="list-style-type: none"> • <code>itemName</code> • <code>Occurs</code> • 型 • 長さ • 小数部 • 非共用 • 説明 <p><code>levelNumber</code> は隠されていますが、レコード内のデータ項目階層を基にしています。</p> <p>注:</p> <ul style="list-style-type: none"> • 型、長さ、10 進数および記述は、レコードに保管されます。 • フィールド名は、<i>DL/I PSB</i> のフィールド名と同じでなければなりません。 	<p>レコード構造 - EGL プリミティブ型を使用したバリエーション 2 の例:</p> <pre>levelNumber itemName dataType(lengthInformation) [occurs] {dliFieldName="nameInPSB"}; // Description</pre> <p>注:</p> <ul style="list-style-type: none"> • 型、長さ、小数部、および説明は、エディターに表示されます。 • EGL では、レコードのフィールド名に、<i>DL/I PSB</i> のフィールド名と異なる名前を使用できます。 	<p>マイグレーション・ツールは非共用項目を、プリミティブ型を使用して定義される EGL 変数に変換します。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。</p> <p><code>occurs</code> が 1 の場合、マイグレーション・ツールは <code>occurs</code> 情報を省略します。</p> <p>EGL 予約語との競合、またはレコード名の先頭が # または @ シンボルであることが原因で、項目が名前変更される場合、マイグレーション・ツールは dliFieldName プロパティをインクルードします。</p>

表 85. レコード - UI レコード・プロパティとレコード構造

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
UI レコード・プロパティ: <ul style="list-style-type: none"> 一般 <ul style="list-style-type: none"> UI タイトル 値項目の実行依頼 編集関数 Web での編集関数の実行 編集順序の入力 ヘルプ・テキスト 	VGUI レコード・プロパティ: <ul style="list-style-type: none"> 一般プロパティ <ul style="list-style-type: none"> title commandValueItem validatorFunction runValidatorFromProgram validationOrder help <p>注: validationOrder プロパティは、レコード内の各項目で指定されます。</p> <p>VGUI レコード定義を、次の例に示します。</p> <pre>Record recordName type VGUIRecord {throwNrfEofExceptions = yes, handleHardIOErrors = no, V60ExceptionCompatibility = yes, I4GLItemsNullable = no, textLiteralDefaultIsString = no, localSQLScope = yes, alias="originalVAGenName", commandValueItem=itemX, validatorFunction=functionY, runValidatorFromProgram=yes, title="Page Title", help="help text line" } recordStructure end // end recordName</pre>	EGL runValidatorFromProgram プロパティは、VAGen プロパティの逆です。マイグレーション・ツールは yes を NO に、no を YES に変換します。 <p>マイグレーション・ツールは、VAGen の振る舞いを保持するために常に次のプロパティを組み込みます。</p> <ul style="list-style-type: none"> throwNrfEofExceptions handleHardIOErrors V60ExceptionCompatibility I4GLItemsNullable textLiteralDefaultIsString localSQLScope
適用不可	alias	レコード名が EGL 予約語と競合するか、レコード名の先頭が # または @ シンボルである場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> UI レコードを名前変更する。 alias プロパティをオリジナルの VAGen UI レコード名にセットする。 <p>特別な考慮事項が適用されます。詳しくは、92 ページの『予約語と UI レコード名』を参照してください。</p>
レコード構造は、それぞれのデータ項目ごとに追加情報がある点を除き、作業用ストレージ・レコードの構造に似ています。 <ul style="list-style-type: none"> UI タイプ UI プロパティ 	レコード構造は、それぞれのフィールドごとに追加プロパティがある点を除き、基本レコードの構造に似ています。 <ul style="list-style-type: none"> uiType 検証プロパティ、フォーマット・プロパティ、および JSFHandler フィールド・プロパティ 	特別な考慮事項なし。

表 86. レコード - UI 項目プロパティ - 一般

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
適用不可	alias	<p>フィールド名がマイグレーション・ツールの拡張予約語リストと競合するか、フィールド名の先頭が # または @ シンボルである場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> フィールドを名前変更する。 alias プロパティをオリジナルのフィールド名にセットする。 <p>特別な考慮事項が適用されます。詳しくは、92 ページの『予約語と UI レコード名』を参照してください。</p>
UI タイプ - 値は次のとおり: <ul style="list-style-type: none"> Form Hidden Input Input/Output None Output Program Link Submit Submit Bypass 	uiType - 値は次のとおりです。 <ul style="list-style-type: none"> uiForm hidden input inputOutput none output programLink submit submitBypass 	特別な考慮事項なし。
UI ラベル	displayName (JSFHandler フィールド・プロパティ)	特別な考慮事項なし。
配列項目: <ul style="list-style-type: none"> 出現箇所項目 選択済み索引項目 	構造化フィールド配列: <ul style="list-style-type: none"> numElementsItem selectedIndexItem 	特別な考慮事項なし。

表 87. レコード - UI 項目プロパティ - 編集

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
Edit タイプ - 値は次のとおり: <ul style="list-style-type: none"> None Boolean Date Time 	EGL は、複数のプロパティをサポートします。 <ul style="list-style-type: none"> 適用不可 isBoolean = yes dateFormat = defaultDateFormat timeFormat = "HH:mm:ss" (フォーマット設定プロパティ)	特別な考慮事項なし。
編集関数	validatorFunction (検証プロパティ)	特別な考慮事項なし。

表 87. レコード - UI 項目プロパティ - 編集 (続き)

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
Web での編集関数の実行	runValidatorFromProgram (検証プロパティ)	EGL プロパティは、VAGen プロパティの逆です。マイグレーション・ツールは yes を NO に、no を YES に変換します。
編集テーブル	validatorDataTable (検証プロパティ)	特別な考慮事項なし。
最小入力	minimumInput (検証プロパティ)	特別な考慮事項なし。
充てん文字 注: <ul style="list-style-type: none"> 以下のデフォルトの充てん文字が使用されます。 <ul style="list-style-type: none"> 文字、混合、および数値項目の場合のブランク。 16 進項目の場合の 0。 DBCS およびユニコードの項目の充てん文字はブランクでなければなりません。 NULL は無効な充てん文字です。 	fillCharacter (フォーマット設定プロパティ) 注: <ul style="list-style-type: none"> 以下のデフォルトの充てん文字が使用されます。 <ul style="list-style-type: none"> 文字、MBCHAR、および数値項目の場合のブランク。 16 進項目の場合の 0。 DBCHAR の場合の充てん文字はブランクでなければなりません。UNICODE 用充てん文字としては、どの文字も有効です。 NULL は無効な充てん文字です。 	特別な考慮事項なし。
大文字変換	upperCase (フォーマット設定プロパティ)	特別な考慮事項なし。
入力必須	inputRequired (検証プロパティ)	特別な考慮事項なし。
SO/SI スペースの検査	needsSOSI (検証プロパティ)	特別な考慮事項なし。
通貨と通貨記号	currency = YES NO currencySymbol = "symbol"	特別な考慮事項なし。
最小値と最大値 注: 最小値または最大値のどちらかを指定する場合は、両方を指定する必要があります。	validValues = [[minimumValue, maximumValue]] (検証プロパティ)	マイグレーション・ツールは、最小値と最大値を結合して EGL の validValues プロパティを指定します。

表 87. レコード - UI 項目プロパティ - 編集 (続き)

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
符号 - なし 先頭 末尾 注: UI レコード内の数値項目に対するデフォルト記号は leading です。	sign = none leading trailing (フォーマット設定プロパティ) 注: 数値フィールドのデフォルトの sign は常に leading です。	<ul style="list-style-type: none"> UI タイプが非表示、入力、出力、または入出力の数値フィールドの場合、マイグレーション・ツールは次のうち最初に該当する基準に基づいて記号を変換します。 <ul style="list-style-type: none"> UI 記号編集が指定されている場合、ツールは対応する sign 記号プロパティにマイグレーションします。 UI 編集タイプとして日付、時刻、またはブール値が指定されている場合、ツールは sign プロパティを none に設定します。 ツールは、sign プロパティをデフォルト値の leading に設定します。 その他の UI タイプを持つ数値フィールドの場合、マイグレーション・ツールは sign プロパティを省略します。
セパレーター	numericSeparator (フォーマット設定プロパティ)	特別な考慮事項なし。
ゼロ編集	zeroFormat (フォーマット設定プロパティ)	特別な考慮事項なし。

表 88. レコード - UI 項目プロパティ - エラー・メッセージ

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
編集テーブル	validatorTableMsgKey (検証プロパティ)	特別な考慮事項なし。
EZE 関数	validatorFunctionMsgKey (検証プロパティ)	特別な考慮事項なし。
最小入力	minimumInputMsgKey (検証プロパティ)	特別な考慮事項なし。
入力必須	inputRequiredMsgKey (検証プロパティ)	特別な考慮事項なし。
データ型	typeChkMsgKey (検証プロパティ)	特別な考慮事項なし。
数値範囲	validValuesMsgKey (検証プロパティ)	特別な考慮事項なし。

表 89. UI 項目プロパティ - ヘルプ

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
ヘルプ・テキスト	help (JSFHandler フィールド・プロパティ)	特別な考慮事項なし。

表 90. UI 項目プロパティ - 実行依頼

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
初期値	<p>配列でないフィールドに関する次のフォーマットでのイニシャライザー。</p> <pre>= "initialValue"</pre> <p>構造化フィールド配列に関する次のフォーマットでのイニシャライザー。</p> <pre>= ["initialValue1", "initialValue2"]</pre>	特別な考慮事項なし。

表 91. レコード - UI 項目プロパティ - プログラム・リンク

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
<p>プログラム・リンク情報</p> <ul style="list-style-type: none"> • Program • 先頭 UI レコード • 新規ウィンドウとして開く • パラメーターのリンク 	<p>プログラム・リンク情報</p> <ul style="list-style-type: none"> • programName • uiRecordName • newWindow • linkParms <p>EGL は、次のように VAGen プログラムのリンク・プロパティを結合して、複素数プロパティにします。</p> <pre>@programLinkData { programName = "PRGA", uiRecordName = "MYUI", newWindow = yes [, linkParmsInfo] }</pre> <p>注: オプションの <i>linkParmsInfo</i> についての詳細は、表の次の行を参照してください。</p>	特別な考慮事項なし。

表 91. レコード - UI 項目プロパティ - プログラム・リンク (続き)

VisualAge Generator 4.5	EGL	マイグレーション・ツールに関する考慮事項
<p>パラメーターのリンク:</p> <ul style="list-style-type: none"> 先頭 UI レコードの項目 値 <ul style="list-style-type: none"> リテラル 現行レコードの項目 <p>注:</p> <ul style="list-style-type: none"> フォーム UI タイプ用のプログラム・リンク・カスタマイズを使用しているときには、現行フォーム内のデータは、次のプログラムの最初の UI レコードに名前ごとに自動的に移動させられます。先頭 UI レコード内の追加フィールドは、リンク・パラメーター内でリスト作成することで初期化することができます。 プログラム・リンク UI タイプに関してプログラム・リンク・カスタマイズを使用しているときには、リンク・パラメーターに明示的にリストされた先頭 UI レコード内のフィールドのみが初期化されます。 	<p>EGL は VAGen リンク・パラメーターを結合して複素数プロパティにします。</p> <pre>linkParms = [@LinkParameter { name = "item1InFirstUI", value = "literal" }, @LinkParameter { name = "item2InFirstUI", valueRef = itemInCurrentUI}]</pre> <p>注: uiForm と programLink UI の両方のタイプに関してプログラム・リンク・カスタマイズを行う場合、EGL は VisualAge Generator と同じ規則に従います。</p>	<p>特別な考慮事項なし。</p>

テーブル

VAGen テーブルに関するセクションは、次の表で構成されています。

- テーブル - 一般構文、テーブル・タイプ、プロパティ、およびプロローグ (332 ページの表 92)
- VAGen テーブル - テーブル構造 (333 ページの表 93)
- VAGen テーブル - テーブルの内容 (334 ページの表 94)

表 92. テーブル - 一般構文、テーブル・タイプ、プロパティ、およびプロローグ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VAGen テーブル・パーツ:</p> <ul style="list-style-type: none"> • テーブル名 • 基本情報 <ul style="list-style-type: none"> - テーブル・タイプ - テーブル構造 (項目リスト) • プロパティ • プロローグ • テーブルの内容 <p>注: テーブルの内容はオプションです。</p>	<p>EGL 構文の例:</p> <pre> /***/ DataTable=tableName***/ // prolog //***** DataTable tableName type tableType { [otherTableProperties] [alias = "originalTableName"] } tableStructure [{ contents = [{rowContents}] }] end // end tableName </pre> <p>注: contents プロパティが必要です。</p>	<p>名前が EGL 予約語リストと競合していても、マイグレーション・ツールはテーブルの名前を変更しません。マイグレーション・ツールは alias プロパティを設定しません。テーブルの名前を変更する必要がある場合は、alias プロパティを使用して、VAGen テーブルのオリジナルの名前を指定してください。詳細については、93 ページの『予約語とテーブル名』のテーブル名に関する情報を参照してください。</p>
<p>テーブル・タイプ:</p> <ul style="list-style-type: none"> • 指定なし • 不適合 • 適合 • 範囲適合 • メッセージ 	<p>DataTable タイプ:</p> <ul style="list-style-type: none"> • basicTable • matchInvalidTable • matchValidTable • rangeChkTable • msgTable 	<p>特別な考慮事項なし。</p>
<p>プロパティ - 実行時属性:</p> <ul style="list-style-type: none"> • 常駐 • 共用 	<p>DataTable プロパティ:</p> <ul style="list-style-type: none"> • resident • shared 	<p>特別な考慮事項なし。</p>
<p>プロパティ - テーブルの内容の大文字変換</p>	<p>適用不可 テーブルの内容を大文字にする必要がある場合は、内容を大文字で入力する必要があります。</p>	<p>VAGen テーブルがテーブルの内容を大文字に変換するように指定している場合、マイグレーション・ツールは、テーブルの内容にある char データ、16 進データ、および混合データが大文字に変換されるようにします。</p>
<p>プロローグ</p>	<p>適用不可</p>	<p>マイグレーション・ツールは、プロローグをコメントに変換し、DataTable 定義の前に配置します。</p>

表 93. テーブル - テーブル構造

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VAGen テーブル構造 - 共用項目を指定:</p> <ul style="list-style-type: none"> • itemName • 共用 • levelNumber は非表示ですが、レコード内のデータ項目階層に基づいています。 <p>注: 型、長さ、小数部、および説明は、テーブル・エディターに表示されますが、テーブルに格納されません。</p>	<p>DataTable 構造 - EGL 型定義を指定:</p> <pre>levelNumber itemName itemName ;</pre> <p>注: 型、長さ、小数部、および説明は、エディターには表示されません。</p>	<p>マイグレーション構文設定「共用データ項目をプリミティブ項目定義に変換」を選択した場合に、データ項目パーツが使用可能であれば、共用項目はマイグレーション・ツールによって EGL 変数に変換されます。この変数はデータ項目パーツに対して指定された型、長さ、および 10 進数に基づいたプリミティブ定義を使用して定義されます。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。</p> <p>マイグレーション構文設定「選択済み共用データ項目をプリミティブ項目定義に変換」を選択解除した場合、またはデータ項目パーツが使用不可の場合は、共用項目はマイグレーション・ツールによって EGL 変数に変換されます。この変数は型定義を使用して定義されます。マイグレーション用の型定義は、常に項目名と同じです。</p>
<p>VAGen テーブル構造 - 非共用項目を指定:</p> <ul style="list-style-type: none"> • itemName • 型 • 長さ • 小数部 • 非共用 • 説明 • levelNumber は非表示ですが、テーブル内のデータ項目階層に基づいています。 <p>注: 型、長さ、小数部、および説明は、項目とともにテーブルに格納されます。</p>	<p>DataTable 構造 - EGL プリミティブ型を指定:</p> <pre>levelNumber itemName dataType(lengthInformation) ; // Description</pre> <p>注: 型、長さ、小数部、および説明は、エディターに表示されます。</p>	<p>マイグレーション・ツールは非共用項目を、プリミティブ型を使用して定義される EGL 変数に変換します。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。</p>

表 94. テーブル - テーブルの内容

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>テーブルの内容:</p> <ul style="list-style-type: none"> • テーブルの内容は、フォーマット設定エディターに入力されます。テーブルの内容は、テーブル構造のトップレベル (親) 項目に対して入力されます。 • 文字データと 16 進データは引用符で囲まれません。 	<p>DataTable の内容:</p> <ul style="list-style-type: none"> • 各行の内容は大括弧で囲まれます。外側に、行のセット全体を囲む大括弧の組があります。 • 行の内容に含まれる値は、コンマで区切る必要があります。 • 文字データ (16 進データを含む) は、二重引用符で囲む必要があります。 <p>例:</p> <pre>contents = [[rowContents] { , [rowContents] }] where rowContents = value { , value }</pre>	<p>VAGen テーブルがテーブルの内容を大文字に変換するように指定している場合、マイグレーション・ツールは、テーブルの内容にある char データ、16 進データ、および混合データが大文字に変換されるようにします。</p> <p>マイグレーション・ツールはまた、文字データ (16 進データを含む) を二重引用符で囲みます。</p>

マップ・グループ

マップ・グループに関するセクションは、次の表で構成されています。

- マップ・グループ - 一般情報 (334 ページの表 95)
- マップ・グループ - 一般構文と浮動域 (336 ページの表 96)
- マップ・グループ - 装置の名前、タイプ、およびサイズ (338 ページの表 97)

表 95. マップ・グループ - 一般情報

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>マップ・グループ・パーツは、浮動域が存在する場合のみ必要です。</p> <p>マップ・グループ・パーツがない場合は、マップ・グループ・パーツが存在する場合と同様に、VisualAge Generator によって同じマップ・グループ名のマップすべてが自動的に生成されます。</p>	<p>FormGroup は必須です。</p>	<p>マイグレーション・ツールは、FormGroup パーツがマイグレーション・セット内に存在しなければ、このパーツを作成します。</p>

表 95. マップ・グループ - 一般情報 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>マップ名は、マップ・グループ名とマップ名からなります。</p>	<p>書式名には、FormGroup 名は含まれません。</p> <p>FormGroup 内で書式を定義 (ネスト) できます。</p> <p>また、書式を FormGroup の外部で定義することもできます。この場合、FormGroup には、フォーム名を指定する use ステートメントと、フォームがあるパッケージをインポートする import ステートメントが含まれている必要があります。この技法により、共通書式 (例えば、ポップアップ・リストの書式) の定義を 1 つ作成すれば、さまざまな FormGroup 内でその書式を使用できます。</p>	<p>マイグレーション・ツールは、マップをすべて書式にマイグレーションします。マイグレーション・ツールは、複数のマップ・グループ間で共通する同一のマップ定義を識別しません。</p> <p>単一ファイル・モードでマイグレーションを行う場合、マイグレーション・ツールは FormGroup 内のそれぞれの書式ごとに use ステートメントを組み込みます。該当する FormGroup 内で書式がネストされるように、書式を移動する必要があります。</p> <p>ステージ 1 から 3 までのマイグレーションを使用する場合、マイグレーション・ツールはすべての書式を FormGroup 内で自動的にネストします。</p>
<p>プログラムがマップ・グループを指定していれば、プログラムはマップ名を参照するだけでマップ・グループ内の任意のマップを使用できます。</p> <p>VisualAge Generator は、プログラム内で XFER ステートメントの CONVERSE、DISPLAY、または CLOSE I/O オプションで明示的に参照されるマップのみを、プログラム・パラメーターとみなし、また、非修飾フィールド名の解決時には最初のマップのみを、プログラム・パラメーターとみなします。</p>	<p>使用する FormGroup を示す use ステートメントがプログラムに組み込まれている場合、プログラムは書式名を参照するのみで FormGroup 内の任意の書式を参照できます。</p> <p>use ステートメントが特定の書式をリストせずに FormGroup を指定している場合、EGL はプログラム内の非修飾フィールド名の解決時に、その FormGroup 内のすべての書式を使用対象として考慮します。</p> <p>use ステートメントで FormGroup 内の書式が指定されている場合、EGL はプログラム内の非修飾フィールド名の解決時に、その FormGroup 内の指定されている書式のみを使用対象として考慮します。</p>	<p>マイグレーション・ツールは use ステートメントを作成して、プログラムで参照される特定の書式のみをリストします。</p>

表 96. マップ・グループ - 一般構文と浮動域

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>マップ・グループ・パーツには、次の情報を含めることができます。</p> <ul style="list-style-type: none"> マップ・グループ名 浮動域情報 <ul style="list-style-type: none"> 装置名 装置サイズ サイズ <ul style="list-style-type: none"> 行 列 位置 <ul style="list-style-type: none"> 開始行 開始列 	<p>FormGroup には、次の情報を含めることができます。</p> <ul style="list-style-type: none"> FormGroup 名 FormGroup プロパティ 画面浮動域の情報 印刷浮動域の情報 FormGroup に組み込まれる書式の use ステートメント <p>以下の例に、FormGroup のフォーマットを示します。</p> <pre> FormGroup groupName { [alias="generationName" [ScreenFloatingAreas = [@ScreenFloatingArea { screenFloatingAreaInfo }] ,] [PrintFloatingAreas = [@PrintFloatingArea { printFloatingAreaInfo }] ,] } Form formName type TextForm {formProperties} [variableFields] [constantFields] end // end formName use formName2; end // end groupName </pre>	<p>マイグレーション・ツールは、VAGen 装置タイプを使用して、浮動域情報の対象が表示マップ (screenFloatingArea) またはプリンター・マップ (printFloatingArea) のどちらであるかを判別します。</p> <p>装置がディスプレイまたはプリンターのどちらであるのかを判別する方法については、338 ページの表 97 を参照してください。</p>
適用不可	alias	<p>マップ・グループ名が EGL 予約語と競合していても、マイグレーション・ツールはマップ・グループの名前を変更しません。特別な考慮事項が適用されます。詳細と起こりうる問題については、94 ページの『予約語と FormGroup 名』を参照してください。</p>

表 96. マップ・グループ - 一般構文と浮動域 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>浮動域情報:</p> <ul style="list-style-type: none"> 装置名 装置サイズ (行 x 列) 浮動域仕様 <ul style="list-style-type: none"> サイズ <ul style="list-style-type: none"> 行 列 位置 <ul style="list-style-type: none"> 開始行 開始列 <p>注:</p> <ul style="list-style-type: none"> VisualAge Generator では、浮動域のサイズと開始位置を定義します。 同じサイズをもつ装置に対して異なる浮動域仕様を指定できます。 	<p>浮動域情報:</p> <ul style="list-style-type: none"> 装置サイズ マージン情報 <p>印刷浮動域の情報にも装置タイプが含まれます。</p> <p>次に、テキスト書式に使用される画面浮動域の例を示します。</p> <pre>ScreenFloatingAreas = [@ScreenFloatingArea { screenSize=[lines,columns], topMargin=nn, bottomMargin=nn, leftMargin=nn, rightMargin=nn }]</pre> <p>次に、印刷書式に使用される印刷浮動域の例を示します。</p> <pre>PrintFloatingAreas = [@PrintFloatingArea { deviceType=singleByte, pageSize=[lines,columns], topMargin=nn, bottomMargin=nn, leftMargin=nn, rightMargin=nn }]</pre> <p>注: screenSize または pageSize に関して指定できる浮動域仕様はただ 1 つです。</p>	<p>マイグレーション・ツールは、VAGen 装置タイプを使用して、浮動域仕様の対象が表示マップ (screenFloatingArea) またはプリンター・マップ (printFloatingArea) のどちらであるかを判別します。</p> <p>マイグレーション・ツールは、マージン情報を次の方法で計算します。</p> <ul style="list-style-type: none"> topMargin は VAGen floatingAreaStartingLine - 1 に設定されます。 bottomMargin は VAGen deviceRows - (floatingAreaStartingLine + floatingAreaLines) + 1 に設定されます。 leftMargin は VAGen floatingAreaStartingColumn - 1 に設定されます。 rightMargin は VAGen deviceColumns - (floatingAreaStartingColumn + floatingAreaColumns) + 1 に設定されます。 <p>装置がディスプレイまたはプリンターのどちらであるのかを判別する方法については、338 ページの表 97 を参照してください。</p>
<p>プリンター・タイプ:</p> <ul style="list-style-type: none"> プリンター DBCS プリンター 	<p>deviceType=singleByte doubleByte</p> <p>注: deviceType プロパティは、印刷書式についてのみ指定できます。</p>	<p>マイグレーション・ツールは、VAGen プリンター・タイプに基づいて EGL deviceType プロパティを設定します。</p> <p>マイグレーション・ツールは、DeviceTypeKind を使用して deviceType 値を常に修飾します (例: deviceType = DeviceTypeKind.doubleByte)。これにより、FormGroup 内の書式にある変数フィールドとの名前の競合が回避されます。</p>

表 97. マップ・グループ - 装置の名前、タイプ、およびサイズ

VisualAge Generator 装置名	装置サイズ (行 x 列)	装置タイプ	マイグレーション・ツールに関する考慮事項
3643-2	6 x 40	ディスプレイ	特別な考慮事項なし。
3277-1	12 x 40	ディスプレイ	特別な考慮事項なし。
3643-4	16 x 64	ディスプレイ	特別な考慮事項なし。
3278-1、3278-1B、 ANY-1D	12 x 80	ディスプレイ	特別な考慮事項なし。
3278-2、3278-2B、 ANY-2D	24 x 80	ディスプレイ	特別な考慮事項なし。
3278-3、3278-3B、 ANY-3D	32 x 80	ディスプレイ	特別な考慮事項なし。
3278-4、3278-4B、 ANY-4D	43 x 80	ディスプレイ	特別な考慮事項なし。
3278-5、3278-5B、 ANY-5D	27 x 132	ディスプレイ	特別な考慮事項なし。
ANY-D (62x160 として 構成された 3290)	255 x 160	ディスプレイ	特別な考慮事項なし。
5550D	24 x 80	DBCS ディスプレ イ	特別な考慮事項なし。
3767、PRINT-B、 PRINTER	255 x 132	プリンター	@printFloatingArea 複合プロパティの場合、EGL deviceType = singleByte
5550P	255 x 158	DBCS プリンター	@printFloatingArea 複合プロパティの場合、EGL deviceType = doubleByte

マップ

マップに関するセクションは、次の表で構成されています。

- マップ - 一般情報 (339 ページの表 98)
- 表示マップ - 一般構文、マップ・タイプ、およびプロパティ (340 ページの表 99)
- プリンター・マップ - 一般構文、マップ・タイプ、およびプロパティ (343 ページの表 100)
- マップの定数フィールドと変数フィールド - 一般情報 (344 ページの表 101)
- マップの定数フィールドと変数フィールド - 一般構文、データ型、長さ、小数部、および説明 (347 ページの表 102)
- マップの定数フィールドと変数フィールド - 属性 (350 ページの表 103)
- マップ変数フィールド - 一般編集 (352 ページの表 104)
- マップ変数フィールド - 数値編集 (354 ページの表 105)
- マップ変数フィールド - エラー・メッセージ (356 ページの表 106)

表 98. マップ - 一般情報

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>マップには次の 2 種類があります。</p> <ul style="list-style-type: none"> 表示マップ プリンター・マップ 	<p>書式には次の 2 種類があります。</p> <ul style="list-style-type: none"> テキスト書式 印刷書式 	<p>特別な考慮事項なし。</p>
<p>マップ名は、マップ・グループ名とマップ名からなります。</p>	<p>書式名には、FormGroup 名は含まれません。</p> <p>FormGroup 内で書式を定義 (ネスト) できます。</p> <p>また、書式を FormGroup の外部で定義することもできます。この場合、FormGroup には、フォーム名を指定する use ステートメントと、フォームがあるパッケージをインポートする import ステートメントが含まれている必要があります。この技法により、共通書式 (例えば、ポップアップ・リストの書式) の定義を 1 つ作成すれば、さまざまな FormGroup 内でその書式を使用できます。</p>	<p>マイグレーション・ツールは、マップをすべて書式にマイグレーションします。マイグレーション・ツールは、複数のマップ・グループ間で共通する同一のマップ定義を識別しません。</p> <p>単一ファイル・モードでマイグレーションを行う場合、マイグレーション・ツールは FormGroup 内のそれぞれの書式ごとに use ステートメントを組み込みます。該当する FormGroup 内で書式がネストされるように、書式を移動する必要があります。</p> <p>ステージ 1 から 3 までのマイグレーションを使用する場合、マイグレーション・ツールはすべての書式を FormGroup 内で自動的にネストします。</p>
<p>プログラムがマップ・グループを指定していれば、プログラムはマップ名を参照するだけでマップ・グループ内の任意のマップを使用できます。</p> <p>VisualAge Generator は、プログラム内で XFER ステートメントの CONVERSE、DISPLAY、または CLOSE I/O オプションで明示的に参照されるマップのみを、プログラム・パラメータとみなし、また、非修飾フィールド名の解決時には最初のマップのみを、プログラム・パラメータとみなします。</p>	<p>使用する FormGroup を示す use ステートメントがプログラムに組み込まれていれば、プログラムは書式名を参照するのみで FormGroup 内の任意のマップを参照できます。</p> <p>use ステートメントが特定の書式をリストせずに FormGroup を指定している場合、EGL はプログラム内の非修飾フィールド名の解決時に、その FormGroup 内のすべての書式を使用対象として考慮します。</p> <p>use ステートメントで FormGroup 内の書式が指定されている場合、EGL はプログラム内の非修飾フィールド名の解決時に、その FormGroup 内の指定されている書式のみを使用対象として考慮します。</p>	<p>マイグレーション・ツールは use ステートメントを作成して、プログラムで参照される特定の書式のみをリストします。</p>

表 99. 表示マップ - 一般構文、マップ・タイプ、およびプロパティ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>表示マップには、次の情報を含めることができます。</p> <ul style="list-style-type: none"> マップ・グループ名とマップ名 マップ・プロパティ <ul style="list-style-type: none"> 一般プロパティ <ul style="list-style-type: none"> ヘルプ・マップ名 ヘルプ・キー バイパス・キー 変数フィールドの大文字への変換 レイアウト・プロパティ <ul style="list-style-type: none"> マップ・サイズ 開始位置 浮動マップ 装置 <ul style="list-style-type: none"> タイプ (ディスプレイまたは印刷) サポートされる装置 定数フィールド 変数フィールド 変数フィールドのフィールド編集順序 	<p>テキスト書式パーツには、次の情報を含めることができます。</p> <ul style="list-style-type: none"> フォーム名 書式タイプ 書式プロパティ 定数フィールド 変数フィールド 変数フィールドの検証順序 <p>マイグレーション・ツールによって作成されるテキスト書式のフォーマットを、次の例に示します。</p> <pre>Form mapName type TextForm { screenSizes=[sizeList], formSize=[24,80], position=[1,1], helpForm="helpFormName", helpKey=pf1, validationBypassKeys=[pf3], msgField="VAGen_EZMSG"} [variableFields] [constantFields] end // end mapName</pre>	<p>マイグレーション・ツールは、VAGen 装置タイプを使用して、マップが表示マップ (テキスト書式) またはプリンター・マップ (印刷書式) のどちらであるかを判別します。</p> <p>装置がディスプレイまたはプリンターのどちらであるのかを判別するためには、338 ページの表 97 を参照してください。</p>
ヘルプ・マップ名	helpForm	特別な考慮事項なし。
ヘルプ・キー	helpKey	特別な考慮事項なし。
バイパス・キー	validationBypassKeys	特別な考慮事項なし。
マップに対して最大 5 つのバイパス・キーを指定できます。	フォームに対して最大 5 つの validationBypassKeys を指定できます。	

表 99. 表示マップ - 一般構文、マップ・タイプ、およびプロパティ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
変数フィールドの大文字への変換	書式に対してはサポートされません。書式にある CHAR または MBCHAR の変数フィールドごとに、ユーザーが入力するデータが自動的に大文字に変換されるかどうかを指定する必要があります。	<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> 変数フィールドの大文字への変換がマップ全体に関して指定されている場合、マイグレーション・ツールは、すべての CHAR および MBCHAR フィールドの upperCase プロパティを YES に設定します。 変数フィールドの大文字への変換がマップ全体に関して指定されていない場合、マイグレーション・ツールは、それぞれの CHAR および MBCHAR フィールドごとに指定された大文字変換情報を使用して、そのフィールドに upperCase プロパティを設定するかどうかを決定します。
マップ・サイズ - 行と列	formSize = [Lines, Columns]	特別な考慮事項なし。
開始位置 - 行と列	position = [Line, Column]	浮動マップが選択された場合、マイグレーション・ツールは position プロパティを省略します。
マップが浮動マップである場合は、NEXT,SAME が必要です。	position プロパティが省略された場合、書式は浮動書式になります。	
浮動マップ	適用不可 position プロパティが省略された場合、書式は浮動書式になります。	浮動マップが選択された場合、マイグレーション・ツールは position プロパティを省略します。
装置タイプ - ディスプレイまたは DBCS ディスプレイ	type TextForm	マイグレーション・ツールは、装置タイプ情報を使用して、マップをテキスト書式または印刷書式のどちらにマイグレーションするかを決定します。
サポートされる装置	screenSizes = [[Lines, Columns], [Lines, Columns]]	マイグレーション・ツールは、装置タイプ情報を使用して、対応する screenSizes プロパティを決定します。複数の VAGen 装置に同じ画面サイズが指定されている場合、マイグレーション・ツールは画面サイズを 1 回のみ組み込みます。
注: サポートされる装置は、装置タイプ、行数、および列数を指示します。	注: 書式に関してサポートしたいそれぞれの画面サイズごとに、[Lines, Columns] の対を組み込んでください。	
適用不可 VisualAge Generator の場合、メッセージ・フィールドの名前は常に EZEMSG です。	msgField これは、EGL エラー・メッセージを格納するフィールドの名前です。	EZEMSG がマップ上のどこかに存在する場合、マイグレーション・ツールは msgField プロパティを設定します。

表 99. 表示マップ - 一般構文、マップ・タイプ、およびプロパティ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
適用不可	alias	<p>マイグレーション・ツールの拡張予約語リストとの競合、またはマップ名の先頭が # または @ シンボルであることが原因で、マップ名を変更する必要がある場合、マイグレーション・ツールは alias プロパティをインクルードします。また、プログラムのメインのマップ・グループにあるマップ名との競合が原因でマップの名前を変更する必要がある場合、マイグレーション・ツールはヘルプ・マップ・グループ内のマップに関して alias プロパティを組み込みます。</p> <p>特別な考慮事項が適用されます。詳しくは、97 ページの『マップ名とヘルプ・マップ名』を参照してください。</p>

表 100. プリンター・マップ - 一般構文、マップ・タイプ、およびプロパティ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>プリンター・マップには、次の情報を含めることができます。</p> <ul style="list-style-type: none"> マップ・グループ名とマップ名 マップ・プロパティ <ul style="list-style-type: none"> 一般プロパティ <ul style="list-style-type: none"> ヘルプ・マップ名 ヘルプ・キー バイパス・キー 変数フィールドの大文字への変換 SO/SI が占める位置 レイアウト・プロパティ <ul style="list-style-type: none"> マップ・サイズ 開始位置 浮動マップ 装置 <ul style="list-style-type: none"> タイプ (ディスプレイまたは印刷) サポートされる装置 定数フィールド 変数フィールド 変数フィールドのフィールド編集順序 	<p>印刷書式には、次の情報を含めることができます。</p> <ul style="list-style-type: none"> フォーム名 書式プロパティ 定数フィールド 変数フィールド <p>マイグレーション・ツールによって作成される印刷書式のフォーマットを、次の例に示します。</p> <pre>Form mapName type printForm {formSize=[255,158], position=[1,1], addSpaceForSOSI=yes } [variableFields] [constantFields] end // end mapName</pre>	<p>マイグレーション・ツールは、VAGen 装置タイプを使用して、マップが表示マップ (テキスト書式) またはプリンター・マップ (印刷書式) のどちらであるかを判別します。</p> <p>マイグレーション・ツールは、印刷書式の次のプロパティを常に省略します。</p> <ul style="list-style-type: none"> 一般プロパティ <ul style="list-style-type: none"> ヘルプ・マップ名 ヘルプ・キー バイパス・キー 変数フィールドの大文字への変換 装置 <ul style="list-style-type: none"> サポートされる装置 変数フィールドのフィールド編集順序 <p>装置がディスプレイまたはプリンターのどちらであるのかを判別するためには、338 ページの表 97 を参照してください。</p>
ヘルプ・マップ名	印刷書式には適用されません。	マイグレーション・ツールは、印刷書式のこのプロパティを省略します。
ヘルプ・キー	印刷書式には適用されません。	マイグレーション・ツールは、印刷書式のこのプロパティを省略します。
バイパス・キー	印刷書式には適用されません。	マイグレーション・ツールは、印刷書式のこのプロパティを省略します。
変数フィールドの大文字への変換	印刷書式には適用されません。	マイグレーション・ツールは、印刷書式のこのプロパティを省略します。
SO/SI が占める位置	addSpaceForSOSI	特別な考慮事項なし。
マップ・サイズ - 行と列	formSize = [Lines, Columns]	特別な考慮事項なし。
開始位置 - 行と列	position = [Line, Column]	浮動マップが選択された場合、マイグレーション・ツールは position プロパティを省略します。
マップが浮動マップである場合は、NEXT,SAME が必要です。	position プロパティが省略された場合、書式は浮動書式になります。	
浮動マップ	適用不可 position プロパティが省略された場合、書式は浮動書式になります。	浮動マップが選択された場合、マイグレーション・ツールは position プロパティを省略します。

表 100. プリンター・マップ - 一般構文、マップ・タイプ、およびプロパティ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
装置タイプ - プリンターまたは DBCS プリンター	タイプ printForm	マイグレーション・ツールは、装置タイプ情報を使用して、マップをテキスト書式または印刷書式のどちらにマイグレーションするかを決定します。
サポートされる装置	印刷書式には適用されません。	マイグレーション・ツールは、印刷書式のこのプロパティを省略します。
適用不可 VisualAge Generator の場合、メッセージ・フィールドの名前は常に EZEMSG です。	msgField これは、EGL エラー・メッセージを格納するフィールドの名前です。	EZEMSG がマップ上のどこかに存在する場合、マイグレーション・ツールは msgField プロパティを設定します。
適用不可	alias	<p>マイグレーション・ツールの拡張予約語リストとの競合、またはマップ名の先頭が # または @ シンボルであることが原因で、マップ名を変更する必要がある場合、マイグレーション・ツールは alias プロパティをインクルードします。また、プログラムのメインのマップ・グループにあるマップ名との競合が原因でマップの名前を変更する必要がある場合、マイグレーション・ツールはヘルプ・マップ・グループ内のマップに関して alias プロパティを組み込みます。</p> <p>特別な考慮事項が適用されます。詳しくは、97 ページの『マップ名とヘルプ・マップ名』を参照してください。</p>

表 101. マップの定数フィールドと変数フィールド - 一般情報

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>マップ上の位置はすべて、次のいずれかで記述する必要があります。</p> <ul style="list-style-type: none"> 変数フィールドとして 定数フィールドとして 定数フィールドまたは変数フィールドの先頭にある属性バイトとして 	書式上の位置は、すべて記述する必要はありません。デフォルト・プロパティ (noHighLight 、 normalIntensity 、 skipProtect 、 defaultColor 、no outlining、および no cursor) を持つブランク定数は、指定する必要はありません。	マイグレーション・ツールは、デフォルト・プロパティをもつブランク定数を省略します。

表 101. マップの定数フィールドと変数フィールド - 一般情報 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>表示マップの定数フィールドには、実際には定数に適用しない属性を指定できます。例えば、次のようになります。</p> <ul style="list-style-type: none"> • 無保護 • 入力必須 • 入力時充てん必須 (Require fill on input) • 数値属性 • 変更データ・タグ 	<p>テキスト書式の定数フィールドには、定数に対して意味をなさないプロパティは指定できません。</p>	<p>サポートされないプロパティの場合、マイグレーション・ツールはテキスト書式の定数のプロパティを省略します。</p>
<p>プリンター・マップの定数フィールドには、実際にはプリンターに適用されない属性を指定できます。例えば、次のようになります。</p> <ul style="list-style-type: none"> • 色 • 輝度 • 下線以外の強調表示 • 保護 • 初期カーソル・フィールド • ライト・ペン検出 	<p>印刷書式の定数フィールドには、定数に対して意味をなさないプロパティは指定できません。</p>	<p>サポートされないプロパティの場合、マイグレーション・ツールは印刷書式の定数のプロパティを省略します。</p>
<p>プリンター・マップの変数フィールドには、実際にはプリンターに適用されない属性を指定できます。例えば、次のようになります。</p> <ul style="list-style-type: none"> • 色 • 輝度 • 下線以外の強調表示 • 保護 • 初期カーソル・フィールド • 入力必須 • 入力時充てん必須 (Require fill on input) • 数値属性 • 変更データ・タグ • ライト・ペン検出 	<p>印刷書式の変数フィールドには、印刷書式に対して意味をなさないプロパティは指定できません。</p>	<p>サポートされないプロパティの場合、マイグレーション・ツールは印刷書式の変数フィールドのプロパティを省略します。</p>

表 101. マップの定数フィールドと変数フィールド - 一般情報 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>プリンター・マップの変数フィールドには、実際にはプリンターに適用されない編集を指定できます。例えば、次のようになります。</p> <ul style="list-style-type: none"> • 編集ルーチン • 最小入力 • 大文字変換 • 16 進編集 • 入力必須 • 最小値 • 最大値 • 編集メッセージ 	<p>印刷書式の変数フィールドには、印刷書式に対して意味をなさないプロパティは指定できません。</p>	<p>サポートされないプロパティの場合、マイグレーション・ツールは印刷書式の変数フィールドのプロパティを省略します。</p>

表 102. マップの定数フィールドと変数フィールド - 一般構文、データ型、長さ、小数部、および説明

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>マップ上の変数フィールドには、次の情報を含めることができます。</p> <ul style="list-style-type: none"> 名前 マップにドロップしたものに基づく次の情報: <ul style="list-style-type: none"> データ型 位置 基本情報: <ul style="list-style-type: none"> 説明 初期値 長さ (バイト) 配列指標 数値編集 属性 編集 (小数部の桁数など) エラー・メッセージ <p>注: 位置は、属性バイトの位置です。バイト単位の長さは、フィールドの長さから属性バイトを除いたバイト数です。バイト単位の長さは、データ値の長さにも使用されます。</p>	<p>書式上の変数フィールドには、次の情報を含めることができます。</p> <ul style="list-style-type: none"> 名前 文字フィールドのタイプと文字数単位の長さ 数値フィールドの型、精度、およびスケール 位置 バイト単位のフィールド長 表示プロパティ フォーマット設定プロパティ 検証プロパティ 値 <p>通常、VAGen マップと EGL 書式は、次のように対応します。</p> <ul style="list-style-type: none"> VAGen の属性は、EGL の表示プロパティに対応します。 VAGen の編集およびメッセージは、EGL のフォーマット設定プロパティ、または検証プロパティに対応します。 ただし、VAGen の属性と編集の一部には、単一の EGL プロパティにマージされたものや、別のカテゴリーに移動したものがあります。 <p>EGL 変数フィールドの例を次に示します。</p> <pre>itemName dataType(lengthInformation) // description { position=[row,column], fieldLen=length, validationOrder=n, [presentationProperties] [formattingProperties] [value="initialValue"] [arrayInformation] }</pre> <p>注: 位置 は、属性バイトの位置です。 fieldLen は、フィールドの長さから属性バイトを除いたバイト数です。 <i>dataType(lengthInformation)</i> に指定されるプリミティブ情報は、データ値の長さです。</p>	<p>マイグレーション・ツールは、EGL の fieldLen プロパティを VAGen のバイト単位の長さに設定します。ツールは、<i>dataType</i> の <i>lengthInformation</i> を次のように設定します。</p> <ul style="list-style-type: none"> CHAR、DBCHAR、および MBCHAR のフィールドについては、マイグレーション・ツールは <i>lengthInformation</i> をバイト数ではなく文字数に設定します。 数値編集を指定している VAGen の char フィールドについては、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> フィールドを EGL の NUM 型に変換します。 精度を VAGen のバイト単位の長さに設定し、また VisualAge Generator のフィールドに対して小数部が指定されている場合は、精度を 1 だけ減らします。 スケールを VisualAge Generator 内で指定されている小数部の桁数に設定します。 <p>特別な考慮事項が適用されます。詳しくは、以下のセクションを参照してください。</p> <ul style="list-style-type: none"> 99 ページの『数値変数フィールド』 103 ページの『名前なしマップ変数フィールド』 105 ページの『row=0, column=0 にあるフィールド』

表 102. マップの定数フィールドと変数フィールド - 一般構文、データ型、長さ、小数部、および説明 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>マップ上の定数フィールドには、次の情報を含めることができます。</p> <ul style="list-style-type: none"> • マップにドロップしたものに基づく次の情報: <ul style="list-style-type: none"> – データ型 – 位置 • 基本情報: <ul style="list-style-type: none"> – 初期値 – 長さ (バイト) • 属性 <p>注: 位置は、属性バイトの位置です。バイト単位の長さは、フィールドの長さから属性バイトを除いたバイト数です。</p>	<p>書式上の定数フィールドには、次の情報を含めることができます。</p> <ul style="list-style-type: none"> • 位置 • フィールド長 • 表示プロパティ • 値 <p>通常、VAGen マップと EGL 書式は、次のように対応します。</p> <ul style="list-style-type: none"> • VAGen の属性は、EGL の表示プロパティに対応します。 • 入力編集のみに適用される属性は、EGL 定数フィールドに対してはサポートされません。 <p>定数のデータ型は、value プロパティに基づいて決定されます。</p> <p>EGL 定数フィールドの例を次に示します。</p> <pre>{ position=[row,column], fieldLen=length, [presentationProperties] [value="initialValue"] }</pre> <p>注: 位置 は、属性バイトの位置です。 fieldLen は、フィールドの長さから属性バイトを除いたバイト数です。</p>	<p>マイグレーション・ツールは、EGL の fieldLen プロパティを VisualAge Generator の長さに設定します。</p> <p>特別な考慮事項が適用されます。詳しくは、以下のセクションを参照してください。</p> <ul style="list-style-type: none"> • 104 ページの『無保護マップ定数』 • 105 ページの『row=0, column=0 にあるフィールド』
<p>データ型:</p> <ul style="list-style-type: none"> • 文字定数 • 文字変数 • DBCS 定数 • DBCS 変数 • 混合定数 • 混合変数 • 数値編集を選択した文字変数 <p>注: 型は、マップにドロップしたフィールドのタイプ、および「数値編集 (Numeric edit)」ボックスを選択したかどうかによって決まります。</p>	<p>EGL データ型:</p> <ul style="list-style-type: none"> • 適用不可 • CHAR • 適用不可 • DBCHAR • 適用不可 • MBCHAR • NUM 	<p>特別な考慮事項なし。</p>
説明	適用不可	<p>マイグレーション・ツールは、説明をコメントに変換し、データ型と長さの情報の後に配置します。</p>

表 102. マップの定数フィールドと変数フィールド - 一般構文、データ型、長さ、小数部、および説明 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
初期値	<p>値</p> <p>注:</p> <ul style="list-style-type: none"> VisualAge Generator 互換モードでは、value プロパティは、値が割り当てられていないフィールドを画面に表示するときのみ使用されます。 value プロパティを使用して、ストレージ内のフィールドの初期値が設定されることはありません。 VisualAge Generator 互換モードが指定されていない場合、value プロパティはプログラムの開始時にプログラムのフィールドの初期値を指定します。 	特別な考慮事項なし。
長さ	<p>EGL 変数フィールドには、次の特性があります。</p> <ul style="list-style-type: none"> length (フィールド内の文字数または桁数) fieldLen プロパティ (マップ上でフィールドが占める、属性バイトを除いたスペース) 	<p>マイグレーション・ツールは、VAGen の長さを使用して、EGL の長さとして EGL の fieldLen プロパティの両方を設定します。数値フィールドに関しては、特別な考慮事項が適用されません。詳しくは、99 ページの『数値変数フィールド』を参照してください。</p>
<p>配列指標</p> <p>注:</p> <ul style="list-style-type: none"> 配列サイズは、変数フィールドの最も大きい配列指標に基づいて決定されます。 カーソル位置、色、強調表示、輝度、保護、カーソル位置など、配列要素の属性の一部はオーバーライドできません。 配列要素の初期値をオーバーライドすることもできます。 	<pre>itemName datatype(lengthInfo) [arraySize] { propertiesForIndex1 , indexOrientation = across down, columns = n1, linesBetweenRows = n2, spacesBetweenColumns = n3, this[n] { propertiesForIndexN } } itemName datatype(lengthInfo) [arraySize] { propertiesForIndex1, this[n] { positionForIndexN, propertiesForIndexN } }</pre> <p>注:</p> <ul style="list-style-type: none"> 配列サイズは、<i>datatype</i> と <i>lengthInfo</i> の直後に指定されます。 カーソル位置、および color、highlight、intensity、protect などの表示プロパティはオーバーライドできません。 value プロパティもオーバーライドできます。 	<p>標準配列では、マイグレーション・ツールは indexOrientation、columns、linesBetweenRows、および spacesBetweenColumns プロパティを使用して位置情報を提供します。カーソル位置または配列要素の表示プロパティが配列の最初の要素と異なる場合、このツールは this[n] のみをインクルードします。</p> <p>非標準配列では、マイグレーション・ツールは配列の各要素ごとに、最初に position = [row,column] プロパティを指定した後で、this[n] をインクルードします。カーソル位置または配列要素の表示プロパティが配列の最初の要素と異なる場合、このツールはこれらもインクルードします。</p>

表 102. マップの定数フィールドと変数フィールド - 一般構文、データ型、長さ、小数部、および説明 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
フィールド編集順序 注: <ul style="list-style-type: none"> フィールド編集順序は、「定義」メニューから指定します。 デフォルトのフィールド編集順序は、マップ上の変数フィールドの位置によって決まります (左から右、上から下の順)。 システム共通プロダクトと VisualAge Generator の一部のバージョンは、フィールド編集順序を外部ソース形式で記録しません。 	validationOrder 注: デフォルトの validationOrder は、マップ上の変数フィールドの位置によって決まります (左から右へ向かって、上から下への順)。	マイグレーション・ツールは、マップの外部ソース形式に含まれていない validationOrder プロパティを省略します。

表 103. マップの定数フィールドと変数フィールド - 属性

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
輝度: <ul style="list-style-type: none"> 標準 低輝度 (Dark) 高輝度 (Bright) 	intensity: <ul style="list-style-type: none"> normalIntensity invisible bold (表示プロパティ)	特別な考慮事項なし。
強調表示: <ul style="list-style-type: none"> 強調表示なし 点滅 反転表示 (Reverse video) 下線 (Underscore) 	highlight: <ul style="list-style-type: none"> noHighlight blink reverse underline (表示プロパティ)	特別な考慮事項なし。
保護: <ul style="list-style-type: none"> 無保護 保護 自動スキップ 	protect: <ul style="list-style-type: none"> noProtect protect skipProtect (表示プロパティ)	特別な考慮事項なし。

表 103. マップの定数フィールドと変数フィールド - 属性 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
色: <ul style="list-style-type: none"> モノクロ (Mono) 青 赤 ピンク 緑 青緑色 黄色 白 	color: <ul style="list-style-type: none"> defaultColor blue red magenta green cyan yellow white (表示プロパティ)	特別な考慮事項なし。
初期カーソル・フィールド	cursor = yes no (書式フィールド・プロパティ)	特別な考慮事項なし。
入力必須	inputRequired (検証プロパティ)	マイグレーション・ツールは、VAGen の「入力必須 (Input required)」属性と入力必須編集を次の方法でマージします。 <ul style="list-style-type: none"> 「入力必須 (Input required)」属性または入力必須編集のどちらかが選択されている場合、マイグレーション・ツールは inputRequired プロパティを組み込みます。 どちらも選択されていない場合、マイグレーション・ツールは inputRequired プロパティを省略します。
入力時充てん必須 (Require fill on input)	fill (検証プロパティ)	特別な考慮事項なし。
数値属性 注: このプロパティは、CHA フィールド (「数値編集 (Numeric edit)」が選択されている CHA フィールドを含む) に関してサポートされています。	isDecimalDigit (検証プロパティ) 注: このプロパティは、CHAR フィールドに関してのみサポートされます。	数値属性が選択されている場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> CHAR フィールドに isDecimalDigit プロパティを組み込みます。 数値フィールドの isDecimalDigit プロパティを省略します。VAGen との互換性を保つために、EGL は数値フィールドのソフトウェア編集機能を備えています。 詳しくは、101 ページの『マップ・フィールドと数値ハードウェア属性』を参照してください。
変更データ・タグ	modified (表示プロパティ)	特別な考慮事項なし。

表 103. マップの定数フィールドと変数フィールド - 属性 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
ライト・ペン検出	detectable (表示プロパティ)	特別な考慮事項なし。
アウトライン: <ul style="list-style-type: none"> 左 (left) 右 (right) 上 (over) 下 (under) ボックス (box) 	outline: <ul style="list-style-type: none"> left right top bottom box (表示プロパティ)	特別な考慮事項なし。

表 104. マップ変数フィールド - 一般編集

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
編集ルーチン	validatorFunction OR validatorDataTable (検証プロパティ)	<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> マップ編集ルーチンが EZEC10 または EZEC11 であれば、validatorFunction プロパティを設定します。 編集ルーチンが関数であれば、validatorFunction プロパティを設定します。 編集ルーチンがテーブルであれば、validatorDataTable プロパティを設定します。 <p>注: マイグレーション時に編集ルーチンが使用できない場合は、特別な考慮事項が適用されます。詳細と起こりうる問題については、100 ページの『マップ変数フィールドと編集ルーチン』を参照してください。</p>
位置調整 - 左揃え 右揃え なし 注: マップ項目のデフォルトは、数値フィールドの場合は右揃え、その他のフィールドの場合は左揃えです。	align = left right none (フォーマット設定プロパティ) 注: 書式フィールドのデフォルトは、数値フィールドの場合は right 、その他のフィールドの場合は left です。	特別な考慮事項なし。

表 104. マップ変数フィールド - 一般編集 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>日付編集マスク</p> <p>以下の値が有効です。</p> <ul style="list-style-type: none"> • SYSGREGRN • SYSJULIAN • dateEditPattern 	<p>dateFormat = <i>value</i></p> <p>以下の EGL 値は、VAGen 値に対応しています。</p> <ul style="list-style-type: none"> • systemGregorianCalendarFormat • systemJulianDateFormat • "dateEditPattern" <p>(フォーマット設定プロパティ)</p> <p>注: <i>dateEditPattern</i> の中で、マイグレーション・ツールは次の EGL 表記への変換を行います。</p> <ul style="list-style-type: none"> • 年を示す yy または yyyy。 • 月を示す MM。 • 日を示す dd。 • 年の通算日を示す DDD。 	<p>特別な考慮事項なし。</p>
<p>最小入力</p>	<p>minimumInput (検証プロパティ)</p>	<p>特別な考慮事項なし。</p>
<p>充てん文字</p> <p>注: 文字、DBCS、または混合の各フィールドの場合、マップに使用される項目のデフォルト充てん文字は NULL です。数値フィールドの場合はブランク、16 進フィールドの場合は 0 です。</p>	<p>fillCharacter</p> <p>(フォーマット設定プロパティ)</p> <p>注:</p> <ul style="list-style-type: none"> • CHAR、DBCHAR、または MBCHAR の各フィールドの場合、マップに使用される項目のデフォルト充てん文字は NULL です。数値フィールドの場合はブランク、HEX フィールドの場合は 0 です。 • strLib.nullFill は、NULL 充てん文字を表す EGL 定数です。あるいは、"" (2 つの連続する引用符) を使用します。 	<p>特別な考慮事項なし。</p>

表 104. マップ変数フィールド - 一般編集 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
大文字変換	upperCase (フォーマット設定プロパティ)	<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> 変数フィールドの大文字への変換がマップ全体に関して指定されている場合、マイグレーション・ツールは、すべての CHAR および MBCHAR フィールドの upperCase プロパティを YES に設定します。 変数フィールドの大文字への変換がマップ全体に関して指定されていない場合、マイグレーション・ツールは、それぞれの CHAR および MBCHAR フィールドごとに指定された大文字変換情報を使用して、そのフィールドに upperCase プロパティを設定するかどうかを決定します。
16 進編集	isHexDigit (検証プロパティ)	特別な考慮事項なし。
入力必須	inputRequired (検証プロパティ)	<p>マイグレーション・ツールは、VAGen の「入力必須 (Input required)」属性と入力必須編集を次の方法でマージします。</p> <ul style="list-style-type: none"> 「入力必須 (Input required)」属性または入力必須編集のどちらかが選択されている場合、マイグレーション・ツールは inputRequired プロパティを組み込みます。 どちらも選択されていない場合、マイグレーション・ツールは inputRequired プロパティを省略します。
SO/SI スペースの検査	needsSOSI (検証プロパティ)	特別な考慮事項なし。

表 105. マップ変数フィールド - 数値編集

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>最小値と最大値</p> <p>注: 最小値または最大値のどちらかを指定する場合は、両方を指定する必要があります。</p>	<p>validValues = [[<i>minimumValue</i>, <i>maximumValue</i>]] (検証プロパティ)</p> <p>注: 複数の値の対、および単一値を validValues プロパティにリストできます。</p>	<p>マイグレーション・ツールは、最小値と最大値を結合して EGL の validValues プロパティを指定します。</p>

表 105. マップ変数フィールド - 数値編集 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>Sign = None Leading Trailing</p> <p>注: マップ上の数値項目に対するデフォルト記号は none です。</p>	<p>sign = none leading trailing (フォーマット設定プロパティ)</p> <p>注: 数値フィールドのデフォルトの sign は常に leading です。</p>	<p>数値フィールドについて、マイグレーション・ツールは次のうち最初に該当する基準に基づいて記号のマイグレーションを行います。</p> <ul style="list-style-type: none"> マップ記号編集が指定されている場合、ツールは対応する sign プロパティにマイグレーションします。 マップ記号編集が指定されていない場合、ツールは sign プロパティを none に設定します。
通貨	<p>currency = YES NO currencySymbol = "記号" (フォーマット設定プロパティ)</p> <p>注: currency = YES で、currencySymbol は指定されていない場合、実行時に実際に使用される通貨記号は、VisualAge Generator の場合と同じように設定されます。</p>	<p>マイグレーション・ツールは、currency を YES または NO に設定するのみで、書式変数フィールドに関して currencySymbol = "symbol" を設定することはありません。これは、VisualAge Generator に同等の情報が存在しないからです。</p>
セパレーター	<p>numericSeparator (フォーマット設定プロパティ)</p>	特別な考慮事項なし。
ゼロ編集	<p>zeroFormat (フォーマット設定プロパティ)</p>	特別な考慮事項なし。

表 106. マップ変数フィールド - エラー・メッセージ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
編集ルーチン	validatorFunctionMsgKey または validatorDataTableMsgKey (検証プロパティ)	<p>マイグレーション・ツールは、編集ルーチン・メッセージを次の方法でマイグレーションします。</p> <ul style="list-style-type: none"> 編集ルーチンが EZEC10 または EZEC11 であれば、validatorFunctionMsgKey を設定します。 編集ルーチンがテーブルであれば、validatorDataTableMsgKey を設定します。 編集ルーチンが関数ならば、この状態では VAGen 内でメッセージは使用されないため、編集ルーチン・メッセージをマイグレーションしません。 <p>詳細と起こりうる問題については、100 ページの『マップ変数フィールドと編集ルーチン』を参照してください。</p>
最小入力	minimumInputMsgKey (検証プロパティ)	特別な考慮事項なし。
入力必須	inputRequiredMsgKey (検証プロパティ)	特別な考慮事項なし。
データ型	typeChkMsgKey (検証プロパティ)	特別な考慮事項なし。
数値範囲	validValuesMsgKey (検証プロパティ)	特別な考慮事項なし。

プログラム

プログラムに関するセクションは、次の表で構成されています。

- プログラム - 一般構文、プログラム・タイプ、呼び出し先パラメーター、およびプロローグ (357 ページの表 107)
- プログラム - プログラム仕様、プロパティ、テーブル、および追加レコード・リスト (359 ページの表 108)
- プログラム - main 関数とフロー・ステートメント (363 ページの表 109)

表 107. プログラム - 一般構文、プログラム・タイプ、呼び出し先パラメーター、およびプロローグ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>プログラム・パーツ:</p> <ul style="list-style-type: none"> • <code>programName</code> • プログラム・タイプ • 仕様 (プログラム・タイプによって異なる): <ul style="list-style-type: none"> – 作業用ストレージ・レコード – PSB – 先頭マップ – 先頭 UI レコード – マップ・グループ – ヘルプ・マップ・グループ • テーブルおよび追加レコード • 呼び出し先パラメーター • プロローグ • プロパティ (プログラム・タイプによって異なる) • 構造ダイアグラム <ul style="list-style-type: none"> – <code>main</code> 関数 – フロー・ステートメント (構造ダイアグラム内では非表示、ただし任意の <code>main</code> 関数に対して指定可能) 	<p>EGL 構文の例:</p> <pre> //*** Program=programName // prolog //***** Program programName type eglProgramType //vagenProgramType [(calledParameters)] { [alias= "originalProgramName"] includeReferencedFunctions =yes, allowUnqualifiedItemReferences =yes, throwNrfEofExceptions=yes, handHardIOErrors=no V60ExceptionCompatibility = yes, I4GLItemsNullable = no, textLiteralDefaultIsString = no, localSQLScope=yes, [propertiesBasedOnType] } [dataDeclarations] [useDeclarations] function main () { functionLabel: functionName() ; [functionFlowStatements]]} end // end main end // end programName </pre>	<p>プログラム名が EGL 予約語リストと競合していても、マイグレーション・ツールはプログラムの名前を変更しません。マイグレーション・ツールは alias プロパティを設定しません。プログラムの名前を変更する必要がある場合は、alias プロパティを使用して、VAGen プログラムのオリジナルの名前を指定してください。107 ページの『プログラム名と予約語』を参照してください。</p> <p>マイグレーション・ツールは、プログラム定義にコメントとして VAGen プログラム・タイプを組み込みます。</p> <p>マイグレーション・ツールは、テーブルと追加レコードのリストを次のようにマイグレーションします。</p> <ul style="list-style-type: none"> • レコードを <i>dataDeclarations</i> にマイグレーションします。 • テーブルを <i>useDeclarations</i> にマイグレーションします。 <p>マイグレーション・ツールは、VAGen の振る舞いを保持するために常に次のプロパティを組み込みます。</p> <ul style="list-style-type: none"> • includeReferencedFunctions • allowUnqualifiedItemReferences • throwNrfEofExceptions • handleHardIOErrors • V60ExceptionCompatibility • I4GLItemsNullable • textLiteralDefaultIsString • localSQLScope
<p>プログラム・タイプ:</p> <ul style="list-style-type: none"> • メインのトランザクション • 呼び出し先トランザクション • メインのバッチ • 呼び出し先バッチ • Web トランザクション <p>注: 詳しくは、この後の行にある『メインのトランザクションの実行モード値』を参照してください。</p>	<p>EGL プログラム・タイプ:</p> <ul style="list-style-type: none"> • <code>textUIProgram</code> • <code>textUIProgram</code> • <code>basicProgram</code> • <code>basicProgram</code> • <code>VGWebTransaction</code> 	<p>マイグレーション・ツールは、プログラム定義にコメントとして VAGen プログラム・タイプを組み込みます。セグメンテーション値と EGL プロパティの対応については、359 ページの表 108 を参照してください。</p>

表 107. プログラム - 一般構文、プログラム・タイプ、呼び出し先パラメーター、およびプロローグ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>呼び出し先パラメーター</p> <p>注:</p> <ul style="list-style-type: none"> 呼び出し先パラメーターは、専用のウィンドウに入力されます。 パラメーター・タイプは、パラメーターが項目、レコード、またはマップのどれであるかを示します。 パラメーター名は、常に別の VAGen パーツの名前です。EGL の型定義またはプリミティブ型に相当するものではありません。 次の 2 つの特殊機能語がパラメーターとしてサポートされています。 <ul style="list-style-type: none"> EZEDLPSB EZEDLPCB[n] (ここで <i>n</i> は数値リテラルです) <p>詳しくは、この表の次の行を参照してください。</p>	<p>EGL 呼び出し先パラメーターの例:</p> <pre>(parameterName typeInfo { , parameterName typeInfo })</pre> <p>注:</p> <ul style="list-style-type: none"> パラメーターはコンマで区切る必要があります。 パラメーターは、DataItem、レコード、または書式のいずれかです。VAGen パラメーター型に直接対応するものではありません。 EGL <i>typeInfo</i> は、次のいずれかです。 <ul style="list-style-type: none"> DataItem のプリミティブ型 DataItem、Record、または Form の型定義 	<p>マイグレーション・ツールは、オリジナルの VAGen パラメーターの型をコメントとして組み込みます。</p> <p>マイグレーション構文設定「共用データ項目をプリミティブ項目定義に変換」を選択した場合に、データ項目パーツが使用可能であれば、共用項目はマイグレーション・ツールによって EGL パラメーターに変換されます。この変数はデータ項目パーツに対して指定された型、長さ、および 10 進数に基づいたプリミティブ定義を使用して宣言されます。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。</p> <p>マイグレーション構文設定「選択済み共用データ項目をプリミティブ項目定義に変換」を選択解除した場合、またはデータ項目パーツが使用不可の場合は、共用項目はマイグレーション・ツールによって EGL パラメーターに変換されます。この変数は、型定義を使用して宣言されます。マイグレーション用の型定義は、常に項目名と同じです。</p> <p>特別な考慮事項が適用されます。詳細と起こりうる問題については、86 ページの『再定義レコード』を参照してください。</p>
<p>呼び出し先パラメーター: EZEDLPSB</p>	<p>PSB をパスするための EGL 呼び出し先パラメーターの例:</p> <pre>parameter リスト: (psbData PSBDataRecord) program プロパティー: { @DLI { psb = psb, psbParm = psbData }}</pre>	<p>特別な考慮事項なし。</p>

表 107. プログラム - 一般構文、プログラム・タイプ、呼び出し先パラメーター、およびプロローグ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>呼び出し先パラメーター: EZEDLPCB[n] (ここで n は数値リテラルです)</p>	<p>PCB をパスするための EGL 呼び出し先パラメーターの例:</p> <p>parameter リスト: (pcbName pcbType_PCBRecord)</p> <p>program プロパティ: { @DLI { psb = psb, pcbParms = [pcbList] } }</p> <p>注:</p> <ul style="list-style-type: none"> pcbList は、PCB パラメーターの名前をプログラムの PSBRecord 内の対応する位置に突き合わせるために使用されます。 pcbType の値は次のとおりです。 <ul style="list-style-type: none"> IO ALT DB GSAM 	<p>マイグレーション・ツールは、常に pcbn という形式で pcbName を使用します (ここで n は、VAGen 呼び出し先パラメーター・リストで使用されるものと同じ数値リテラルです)。</p> <p>プログラムによって指定された PSB パーツが使用可能な場合、マイグレーション・ツールは次の処理を行います。</p> <ul style="list-style-type: none"> pcbType 情報をインクルードします。 それぞれの PCB パラメーターを EGL PSBRecord 内の対応する PCB に関連付けるための、pcbList 情報をインクルードします。 <p>プログラムの PSB パーツが使用不可である場合、特別な考慮事項が適用されます。詳しくは、110 ページの『呼び出しパラメーター・リストに EZEDLPCB が含まれるプログラム』を参照してください。</p>
プロローグ	適用不可	マイグレーション・ツールは、プロローグをコメントに変換し、プログラム定義の前に配置します。

表 108. プログラム - プログラム仕様、プロパティ、テーブル、および追加レコード・リスト

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>一般情報:</p> <ul style="list-style-type: none"> VAGen プロパティと仕様の一部は、EGL プロパティ、データ宣言、または use 宣言にマイグレーションされます。 	<p>一般情報:</p> <ul style="list-style-type: none"> 後続の行によって、対応する EGL 言語要素がプログラム・プロパティ、データ宣言、または use 宣言のどれであるかが示されます。 	特別な考慮事項なし。
<p>メインのトランザクションの実行モード値:</p> <ul style="list-style-type: none"> 非セグメント化 (Nonsegmented) セグメント化 単一セグメント (Single segment) <p>注: 呼び出し先トランザクションは、常に非セグメント化モードで実行されます。</p>	<p>segmented — 値は次のとおりです。</p> <ul style="list-style-type: none"> segmented = no segmented = yes segmented = yes <p>(プログラム・プロパティ)</p> <p>注: segmented プロパティは、呼び出し先プログラムに関しては指定されません。</p>	<p>セグメント化情報が外部ソース形式ファイルに含まれていない場合、マイグレーション・ツールは segmented プロパティをデフォルト値の NO に設定します。</p>

表 108. プログラム - プログラム仕様、プロパティ、テーブル、および追加レコード・リスト (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>作業用ストレージ・レコード (仕様)</p> <ul style="list-style-type: none"> 作業用ストレージ・レコードは、メインプログラムと呼び出し先プログラムの両方に対して指定できます。このレコードは、プログラムの 1 次作業用ストレージ・レコードと呼ばれることもあります。 1 次作業用ストレージ・レコードは、常に初期化されます。 	<p>inputRecord (プログラム・プロパティ)</p> <ul style="list-style-type: none"> inputRecord プロパティは、メインプログラムについてのみ指定できます。 入力レコードは常に初期化されます。 データ宣言も必要です。 	<p>マイグレーション・ツールは、1 次作業用ストレージ・レコードをメインプログラムの inputRecord プロパティに変換します。</p> <p>またマイグレーション・ツールは、メインプログラムと呼び出し先プログラムの両方に 1 次作業用ストレージ・レコードのデータ宣言を組み込みます。ツールは、呼び出し先プログラムのデータ宣言に initialized = yes プロパティを組み込みます。</p> <p>1 次作業用ストレージ・レコードにレベル 77 項目が含まれている場合、マイグレーション・ツールはレベル 77 レコードのデータ宣言ステートメントを組み込みます。</p> <p>詳細と起こりうる問題については、87 ページの『レコード内のレベル 77 項目』を参照してください。</p>
<p>PSB (仕様)</p> <p>注: VisualAge Generator では、PSB はパーツ型です。</p>	<p>EGL は、PSB を指定するためにプログラム・プロパティとレコード宣言の両方を使用します。</p> <p>program プロパティ:</p> <pre>@DLI { psb = psbName, callInterface = DLICallInterfaceKind.CBLTDLI, handleHardDLIErrors = yes }</pre> <p>program record declaration:</p> <pre>psbName psbRecordName ;</pre> <p>注: EGL では、PSB はレコード・パーツのステレオタイプです。</p>	<p>マイグレーション・ツールは、psb プロパティの値として常に <i>psb</i> を使用します。これにより、PSB で変数を参照する必要がある関数が、<i>psb</i> で変数を修飾できるようになります。</p> <p>マイグレーション・ツールは、IMS または DL/I プログラムが VAGen の振る舞いを保存できるように、常に次のプロパティをインクルードします。</p> <ul style="list-style-type: none"> callInterface handleHardDLIErrors <p>マイグレーション・ツールは、<i>psb</i> という変数の宣言をインクルードし、必要な名前変更を行った後で、<i>psbRecordName</i> を VAGen PSB パーツの名前に設定します。</p>
先頭マップ (仕様)	<p>inputForm (プログラム・プロパティ)</p>	特別な考慮事項なし。
先頭 UI レコード (仕様)	<p>inputUIRecord (プログラム・プロパティ)</p>	特別な考慮事項なし。

表 108. プログラム - プログラム仕様、プロパティ、テーブル、および追加レコード・リスト (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
マップ・グループ (仕様)	<p>use <i>FormGroup.formName</i> {, <i>FormGroup.formName</i> }</p> <p>注: use <i>FormGroup</i> も許可されています。この場合、<i>FormGroup</i> 内のすべての書式はプログラム内の非修飾の変数名を解決するために使用されます。</p>	<p>マイグレーション・ツールは次のマップをインクルードします。</p> <ul style="list-style-type: none"> • CONVERSE、DISPLAY、および CLOSE I/O オプションで 사용되는マップ。 • マップ・ステートメントのある XFER で使用されるマップ。 • プログラムの呼び出し先パラメーター・リストでリストされているマップ。 • プログラムの最初のマップとして指定されているマップ。
ヘルプ・マップ・グループ (仕様)	use <i>FormGroup</i> { helpGroup=yes } (use 宣言)	特別な考慮事項なし。
メッセージ・テーブル接頭部 (プログラム・プロパティ)	msgTablePrefix (プログラム・プロパティ)	特別な考慮事項なし。
暗黙データ項目の許可 (プログラム・プロパティ)	サポートされていません。	マイグレーション・ツールは、暗黙定義を自動的に作成しません。詳細と起こりうる問題については、107 ページの『プログラム内の暗黙データ項目』を参照してください。
<p>キー割り当て:</p> <ul style="list-style-type: none"> • ヘルプ・キー (1 つのキー) • バイパス・キー (5 つまでのキー) • F1-12 = F13-24 <p>(プログラム・プロパティ)</p> <p>注: キー割り当ては、プログラムに対して 1 回指定され、マップ・グループとヘルプ・マップ・グループの両方に適用されます。</p>	<p>EGL のキー割り当ての例:</p> <pre>use <i>FormGroup</i> { [helpGroup = yes,] helpKey = <i>pfNumber</i>, validationBypassKeys = [<i>pfNumberList</i>], pfKeyEquate = yes no } ;</pre> <p>(プログラムの <i>FormGroup</i> およびヘルプ <i>FormGroup</i> の use 宣言プロパティ)</p> <p>注:</p> <ul style="list-style-type: none"> • validationBypassKeys リストの値は、コンマで区切る必要があります。 • プログラムのヘルプ <i>FormGroup</i> の use 宣言で、validationBypassKeys プロパティが指定されていません。 	<p>マイグレーション・ツールは、<i>FormGroup</i> とヘルプ <i>FormGroup</i> の両方に関する use 宣言ステートメントに、キー割り当て情報に相当する EGL ステートメントを組み込みます。マイグレーション・ツールは、ヘルプ <i>FormGroup</i> の use 宣言の validationBypassKeys プロパティを省略します。</p>

表 108. プログラム - プログラム仕様、プロパティ、テーブル、および追加レコード・リスト (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>テーブルおよび追加レコード:</p> <ul style="list-style-type: none"> レコード <p>注:</p> <ul style="list-style-type: none"> 再定義情報は、プログラムではなく VAGen 再定義レコードに格納されます。 入出力オブジェクトとして使用されるレコードは、テーブルおよび追加レコードのリストには組み込まれません。 	<p>EGL 追加レコードの例:</p> <pre>recordName recordName [{ redefines = otherRecord}];</pre> <p>(データ宣言)</p> <p>注:</p> <ul style="list-style-type: none"> そのレコードが、別のレコードと同じ物理ストレージに関して異なるレコード・レイアウトを指定している場合は、プログラムでレコードを宣言するときに redefines プロパティを指定する必要があります。 データ宣言は、プログラム内で使用されるすべてのレコード (入出力レコードを含む) に必要です。 	<p>マイグレーション・ツールは、型定義と同じレコード名を常に使用します。</p> <p>VAGen レコードが再定義レコードとしてプログラム内で使用されている場合、マイグレーション・ツールは、データ宣言ステートメントに redefines プロパティを組み込みます。詳細と起こりうる問題については、86 ページの『再定義レコード』を参照してください。</p> <p>さらにマイグレーション・ツールは、プログラムによって入出力オブジェクトとして使用されているすべてのレコードにデータ宣言を組み込みます。</p> <p>マイグレーション・ツールは、プログラムによって入出力オブジェクトとして使用される MQ メッセージ・レコードの属性として指定されたレコードに、データ宣言を組み込みます。</p> <p>プログラムで PSB が指定された場合、I/O オブジェクトとして使用されるか、または I/O オブジェクトへの階層パスで使用されるすべての DL/I セグメントに関して、マイグレーション・ツールはデータ宣言をインクルードします。</p> <p>Web トランザクション・プログラムの場合、マイグレーション・ツールは、CONVERSE I/O ステートメントまたは XFER ステートメントで参照された UI レコードに関して、データ宣言をインクルードします。</p>

表 108. プログラム - プログラム仕様、プロパティ、テーブル、および追加レコード・リスト (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>テーブルおよび追加レコード:</p> <ul style="list-style-type: none"> • テーブル • それぞれのテーブルごとに、Keep After Use を指定できます。 	<p>EGL use 宣言の例を次に示します。</p> <pre>use tableName [{deleteAfterUse = yes}];</pre> <p>(use 宣言)</p>	<p>マイグレーション・ツールは、テーブルおよび追加レコードのリストにあるテーブルを use 宣言に変換します。</p> <p>deleteAfterUse プロパティは、VAGen の Keep After Use と反対の意味を持ちます。マイグレーション・ツールは、yes と no を反転します。</p> <p>マイグレーション設定で「テーブルの deleteAfterUse を組み込まない」を選択すると、マイグレーション・ツールは自動的に deleteAfterUse プロパティを省略し、影響を受けるプログラムおよびテーブルに関して警告メッセージを出します。</p>

表 109. プログラム - main 関数とフロー・ステートメント

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VAGen プログラムは、VAGen 構造ダイアグラムの最上位関数として main 関数を指定します。その他の関数はすべて、構造ダイアグラムを展開したときにのみ表示されます。</p> <p>それぞれの main 関数には、フロー・ステートメントが含まれていることがあります。これらのステートメントは構造ダイアグラムには表示されませんが、ダイアグラムからアクセス可能です。</p>	<p>EGL プログラムは、1 つの main 関数のみ指定します。この関数の名前は、常に main です。</p> <p>フロー・ステートメントは存在しません。</p> <p>次の例は EGL main 関数の構文を示しています。</p> <pre>function main () { functionLabel: functionName() ; [{ functionFlowStatements }] } end // end main</pre>	<p>マイグレーション・ツールは、EGL の main 関数を作成します。ツールは、VAGen のそれぞれの main 関数ごとに、EGL main 関数に次の情報を組み込みます。</p> <ul style="list-style-type: none"> • <i>functionLabel</i>。これにより、VAGen の main 関数を EGL の exit stack functionLabel ステートメント内で参照できます。ツールは、常に <i>functionLabel</i> を <i>functionName</i> に設定します。 • VAGen の main 関数を呼び出すための関数呼び出しステートメント。 • VAGen の main 関数のフロー・ステートメント (存在する場合)。 <p>フロー・ステートメントのマイグレーションについて詳しくは、以下のセクションを参照してください。</p> <ul style="list-style-type: none"> • 386 ページの『ステートメント』 • 404 ページの『EZE ワード』 • 415 ページの『サービス・ルーチン』

関数

次の表は、VAGen 演算部を EGL 関数と比較したもので、マイグレーション・ツールによる変換の処理方法を示しています。

関数に関するセクションは、次の表で構成されています。

- 関数 - 一般構文、説明、パラメーター、戻り値、およびローカル・ストレージ (365 ページの表 110)
- 関数 - EXECUTE 入出力オプション (367 ページの表 111)
- 関数 - マップと UI レコードの入出力オプション (368 ページの表 112)
- 関数 - ファイルまたはデータベースの入出力 - 一般情報と入出力エラー・ルーチン (368 ページの表 113)
- 関数 - シリアル・レコード、索引付きレコード、相対レコード、およびメッセージ・キュー・レコードの入出力オプション (369 ページの表 114)
- 関数 - Execution time statement build を使用しないデフォルト (未変更) の SQL ステートメントの入出力オプション (370 ページの表 115)
- 関数 - Execution time statement build を使用しない変更済み SQL ステートメントの入出力オプション (373 ページの表 116)
- 関数 - Execution time statement build を使用する SQL ステートメントの入出力オプション (377 ページの表 117)
- 関数 - デフォルト (未変更) の DL/I ステートメントの入出力オプション (380 ページの表 118)
- 関数 - 変更済み DL/I ステートメントのセグメント検索指数 (382 ページの表 119)

表 110. 関数 - 一般構文、説明、パラメーター、戻り値、およびローカル・ストレージ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>演算部には、次の情報を含めることができます。</p> <ul style="list-style-type: none"> 関数名 関数仮パラメーター 関数からの戻り値 関数ローカル・ストレージ プロパティ: <ul style="list-style-type: none"> エラー・ルーチン 説明 入出力オプション 入出力オブジェクト SQL ステートメント DL/I 呼び出し 入出力オプションの前のステートメント 入出力オプションの後のステートメント <p>注: 入出力オプションは、1 つの関数に 1 つのみです。</p>	<p>関数には、次の情報を含めることができます。</p> <ul style="list-style-type: none"> functionName functionParameterList returnItemType dataDeclarations 入出力ステートメントの前のステートメント 入出力ステートメント 入出力ステートメントの後のステートメント <p>以下の例に、マイグレーション・ツールによって作成される関数の形式を示します。</p> <pre>// Description Function functionName (functionParamterList) [returns(returnItemType)] [dataDeclarations] [beforeStatements] [IOStatement] [afterStatements] end // end functionName</pre> <p>注:</p> <ul style="list-style-type: none"> EGL <i>IOStatement</i> の作成には、VAGen の入出力オプション、入出力オブジェクト、エラー・ルーチン、SQL ステートメント、および DL/I 呼び出しが使用されます。 1 つの関数に、複数の <i>IOStatement</i> を使用することが可能です。 	<p>マイグレーション・ツールは、すべての VAGen 演算部を EGL スタンドアロン演算部に変換します。</p> <p>説明、関数からの戻り値、関数仮パラメーター、および関数ローカル・ストレージをマイグレーション・ツールが処理する方法については、この表の後続の行を参照してください。</p> <p>前後のステートメントのマイグレーションについて詳しくは、次のセクションを参照してください。</p> <ul style="list-style-type: none"> ステートメントについては、386 ページの『ステートメント』を参照。 EZE ワードについては、404 ページの『EZE ワード』を参照。 サービス・ルーチンについては、415 ページの『サービス・ルーチン』を参照。 <p>入出力オプションとエラー・ルーチンについて詳しくは、次の表を参照してください。</p> <ul style="list-style-type: none"> EXECUTE 入出力オプションについては、367 ページの表 111 を参照。 マップと UI レコードの入出力オプションについては、368 ページの表 112 を参照。 ファイルおよびデータベース入出力の一般情報については、368 ページの表 113 を参照。 シリアル・レコード、索引付きレコード、相対レコード、およびメッセージ・キュー・レコードの入出力オプションについては、369 ページの表 114 を参照。 <p>SQL ステートメントについて詳しくは、このセクションにある次の表を参照してください。</p> <ul style="list-style-type: none"> 未変更の入出力ステートメントの入出力オプションについては、370 ページの表 115 を参照。 Execution time statement build を使用しない変更済みの SQL ステートメントの入出力オプションについては、373 ページの表 116 を参照。 Execution time statement build を使用する SQL ステートメントの入出力オプションについては、377 ページの表 117 を参照。 <p>DL/I 呼び出しについて詳しくは、このセクションにある次の表を参照してください。</p> <ul style="list-style-type: none"> デフォルト (未変更) の DL/I ステートメントの入出力オプションについては、380 ページの表 118 を参照。 変更済み DL/I ステートメントのセグメント検索索引数については、382 ページの表 119 を参照。
説明	適用不可	マイグレーション・ツールは、関数の説明をコメントに変換し、関数定義の前に配置します。

表 110. 関数 - 一般構文、説明、パラメーター、戻り値、およびローカル・ストレージ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>関数仮パラメーター:</p> <ul style="list-style-type: none"> 関数仮パラメーターは、専用のウィンドウに入力されます。 関数仮パラメーターとして使用される項目は、共用の場合も非共用の場合もあります。非共用項目の定義は、関数に格納されます。 	<p>関数仮パラメーター:</p> <ul style="list-style-type: none"> パラメーターはコンマで区切る必要があります。 それぞれのパラメーターに型情報が指定されます。 オプションで、それぞれのパラメーターにパラメーター型情報が指定されます。 <p>以下の例に、関数仮パラメーターのフォーマットを示します。</p> <pre>(parameterName typeInfo [parameterType] { , parameterName typeInfo [parameterType] })</pre> <p>以下に、関数仮パラメーターの具体例を示します。</p> <pre>(parmSharedItem parmSharedItem field, parmNonSharedItem char(10) sqlNullable, parmRecord parmRecord)</pre>	<p>マイグレーション・ツールは、<i>typeInfo</i> を次のように設定します。</p> <ul style="list-style-type: none"> レコードの場合、<i>typeInfo</i> は同じレコード名を指定した型定義です。 項目の型が VAGen Any* 型のいずれかである場合、<i>typeInfo</i> は対応する EGL 特殊項目型です。 項目が共用データ項目の場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> マイグレーション構文設定「選択済み共用データ項目をプリミティブ項目定義に変換」を選択した場合に、データ項目パーツが使用可能であれば、共用項目はマイグレーション・ツールによって EGL 関数仮パラメーターに変換されます。このパラメーターはデータ項目パーツに対して指定された型、長さ、および小数部に基づいたプリミティブ定義を使用して宣言されます。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。 マイグレーション構文設定「選択済み共用データ項目をプリミティブ項目定義に変換」を選択解除した場合、またはデータ項目パーツが使用不可である場合は、共用項目はマイグレーション・ツールによって EGL 関数仮パラメーターに変換されます。この変数は型定義を使用して宣言されます。マイグレーション用の型定義は、常に項目名と同じです。 <p>項目が非共用データ項目の場合、<i>typeInfo</i> は項目の型、長さ、および小数部に基づいてマイグレーションされ、306 ページの表 75 で説明した規則に準拠します。</p>
<p>関数仮パラメーター:</p> <ul style="list-style-type: none"> 関数仮パラメーターの型: <ul style="list-style-type: none"> レコード 項目 マップ項目 SQL 項目 	<p>関数仮パラメーター:</p> <ul style="list-style-type: none"> 関数仮パラメーターの型: <ul style="list-style-type: none"> 適用不可 適用不可 field sqlNullable 	
<p>関数仮パラメーター:</p> <ul style="list-style-type: none"> 特殊項目型、長さの指定なし: <ul style="list-style-type: none"> AnyChar AnyDBCS AnyMix AnyHex AnyUnicode AnyNumeric 	<p>関数仮パラメーター:</p> <ul style="list-style-type: none"> 特殊項目型、長さの指定なし: <ul style="list-style-type: none"> CHAR DBCHAR MBCHAR HEX UNICODE number 	
<p>関数からの戻り値:</p> <ul style="list-style-type: none"> データ型 長さ 小数部 説明 	<p>EGL の戻り値:</p> <ul style="list-style-type: none"> 次に、returns ステートメントのフォーマットの例を示します。 <pre>returns(returnType) // Description</pre>	<p>関数に戻り値がある場合、マイグレーション・ツールは 306 ページの表 75 で説明した規則に従って、データ型、長さ、および小数部をマイグレーションします。</p>

表 110. 関数 - 一般構文、説明、パラメーター、戻り値、およびローカル・ストレージ (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>関数ローカル・ストレージ:</p> <ul style="list-style-type: none"> 関数ローカル・ストレージは、専用のウィンドウに入力されます。 関数ローカル・ストレージとして使用される項目は、共用の場合も非共用の場合もあります。非共用項目の定義は、関数に格納されます。 	<p>関数の変数宣言:</p> <ul style="list-style-type: none"> 関数の変数宣言には、変数名とそれに関連した型情報が含まれている必要があります。 以下の例に、関数の変数宣言のフォーマットを示します。 <pre>// Function Declarations variableName typeInfo ; { variableName typeInfo ; }</pre>	<p>マイグレーション・ツールは、<i>typeInfo</i> を次のように設定します。</p> <ul style="list-style-type: none"> レコードの場合、<i>typeInfo</i> は同じレコード名を指定した型定義です。 項目が共用データ項目の場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> マイグレーション構文設定「選択済み共用データ項目をプリミティブ項目定義に変換」を選択した場合に、データ項目パーツが使用可能であれば、共用項目はマイグレーション・ツールによって EGL 変数に変換されます。この変数はデータ項目パーツに関して指定された型、長さ、および小数部に基づいたプリミティブ定義を使用して宣言されます。型、長さ、および小数部の情報のマイグレーションは、306 ページの表 75 の説明と同じです。 マイグレーション構文設定「選択済み共用データ項目をプリミティブ項目定義に変換」を選択解除した場合、またはデータ項目パーツが使用不可の場合は、共用項目はマイグレーション・ツールによって EGL 変数に変換されます。この変数は、型定義を使用して宣言されます。マイグレーション用の型定義は、常に項目名と同じです。 項目が非共用データ項目の場合、<i>typeInfo</i> は項目の型、長さ、および小数部に基づいてマイグレーションされ、306 ページの表 75 で説明した規則に準拠します。
<p>関数ローカル・ストレージ:</p> <ul style="list-style-type: none"> 関数ローカル・ストレージのタイプ: <ul style="list-style-type: none"> レコード 項目 	<p>関数ローカル・ストレージ:</p> <ul style="list-style-type: none"> 関数ローカル・ストレージのタイプ: <ul style="list-style-type: none"> 適用不可 適用不可 	

表 111. 関数 - EXECUTE 入出力オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: なし 入出力オプション: EXECUTE 	<p>同等なステートメントなし。</p>	<p>マイグレーション・ツールは、EXECUTE 入出力オプションを除去します。</p>

表 112. 関数 - マップと UI レコードの入出力オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: mapName 入出力オプション: DISPLAY <p>注: DISPLAY は、表示マップとプリンター・マップの両方に使用されます。</p>	<p>テキスト書式を表示するには、display ステートメントを使用します。印刷書式を印刷するには、print ステートメントを使用します。</p> <p>以下の例は、display ステートメントおよび print ステートメントを示します。</p> <pre>display mapName; print mapName;</pre> <p>注: VisualAge Generator 互換モードでは、display printForm は print printForm と同様に扱われます。</p>	<p>マイグレーション・ツールは、マップのタイプに基づいて display ステートメントまたは print ステートメントへの変換を行います。詳細と起こりうる問題については、113 ページの『マップの DISPLAY 入出力オプション』を参照してください。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: mapName 入出力オプション: CONVERSE 	<p>converse ステートメントを使用します。</p> <p>以下に、converse ステートメントの例を示します。</p> <pre>converse mapName;</pre>	<p>特別な考慮事項なし。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: UIRecordName 入出力オプション: CONVERSE 	<p>converse ステートメントを使用します。</p> <p>以下に、converse ステートメントの例を示します。</p> <pre>converse UIRecordName;</pre>	<p>特別な考慮事項なし。</p>

表 113. 関数 - ファイルまたはデータベースの入出力 - 一般情報と入出力エラー・ルーチン

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VAGen レコード入出力:</p> <ul style="list-style-type: none"> 入出力オプション 入出力オブジェクト (常にレコード) 入出力エラー・ルーチン (オプション) <p>注: レコードは、シリアル・レコード、索引付きレコード、相対レコード、メッセージ・キュー・レコード、SQL 行レコード、DL/I セグメント・レコードのいずれかです。</p>	<p>EGL レコード入出力:</p> <ul style="list-style-type: none"> 入出力ステートメント レコード名 try ... onException ステートメント (エラー・ルーチン名を指定) (オプション) <p>入出力エラー・ルーチンを指定する場合は、ステートメントを try ブロックで囲みます。以下の例に、エラー・ルーチンを含むファイルまたはデータベースの入出力を示します。</p> <pre>try add recordName ; [onException error-routine ;] end</pre>	<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> VAGen の入出力オプションを、対応する EGL の入出力ステートメントに変更します。 VAGen のエラー・ルーチンがある場合、try ... onException ステートメントを組み込みます。

表 113. 関数 - ファイルまたはデータベースの入出力 - 一般情報と入出力エラー・ルーチン (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>ファイルまたはデータベースの入出力を行う関数の場合、エラー・ルーチンはオプションです。</p> <p>注: ソフト・エラーが発生した場合、または EZEFECE = 1 の場合に、エラー・ルーチンが呼び出されます。</p>	<p>レコードの入出力を行う関数の場合、エラー・ルーチンはオプションです。以下の例に、エラー・ルーチンを含まない入出力を示します。</p> <pre>add recordName;</pre> <p>以下の例に、エラー・ルーチンを含む入出力を示します。</p> <pre>try add recordName ; onException error-routine ; end</pre> <p>注: ソフト・エラーが発生した場合、または <code>vgVar.handleHardIOErrors</code> が 1 に設定されている場合に、<code>onException</code> ステートメントが呼び出されます。関数が DL/I 入出力を行う場合は、<code>vgVar.handleHardDLIErrors</code> が 1 に設定されている場合にも <code>onException</code> ステートメントが呼び出されます。</p>	<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> エラー・ルーチンが指定されていない場合、ツールは <code>try ... onException</code> ステートメントを組み込みません。 エラー・ルーチンが指定されている場合、ツールは <code>try</code> ブロックを組み込みます。 マイグレーション・ツールは、VAGen のエラー・ルーチン名に基づいて <code>onException</code> ステートメントへの変換を行います。マイグレーション・ツールがプログラムをマイグレーションする際に、VAGen の main 関数名は、main 関数ラベルと main 関数呼び出しステートメントの両方に常にマイグレーションされます。したがって、関数の入出力エラー・ルーチンをマイグレーションする際、<code>mainFunctionLabel</code> は <code>mainFunctionName</code> と常に同じです。 <p>関数名であるエラー・ルーチンのマイグレーションに関しては、特別な考慮事項が適用されます。詳細と起こりうる問題については、115 ページの『入出力エラー・ルーチン』を参照してください。</p> <p>Execution time statement build を指定する SQL 入出力関数に関しても、特別な考慮事項が適用されます。詳しくは、123 ページの『SQL 入出力と Execution time statement build』を参照してください。</p>
<p>エラー・ルーチンの値:</p> <p>EZECLOS EZEFL0 EZERTN mainFunctionName nonmainFunctionName</p>	<p><code>onException</code> ブロック・ステートメント:</p> <pre>onException exit program; onException exit stack; Omit the onException statement. onException exit stack mainFunctionLabel; onException nonmainFunctionName();</pre>	

表 114. 関数 - シリアル・レコード、索引付きレコード、相対レコード、およびメッセージ・キュー・レコードの入出力オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: ADD 	<p><code>add</code> ステートメントを使用します。次に例を示します。</p> <pre>add recordName;</pre>	<p>特別な考慮事項なし。</p>

表 114. 関数 - シリアル・レコード、索引付きレコード、相対レコード、およびメッセージ・キュー・レコードの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: SCAN 	get next ステートメントを使用します。次に例を示します。 get next recordName;	特別な考慮事項なし。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: SCANBACK 	get previous ステートメントを使用します。次に例を示します。 get previous recordName;	特別な考慮事項なし。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: CLOSE 	close ステートメントを使用します。次に例を示します。 close recordName;	特別な考慮事項なし。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: INQUIRY 	get ステートメントを使用します。次に例を示します。 get recordName;	特別な考慮事項なし。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: UPDATE 	get forUpdate ステートメントを使用します。次に例を示します。 get recordName forUpdate ;	特別な考慮事項なし。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: DELETE 	delete ステートメントを使用します。次に例を示します。 delete recordName;	特別な考慮事項なし。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: REPLACE 	replace ステートメントを使用します。次に例を示します。 replace recordName;	特別な考慮事項なし。

表 115. 関数 - *Execution time statement build* を使用しないデフォルト (未変更) の SQL ステートメントの入出力オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: ADD 	add ステートメントを使用します。次に例を示します。 add recordName;	特別な考慮事項なし。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: SCAN 	get next ステートメントを使用します。次に例を示します。 get next recordName;	特別な考慮事項なし。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: CLOSE 	close ステートメントを使用します。次に例を示します。 close recordName;	特別な考慮事項なし。

表 115. 関数 - *Execution time statement build* を使用しないデフォルト (未変更) の SQL ステートメントの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: INQUIRY (Single row select を使用する場合、および使用しない場合)	<p>get ステートメントを使用します。single row select を実行する場合は、singleRow も使用します。以下の例に、single row select を使用しない get ステートメントを示します。</p> <p>get recordName;</p> <p>以下の例に、single row select を示します。</p> <p>get recordName singleRow;</p>	<p>Single row select が VisualAge Generator 内で指定されている場合、マイグレーション・ツールは EGL の singleRow オプションを組み込みます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: UPDATE 	<p>get forUpdate ステートメントを使用します。次に例を示します。</p> <p>get recordName forUpdate resultSetID;</p>	<p>マイグレーション・ツールは、SQL レコードの UPDATE 入出力オプションをマイグレーションする際に、<i>resultSetID</i> を常に組み込みます。ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>レコードが SQL または非 SQL のどちらであるかマイグレーション・ツールが判別できない場合は、特別な考慮事項が適用されます。詳細と起こりうる問題については、126 ページの『複数の UPDATE 関数または SETUPD 関数を使用する SQL 入出力』を参照してください。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: DELETE 	<p>delete ステートメントを使用します。次に例を示します。</p> <p>delete recordName;</p>	<p>特別な考慮事項なし。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: REPLACE (UPDATE/SETUPD functionName を使用する場合、または使用しない場合)	<p>replace ステートメントを使用します。次に、いくつかの例を示します。</p> <p>replace recordName;</p> <p>replace recordName from resultSetID;</p>	<p>VisualAge Generator で、UPDATE/SETUPD 関数名が組み込まれていた場合、マイグレーション・ツールは <i>resultSetID</i> を組み込み、<i>resultSetID</i> を UPDATE/SETUPD 関数名に設定して、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p>

表 115. 関数 - *Execution time statement build* を使用しないデフォルト (未変更) の *SQL* ステートメントの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: SETINQ <p>(Declare cursor with hold を使用する場合、および使用しない場合)</p>	<p>open ステートメントを使用します。Declare cursor with hold を使用する場合は、hold オプションも使用します。次に、両タイプのステートメントの例を示します。</p> <pre>open resultSetID for recordName; open resultSetID hold for recordName;</pre>	<p>マイグレーション・ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>Declare cursor with hold が VisualAge Generator で選択されていた場合、マイグレーション・ツールは EGL の hold オプションを組み込みます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: SETUPD <p>(Declare cursor with hold を使用する場合、および使用しない場合)</p>	<p>open forUpdate ステートメントを使用します。Declare cursor with hold を使用する場合は、hold オプションも使用します。次に、両タイプのステートメントの例を示します。</p> <pre>open resultSetID forUpdate for recordName; open resultSetID hold forUpdate for recordName;</pre>	<p>ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>Declare cursor with hold が VisualAge Generator で選択されていた場合、マイグレーション・ツールは EGL の hold オプションを組み込みます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: SQLEXEC <p>Model SQL Statement を使用する場合</p> <p>注:</p> <ul style="list-style-type: none"> この形式の SQLEXEC には、SQL レコード名が含まれています。 モデル・タイプの値は、次のとおりです。 <ul style="list-style-type: none"> None Update Delete モデル・タイプがなし (None) の場合、VisualAge Generator は入出力を行いません。それでも入出力エラー・ルーチンは生成機能によって処理されますが、実行時にエラーは発生しません。 	<p>execute ステートメントを使用します。次に例を示します。</p> <pre>execute modelType for recordName;</pre> <p>注: <i>modelType</i> は、update または delete のどちらかです。</p>	<p>マイグレーション・ツールは、VAGen の Model SQL Statement の値に基づいて EGL の <i>modelType</i> を設定します。</p> <p>VAGen の Model SQL Statement が None の場合、VAGen の入出力ステートメントは効力を持たないので、マイグレーション・ツールは入出力ステートメントを省略します。マイグレーション・ツールは、関数の入出力エラー・ルーチンに基づいて try ... onException ステートメントを組み込みます。</p>

表 116. 関数 - *Execution time statement build* を使用しない変更済み SQL ステートメントの入出力オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>変更済み SQL ステートメントに関する一般情報:</p> <ul style="list-style-type: none"> VisualAge Generator は、テストおよび生成時に SQL 行レコードから table 文節を作成します。次に示す table 文節が使用されます。 <ul style="list-style-type: none"> insert into <i>sqlTableName</i> (ADD 入出力オプションの場合) from <i>sqlTableName</i> <i>sqlTableLabel</i> (INQUIRY、UPDATE、SETINQ、および SETUPD の各入出力オプションの場合) update <i>sqlTableName</i> (REPLACE 入出力オプションの場合) 関数の最終変更時点によっては、他の SQL 文節が関数定義に保管されない場合があります。SQL 文節が保管されていない場合、VisualAge Generator は、入出力オブジェクトのレコード定義に基づいてデフォルト文節を作成します。 <i>!itemColumnName</i> 変数が使用できます。この変数は、SQL 行レコード内の項目の名前を指定します。テストまたは生成時に、VisualAge Generator は対応する SQL 列名への置換を行います。 SQL 文節は、SQL 構文で作成されます。 	<p>変更済み SQL ステートメントに関する一般情報:</p> <ul style="list-style-type: none"> SQL 文節を変更する必要がある場合、EGL は文節すべての明示的な指定を必要とします。SQL ステートメントに table 文節を明示的に組み込む必要があります。次に示す table 文節が使用されます。 <ul style="list-style-type: none"> insert into <i>sqlTableName</i> (add ステートメントの場合) from <i>sqlTableName</i> <i>sqlTableLabel</i> (get と open ステートメントの場合) update <i>sqlTableName</i> (replace ステートメントの場合) SQL 文節が指定されている場合、EGL は文節すべての明示的な指定を必要とします。必要な SQL 文節は、入出力のタイプによって異なります。 EGL は、SQL 列名が SQL ステートメントに明示的に含まれていることを必要とします。<i>!itemColumnName</i> 変数はサポートされません。 SQL 文節は、SQL 構文で作成されます。 	<p>マイグレーション・ツールは、SQL 行レコードからの表と表ラベルを使用して、EGL 入出力ステートメントの tables 文節を作成します。表名と表名ホスト変数の両方が、EGL 入出力ステートメントの table 文節に組み込まれます。</p> <p>必要な SQL 文節が関数定義に保管されていない場合、マイグレーション・ツールは VisualAge Generator の場合と同じように、レコード定義に基づいてデフォルト文節を作成します。</p> <p>マイグレーション・ツールは、<i>!itemColumnName</i> 変数を対応する SQL 列名に変換します。</p> <p>マイグレーション・ツールは、VAGen のコメント (/*) を SQL のコメント (一) に変換します。</p> <p>SQL レコードとその代替仕様レコード (存在する場合) がマイグレーション時に使用できない場合は、特別な考慮事項が適用されます。詳細と起こりうる問題については、次のセクションを参照してください。</p> <ul style="list-style-type: none"> 116 ページの『SQL 入出力ステートメント』 119 ページの『SQL 入出力と必須 SQL 文節の欠落』 125 ページの『SQL 入出力と <i>!itemColumnName</i>』
<ul style="list-style-type: none"> 入出力オブジェクト: <i>recordName</i> 入出力オプション: ADD <p>変更可能な文節:</p> <ul style="list-style-type: none"> INSERTCOLNAMES VALUES 	<p>add ステートメントを使用します。次に例を示します。</p> <pre>add recordName with #sql{ insert into sqlTablename (columnName1, columnName2) values (valueInfo1, valueInfo2) };</pre>	<p>マイグレーション・ツールは、レコード定義にある表名に基づいて INSERT INTO 文節を作成します。特別な考慮事項が適用されます。詳細と起こりうる問題については、116 ページの『SQL 入出力ステートメント』を参照してください。</p>

表 116. 関数 - *Execution time statement build* を使用しない変更済み SQL ステートメントの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: INQUIRY <p>(Single row select を使用する場合、および使用しない場合)</p> <p>変更可能な文節:</p> <ul style="list-style-type: none"> SELECT INTO WHERE、GROUP BY、HAVING ORDER BY 	<p>get ステートメントを使用します。</p> <p>次に、single row select を使用する場合の例を示します。</p> <pre> get recordName singleRow with #sql{ select Name1, Name2, Age from sqlTable1 sqlLabel1, sqlTable2 sqlLabel2 where Name1 = :NameX order by Age } into nameA, nameB, myage; </pre>	<p>Single row select が VisualAge Generator 内で指定されている場合、マイグレーション・ツールは EGL の singleRow オプションを組み込みます。</p> <p>マイグレーション・ツールは、レコード定義にある表名と表ラベルに基づいて FROM 文節を作成します。特別な考慮事項が適用されます。詳細と起こりうる問題については、116 ページの『SQL 入出力ステートメント』を参照してください。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: UPDATE <p>変更可能な文節:</p> <ul style="list-style-type: none"> SELECT INTO WHERE FOR UPDATE OF 	<p>get forUpdate ステートメントを使用します。</p> <p>次に例を示します。</p> <pre> get recordName forUpdate resultsetID with #sql{ select Name1, Name2, Age from sqlTable1 sqlLabel1 where Name1 = :NameX for update of Name2, Age } into Name1, Name2, Age; </pre>	<p>マイグレーション・ツールは、SQL レコードの UPDATE 入出力オプションをマイグレーションする際に、<i>resultSetID</i> を常に組み込みます。ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>マイグレーション・ツールは、レコード定義にある表名と表ラベルに基づいて FROM 文節を作成します。特別な考慮事項が適用されます。詳細と起こりうる問題については、116 ページの『SQL 入出力ステートメント』を参照してください。</p>

表 116. 関数 - *Execution time statement build* を使用しない変更済み SQL ステートメントの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: REPLACE <p>(オプションで UPDATE/SETUPD functionName)</p> <p>指定できる文節:</p> <ul style="list-style-type: none"> SET 	<p>replace ステートメントを使用します。</p> <p>次に、replace ステートメントの例を示します。</p> <pre> replace recordName with #sql{ update sqlTableName set columnName1 = value1, columnName2 = value2 } from resultSetID;</pre>	<p>VisualAge Generator では、UPDATE/SETUPD 関数名が組み込まれていた場合、マイグレーション・ツールは from resultSetID 文節を組み込みます。マイグレーション・ツールは、<i>resultSetID</i> を UPDATE/SETUPD 関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>マイグレーション・ツールは、レコード定義にある表名に基づいて UPDATE 文節を作成します。特別な考慮事項が適用されます。詳細と起こりうる問題については、116 ページの『SQL 入出力ステートメント』を参照してください。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オブジェクト: SETINQ <p>(Declare cursor with hold を使用する場合、または使用しない場合)</p> <p>変更可能な文節:</p> <ul style="list-style-type: none"> SELECT INTO WHERE、GROUP BY、HAVING ORDER BY 	<p>open ステートメントを使用します。Declare cursor with hold を使用する場合は、hold オプションも使用します。</p> <p>以下にhold オプションを使用した open ステートメントの例を示します。</p> <pre> open resultSetID hold with #sql{ select Name1, Name2 from sqlTable1 sqlLabel1, sqlTable2 sqlLabel2 where Name1 > :Name2 order by Name1 } into Name1, Name2 for recordName;</pre>	<p>マイグレーション・ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>Declare cursor with hold が VisualAge Generator で選択されていた場合、マイグレーション・ツールは EGL の hold オプションを組み込みます。</p> <p>マイグレーション・ツールは、レコード定義にある表名と表ラベルに基づいて FROM 文節を作成します。特別な考慮事項が適用されます。詳細と起こりうる問題については、116 ページの『SQL 入出力ステートメント』を参照してください。</p>

表 116. 関数 - *Execution time statement build* を使用しない変更済み SQL ステートメントの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: record 入出力オプション: SETUPD <p>(Declare cursor with hold を使用する 場合、または使用しない場合)</p> <p>変更可能な文節:</p> <ul style="list-style-type: none"> SELECT INTO WHERE FOR UPDATE OF 	<p>open forUpdate ステートメントを使用します。次に、hold オプションの使用例を示します。</p> <pre> open resultSetID hold forUpdate with #sql{ select Column1, Column2 from sqlTable1 sqlLabel1 where Column1 > :Item1 for update of Column2 } into Item1, Item2 for recordName; </pre>	<p>マイグレーション・ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>Declare cursor with hold が VisualAge Generator で選択されていた場合、マイグレーション・ツールは EGL の hold オプションを組み込みます。</p> <p>マイグレーション・ツールは、レコード定義にある表名と表ラベルに基づいて FROM 文節を作成します。特別な考慮事項が適用されます。詳細と起こりうる問題については、116 ページの『SQL 入出力ステートメント』を参照してください。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: record 入出力オプション: SQLEXEC <p>Model SQL Statement を使用する 場合</p> <p>注:</p> <ul style="list-style-type: none"> この形式の SQLEXEC には、SQL レコード名がオプションに含まれています。 モデル・タイプでは、以下の値が有効です。 <ul style="list-style-type: none"> None Update Delete 	<p>execute ステートメントを使用します。次に、このステートメントの例を示します。</p> <pre> execute modelType #sql{ UPDATE mysqltable set Column1 = Column1 * 2 where Column2 = :Column2 } for recordName; </pre> <p>注: <i>modelType</i> の値には、Update と Delete があります。</p>	<p>マイグレーション・ツールは、以下の処理を行います。</p> <ul style="list-style-type: none"> SQLEXEC を execute ステートメントに変換します。 入出力オブジェクト (指定されている場合) を for 文節の <i>recordName</i> として使用します。 <p>マイグレーション・ツールは、VAGen の Model SQL Statement の値 (存在する場合) を、EGL の execute ステートメントのコメントとして組み込みます。</p> <p>マイグレーション・ツールは、VAGen の SQLEXEC 文節を EGL の SQL 文節にマイグレーションします。</p>

表 117. 関数 - *Execution time statement build* を使用する SQL ステートメントの入出力オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>Execution time statement build は、次の入出力オプションのみと組み合わせで使用できます。</p> <ul style="list-style-type: none"> • INQUIRY • UPDATE • SETINQ • SETUPD • SQLEXEC <p>注:</p> <ul style="list-style-type: none"> • Execution time statement build を指定すると、入出力ステートメントが実行されるたびに、SQL ステートメントが VisualAge Generator によって動的に準備されます。 • Execution time statement build は、未変更 (デフォルト) の SQL または変更済み SQL のどちらを使用する場合にも使用することができます。 	<p>EGL では、SQL ステートメントを動的に準備する必要が生じるごとに、EGL の prepare ステートメントを直接コーディングします。 prepare の後に、open、execute、または get ステートメントをコーディングする必要があります。以下の例に、Execution time statement build を使用する VAGen INQUIRY 入出力オプションに相当する EGL を示します。</p> <pre> prepare prepID from "sqlStatementString" for recordName; get recordName with prepID into itemList; </pre> <p>注:</p> <ul style="list-style-type: none"> • prepare ステートメント内の <i>sqlStatementString</i> は、SQL 表記で書かれた定数と変数を連結したストリングです。以下の例に、列名と変数の両方を使用する WHERE 文節を示します。 <pre> [other clauses] + " where columnName = " + itemName + " AND columnName2 = " + itemName2 + [other clauses] </pre> <ul style="list-style-type: none"> • この表で後に示す例には、二重引用符の外側に分割された変数は示されていません。 • SQL ステートメントは、prepare ステートメントの中で明示的に指定されている必要があります。 • VisualAge Generator でのエラー処理と同じエラー処理が行われるようにするには、追加の EGL ロジックが必要です。例えば、この表の次の 2 行を参照してください。 	<p>マイグレーション・ツールは、<i>prepID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>マイグレーション・ツールは、関数内の SQL 文節と、レコード定義にある表名および表名変数を使用して、<i>sqlStatementString</i> を作成します。マイグレーション・ツールは、<i>sqlStatementString</i> を次のように作成します。</p> <ul style="list-style-type: none"> • Execution time statement build が指定されていない場合と同様に、次のような処理をすべて行います。 <ul style="list-style-type: none"> – SQL 行レコードから得られる表名および表ラベルを使用して、EGL 入出力ステートメントの <i>tables</i> 文節を作成します。表名と表名ホスト変数の両方が、組み込まれます。 – レコード定義に基づいて、必要なデフォルト文節を作成します。 – <i>!itemColumnName</i> 変数を対応する SQL 列名に変換します。 – VAGen コメント (/*) を、prepare ステートメント内の EGL コメント (//) に変換します。 <p>その後、マイグレーション・ツールは次のような追加処理を行って、<i>sqlStatementString</i> を作成します。</p> <ul style="list-style-type: none"> • 定数、列名、および SQL 演算子を二重引用符で囲みます。 • 変数を二重引用符の外側に配置します。 • + ストリング連結記号を使用して、ストリングと変数を連結します。 <p>SQL レコードとその代替仕様レコード (存在する場合) がマイグレーション時に使用できない場合は、特別な考慮事項が適用されます。詳細と起こりうる問題については、次のセクションを参照してください。</p> <ul style="list-style-type: none"> • 116 ページの『SQL 入出力ステートメント』 • 119 ページの『SQL 入出力と必須 SQL 文節の欠落』 • 125 ページの『SQL 入出力と <i>!itemColumnName</i>』

表 117. 関数 - Execution time statement build を使用する SQL ステートメントの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>INQUIRY、UPDATE、SETINQ、および SETUPD 入出力オプションを指定した場合、エラー処理の目的で入出力ステートメント全体が 1 つのユニットとして扱われます。SQL PREPARE の段階でソフト・エラーが発生しても、入出力オプションに必要なその他の SQL ステートメントの処理は続行されます。</p>	<p>各入出力ステートメントは、エラー処理の目的で別々に扱われます。VisualAge Generator でのエラー処理と同じエラー処理が行われるようにするには、例えば以下のようにします。</p> <pre> try prepare prepID from "sqlStatementString" for recordName; end if (recordName not HardIOError) // continue on soft error try get recordName with prepID into itemList ; onException errorRoutine; end else // hard error on prepare errorRoutine; end </pre>	<p>VAGen 関数が入出力エラー・ルーチンを指定している場合、EGL の prepare ステートメントにソフト・エラーが発生しても処理が続行されるように、マイグレーション・ツールは EGL で特別なエラー処理ロジックを組み込みます。VAGen の入出力エラー・ルーチンと EGL の onException ステートメントの関係については、368 ページの表 113 を参照してください。</p> <p>VAGen 関数が入出力エラー・ルーチンを指定していない場合、マイグレーション・ツールは追加ロジックを組み込むことができないため、代わりに警告メッセージを出します。</p> <p>この表の残りの行では、エラー処理ロジックは示されず、prepare および get または open ステートメントのみが示されます。</p>
<p>SQLEXEC 入出力オプションを指定した場合、このステートメントは SQL EXECUTE IMMEDIATE として扱われ、1 つの SQL コマンドとして prepare、execute、および destroy を行います。ソフト・エラーが発生した場合、処理が続くかどうかは特定のエラーの SQL 処理に基づいて決まります。</p>	<p>execute ステートメントが後に続く prepare ステートメントが、最も近い EGL コードです。次に例を示します。</p> <pre> try prepare prepID from "sqlStatementString" for recordName; execute prepID for recordName ; onException errorRoutine; end </pre>	<p>VAGen 関数がデフォルトの SQL を使用し、レコードおよび Update または Delete モデル・オプションを指定している場合、マイグレーション・ツールはレコード定義を使用し、prepare ステートメントおよび execute ステートメントに対応するデフォルトの SQL 文節を作成します。</p> <p>VAGen 関数が変更済みの SQL を使用する場合、マイグレーション・ツールはその SQL を使用し、prepare および execute ステートメントを作成します。</p> <p>この表の残りの行では、エラー処理ロジックは示されず、prepare および execute ステートメントのみが示されます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: record 入出力オプション: INQUIRY <p>(Execution time statement build を使用する場合、Single row select はサポートされません。)</p> <p>変更可能な文節:</p> <ul style="list-style-type: none"> SELECT INTO WHERE、GROUP BY、HAVING ORDER BY 	<p>prepare ステートメントを使用し、その後に get ステートメントを使用します。次に例を示します。</p> <pre> prepare prepID from " select columnName " + ", columnName2 " + " from table1 t1 " + "[where whereClause]" + "[order by orderByClause]" for recordName; get recordName with prepID into itemList; </pre>	<p>特別な考慮事項なし。</p>

表 117. 関数 - Execution time statement build を使用する SQL ステートメントの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: record 入出力オプション: UPDATE <p>変更可能な文節:</p> <ul style="list-style-type: none"> SELECT INTO WHERE FOR UPDATE OF 	<p>prepare ステートメントを使用し、その後に get forUpdate ステートメントを使用します。次に例を示します。</p> <pre> prepare prepID from " select columnName " + ", columnName2 " + " from table1 t1 " + "[where whereClause]" + " for Update of columnList " for recordName; get recordName forUpdate resultSetID with prepID into itemList; </pre>	<p>マイグレーション・ツールは、SQL レコードの UPDATE 入出力オプションをマイグレーションする際に、<i>resultSetID</i> を常に組み込みます。ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: record 入出力オプション: SETINQ <p>(Declare cursor with hold を使用する 場合、または使用しない場合)</p> <p>変更可能な文節:</p> <ul style="list-style-type: none"> SELECT INTO WHERE、GROUP BY、HAVING ORDER BY 	<p>prepare ステートメントを使用し、その後に open ステートメントを使用します。次に例を示します。</p> <pre> prepare prepID from " select columnName " + ", columnName2 " + " from table1 t1 " + "[where whereClause]" + "[order by orderByClause]" for recordName; open resultSetID [hold] with prepID into itemList for recordName; </pre>	<p>マイグレーション・ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>Declare cursor with hold が VisualAge Generator で選択されていた場合、マイグレーション・ツールは EGL の hold オプションを組み込みます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: record 入出力オプション: SETUPD <p>(Declare cursor with hold を使用する 場合、または使用しない場合)</p> <p>変更可能な文節:</p> <ul style="list-style-type: none"> SELECT INTO WHERE FOR UPDATE OF 	<p>prepare ステートメントを使用し、その後に open forUpdate ステートメントを使用します。次に例を示します。</p> <pre> prepare prepID from " select columnName " + ", columnName2 " + " from table1 t1 " + "[where whereClause] " + " for update of columnList " for recordName; open resultSetID [hold] forUpdate with prepID into itemList for recordName; </pre>	<p>ツールは、<i>resultSetID</i> を関数名に設定し、その後にユーザー指定の接尾部を付けます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。</p> <p>Declare cursor with hold が VisualAge Generator で選択されていた場合、マイグレーション・ツールは EGL の hold オプションを組み込みます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: record 入出力オプション: SQLEXEC <p>Model SQL Statement を使用する 場合注:</p> <ul style="list-style-type: none"> この形式の SQLEXEC には、SQL レコード名がオプションで含まれています。 モデル・タイプでは、以下の値が有効です。 <ul style="list-style-type: none"> None Update Delete 	<p>prepare ステートメントを使用し、その後に execute ステートメントを使用します。次に例を示します。</p> <pre> prepare prepID from " grant " + group_privileges + " on " + table_name + " to " + userid [for recordName]; execute prepID [for recordName]; // model = type </pre>	<p>VAGen 関数がデフォルトの SQL を使用し、レコードおよび Update または Delete のモデル・オプションを指定している場合、マイグレーション・ツールはレコード定義を使用し、prepare ステートメントおよび execute ステートメントに対応するデフォルトの SQL 文節を作成します。ツールは、VAGen の Model SQL Statement の値 (存在する場合) を、EGL の execute ステートメントのコメントとして組み込みます。</p> <p>VAGen 関数が変更済みの SQL を使用する 場合、マイグレーション・ツールは、VAGen の SQLEXEC 文節を EGL の SQL 文節に変換します。</p>

表 118. 関数 - デフォルト (未変更) の DLI ステートメントの入出力オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: ADD PSB: psbName データベース ID: databaseName pcbNumber 	<p>add ステートメントを使用します。次に例を示します。</p> <pre>add recordName [usingPCB pcbInfo] ;</pre>	<p>VAGen 関数でデータベース ID を指定すると、マイグレーション・ツールは usingPCB キーワードを組み込みます。マイグレーション・ツールは、VAGen 関数に保管されている情報に基づいて、<i>pcbInfo</i> の値を設定します。</p> <ul style="list-style-type: none"> <i>databaseName</i> には、カスタマー指定の接尾部が続きます。接尾部は、ステージ 2 の「VAGen マイグレーションの設定」によって制御できます。 <i>pcbn</i> (<i>n</i> は、VisualAge Generator で指定した <i>pcbNumber</i>)。
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: SCAN PSB: psbName データベース ID: databaseName pcbNumber SCAN オプション: <ul style="list-style-type: none"> Scan update Scan parent 	<p>get next ステートメントを使用します。次に例を示します。</p> <pre>get next recordName [usingPCB pcbInfo] ;</pre> <p>以下の例に、Scan update と Scan parent の両方を示します。</p> <pre>get next inParent recordName forUpdate [usingPCB pcbInfo] ;</pre>	<p>VAGen 関数でデータベース ID を指定した場合は、ADD 入出力オプションについて説明されているように、マイグレーション・ツールによって usingPCB キーワードが組み込まれます。</p> <p>VAGen 関数で Scan parent オプションを指定した場合は、マイグレーション・ツールによって inParent キーワードが組み込まれます。</p> <p>VAGen 関数で Scan update オプションを指定した場合は、マイグレーション・ツールによって forUpdate キーワードが組み込まれます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: INQUIRY PSB: psbName データベース ID: databaseName pcbNumber 	<p>get ステートメントを使用します。次に例を示します。</p> <pre>get recordName [usingPCB pcbInfo] ;</pre>	<p>VAGen 関数でデータベース ID を指定した場合は、ADD 入出力オプションについて説明されているように、マイグレーション・ツールによって usingPCB キーワードが組み込まれます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: UPDATE PSB: psbName データベース ID: databaseName pcbNumber 	<p>get forUpdate ステートメントを使用します。次に例を示します。</p> <pre>get recordName forUpdate [usingPCB pcbInfo] ;</pre>	<p>VAGen 関数でデータベース ID を指定した場合は、ADD 入出力オプションについて説明されているように、マイグレーション・ツールによって usingPCB キーワードが組み込まれます。</p>

表 118. 関数 - デフォルト (未変更) の DLI ステートメントの入出力オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: DELETE PSB: psbName データベース ID: databaseName pcbNumber 	<p>delete ステートメントを使用します。次に例を示します。</p> <pre>delete recordName [usingPCB pcbInfo] ;</pre>	<p>VAGen 関数でデータベース ID を指定した場合は、ADD 入出力オプションについて説明されているように、マイグレーション・ツールによって usingPCB キーワードが組み込まれます。</p>
<ul style="list-style-type: none"> 入出力オブジェクト: recordName 入出力オプション: REPLACE PSB: psbName データベース ID: databaseName pcbNumber 	<p>replace ステートメントを使用します。次に例を示します。</p> <pre>replace recordName [usingPCB pcbInfo] ;</pre>	<p>VAGen 関数でデータベース ID を指定した場合は、ADD 入出力オプションについて説明されているように、マイグレーション・ツールによって usingPCB キーワードが組み込まれます。</p>

表 119. 関数 - 変更済み DL/I ステートメントのセグメント検索指数

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>変更済み DL/I 呼び出し (SSA を使用) で指定される一般情報:</p> <ul style="list-style-type: none"> 入出力オプション 入出力オブジェクト PSB データベース ID Scan update Scan parent 次の SSA のうちゼロ、1 つ、またはそれ以上が組み込まれます。 <ul style="list-style-type: none"> セグメント名 コマンド・コード ブール演算子 セグメント・フィールド 関係演算子 比較値項目 <p>注:</p> <ul style="list-style-type: none"> SSA は、専用の DL/I Call Editor に入ります。 VisualAge Generator は、生成時に SSA のフォーマットを設定します。 	<p>変更済み DL/I 呼び出し (SSA を使用) の一般的なフォーマットでは、SSA は次のように中括弧の中に入ります。</p> <pre>with #dli { dliFunction SSAList } ;</pre> <p>例えば、次のようになります。</p> <pre>ioOptionWithModifiers recordNameList [usingPCB pcbInfo] with #dli { dliFunction segmentName1*cmdCodes (segmentFieldA relationalOperatorA :comparisonValueItemA booleanOperator segmentFieldB relationalOperatorB :comparisonValueItemB) segmentName2*cmdCodes (segmentFieldC relationalOperatorC :comparisonValueItemC) };</pre> <p>注:</p> <ul style="list-style-type: none"> SSA はテキスト・エディターに入ります。 EGL は、DL/I のフォーマット規則に従って、生成時に SSA のフォーマットを設定します。 	<p>マイグレーション・ツールは、入出力オブジェクトと SSA 用に指定されるコマンド・コードとの組み合わせに基づいて、<i>recordNameList</i> を作成します。</p>
<p>変更済み DL/I 呼び出し (SSA はすべて削除)。</p> <p>注: この手法は、データベース内のすべてのセグメントをスキャンする場合に使用されます。</p>	<p>変更済み DL/I 呼び出し (SSA はすべて削除) の一般的なフォーマット:</p> <pre>ioOptionWithModifiers recordNameList [usingPCB pcbInfo] with #dli{ dliFunction } ;</pre>	<p>マイグレーション・ツールは、<i>recordNameList</i> を入出力オブジェクトの名前に設定します。</p>

表 119. 関数 - 変更済み DLI ステートメントのセグメント検索指数 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VisualAge Generator 入出力オプション:</p> <ul style="list-style-type: none"> • ADD • SCAN • SCAN - Scan update • SCAN - Scan parent • SCAN - Scan update および Scan parent • INQUIRY • UPDATE • DELETE • REPLACE 	<p>対応する EGL の <i>ioOptionWithModifiers</i>:</p> <ul style="list-style-type: none"> • add • get next • get next forUpdate • get next inParent • get next inParent forUpdate • get • get forUpdate • delete • replace 	<p>特別な考慮事項なし。</p> <p><i>ioOptionWithModifiers</i> の値は、未変更の DLI 入出力オプションの変換時に使用される値と同じです。</p>
<p>VisualAge Generator 入出力オプション:</p> <ul style="list-style-type: none"> • ADD • SCAN • SCAN - Scan update • SCAN - Scan parent • SCAN - Scan update および Scan parent • INQUIRY • UPDATE • DELETE • REPLACE 	<p>対応する EGL の <i>dliFunction</i>:</p> <ul style="list-style-type: none"> • ISRT • GN • GHN • GNP • GHNP • GU • GHU • DLET • REPL <p>注: <i>dliFunction</i> は、<i>ioOptionWithModifiers</i> と整合性がなければなりません。</p>	<p>特別な考慮事項なし。</p>
<p>VisualAge Generator の入出力オプション、SSA、およびコマンド・コード</p> <p>注:</p> <ul style="list-style-type: none"> • コマンド・コードはオプションです。SSA には、最大で 4 つのコマンド・コードがあります。 • VisualAge Generator は、生成時およびランタイム時に特殊な処理を実行して、パス呼び出しに関連する D および N コマンド・コードをサポートします。 	<p>EGL の入出力オプション、<i>recordNameList</i>、SSA、およびコマンド・コード</p> <p>注:</p> <ul style="list-style-type: none"> • * が指定されるのは、オプションのコマンド・コードが 1 つ以上指定されている場合に限られます。SSA には、最大で 4 つのコマンド・コードがあります。 • EGL <i>recordNameList</i> は、SSA で指定される入出力オプションとコマンド・コードによって異なります。詳しくは、この表の次の行を参照してください。 	<p>マイグレーション・ツールは、入出力オブジェクトと SSA 用に指定されるコマンド・コードとの組み合わせに基づいて、<i>recordNameList</i> を作成します。</p>

表 119. 関数 - 変更済み DL/I ステートメントのセグメント検索指数 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VisualAge Generator の入出力オブジェクトとコマンド・コード (D または N コマンド・コードは使用しない)</p> <p>注: データベースで検索または更新されるのは、入出力オブジェクト (ターゲット DL/I セグメント) のみです。</p>	<p>EGL <i>recordNameList</i> は、ターゲット DL/I セグメントの名前です。</p>	<p>マイグレーション・ツールは、<i>recordNameList</i> を入出力オブジェクトの名前に設定します。</p>
<p>VisualAge Generator の ADD 入出力オプション (D コマンド・コードを使用)</p> <p>注:</p> <ul style="list-style-type: none"> D コマンド・コードは、データベースに挿入される階層の先頭のセグメントを表します。すなわち、D コマンド・コードよりあとのセグメントは、すべて挿入されます。 D コマンド・コードを指定できるセグメントは 1 つのみです。 ターゲット・セグメントでは、D コマンド・コードが指定されることはありません。 	<p>add ステートメントを使用します。挿入されるすべてのセグメントを、入出力ステートメントのオブジェクトとしてリストする必要があります。以下は、DL/I Insert 呼び出しを指定する場合の例です。</p> <pre>add recordName1 , recordName2 , recordName3 [usingPCB pcbInfo] with #dli{ ISRT recordName1*D recordName2 recordName3 } ;</pre>	<p>マイグレーション・ツールは、入出力ステートメントの <i>recordNameList</i> を作成し、D コマンド・コードを指定する DL/I セグメント・レコードと、階層内のそれ以降のすべての DL/I セグメント・レコードがそのリストに含まれるようにします。</p>
<p>VisualAge Generator の SCAN、INQUIRY、および UPDATE 入出力オプション (D コマンド・コードを使用)</p> <p>注:</p> <ul style="list-style-type: none"> D コマンド・コードは、階層内の、データベースから検索されるセグメントを表します。データベースから検索される、ターゲット・セグメント以外の各セグメントでは、D コマンド・コードを指定する必要があります。ターゲット・セグメントは、必ず検索されます。 D コマンド・コードは、複数の SSA で指定できます。 ターゲット・セグメントでは、D コマンド・コードが指定されることはありません。 	<p>入出力オプションに対応する get ステートメントのフォームを使用します。データベースから検索されるすべてのセグメントを、入出力ステートメントのオブジェクトとしてリストする必要があります。以下は、DL/I DL/I Get Hold Next in Parent 呼び出しを指定する場合の例です。</p> <pre>get next inParent recordName1 , recordName3 forUpdate [usingPCB pcbInfo] with #dli{ GHNP recordName1*D recordName2 recordName3 } ;</pre>	<p>マイグレーション・ツールは、入出力ステートメントの <i>recordNameList</i> を作成し、D コマンド・コードを指定する個々の DL/I セグメント・レコードと、ターゲット・セグメントがそのリストに含まれるようにします。</p>

表 119. 関数 - 変更済み DL/I ステートメントのセグメント検索引数 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VisualAge Generator の REPLACE 入出力オプション (N コマンド・コードを使用)</p> <p>注:</p> <ul style="list-style-type: none"> • N コマンド・コードは、セグメントが、以前の SCAN または UPDATE で D コマンド・コードを使用して更新するために検索された場合でも、置換はされないことを意味します。 • N コマンド・コードは、複数のセグメントで指定できます。 • ターゲット・セグメントでは、N コマンド・コードが指定されることはありません。 	<p>replace ステートメントを使用します。ターゲット・セグメントのみが <i>recordNameList</i> で指定されます。以下は、DL/I Replace 呼び出しを指定する場合の例です。</p> <pre>replace recordName3 [usingPCB pcbInfo] with #dli{ REPL recordName1*N recordName3 } ;</pre>	<p>マイグレーション・ツールは、<i>recordNameList</i> を入出力オブジェクトの名前に設定します。</p>
<p>VisualAge Generator 関係演算子 (SSA 用):</p> <ul style="list-style-type: none"> • EQ および = • GT および > • LT および < • GE および >= および => • LE および <= および =< • NE および ^= および ^=^ 	<p>対応する EGL 関係演算子 (SSA 用):</p> <ul style="list-style-type: none"> • = • > • < • >= • <= • != <p>注:</p> <ul style="list-style-type: none"> • EGL がサポートするのは、SSA 用関係演算子の 1 つのバリエーションのみです。 • EGL は、!= を、DL/I で許容される不等演算子に変換します。 	<p>特別な考慮事項なし。</p>
<p>VisualAge Generator ブール演算子 (SSA 用):</p> <ul style="list-style-type: none"> • AND および & • OR および <p>注: VisualAge Generator は、従属の OR はサポートしていません。</p>	<p>対応する EGL ブール演算子 (SSA 用):</p> <ul style="list-style-type: none"> • & • (垂直バー) <p>注:</p> <ul style="list-style-type: none"> • EGL がサポートするのは、SSA 用ブール演算子の 1 つのバリエーションのみです。 • EGL は、従属の OR (# 記号) もサポートしています。 	<p>特別な考慮事項なし。</p>

表 119. 関数 - 変更済み DL/I ステートメントのセグメント検索指数 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>比較値項目</p> <p>注:</p> <ul style="list-style-type: none"> 比較値項目は、プログラム内の任意の項目にすることができます。 その項目が修飾されていない場合、VisualAge Generator はまず、セグメント・レコード内で、現在の SSA に関連付けられている項目を探します。 	<p>比較値項目</p> <p>注:</p> <ul style="list-style-type: none"> 比較値項目は、リテラルまたはプログラム内の任意の項目にすることができます。 その項目が修飾されていない場合、EGL はまず、ターゲット・セグメントを調べます。これは、階層内で最下位にあるセグメントです。 比較値項目の前には、ホスト変数を表すセミコロンを付ける必要があります。例えば、次のようになります。 <p style="text-align: center;">:qualifier.itemName</p> <p>EGL は、生成時にセミコロンを除去します。</p>	<p>比較値項目が修飾されていない場合、マイグレーション・ツールは、現在の SSA に関連付けられている DL/I セグメント・レコードを検査して、項目がそのレコードに含まれているかどうかを判断します。含まれていれば、マイグレーション・ツールは、その DL/I セグメント・レコードを比較値項目の修飾子として組み込みます。</p> <p>DL/I セグメント・レコードが使用できない場合、あるいは比較値項目がそのレコード内に存在しない場合は、特別な考慮事項が適用されます。詳細および起こりうる問題については、127 ページの『DL/I I/O および比較値項目』を参照してください。</p>

ステートメント

ステートメントに関するセクションは、次の表で構成されています。

- 一般規則 - データ項目修飾と数値リテラル (387 ページの表 120)
- 関数呼び出し (389 ページの表 121)
- 割り当て、MOVE、および MOVEA (389 ページの表 122)
- SET、391 ページの表 123
- RETR および FIND、394 ページの表 124
- EZEAD、EZESYS、および入出力エラー値を含む IF、WHILE および TEST (396 ページの表 125)
- CALL (401 ページの表 126)
- DXFR (402 ページの表 127)
- XFER (402 ページの表 128)

表 120. ステートメント - 一般規則 - データ項目修飾と数値リテラル

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>項目修飾規則: 項目が修飾されていない場合、VisualAge Generator は次の順序で項目を検索します。</p> <ul style="list-style-type: none"> 関数のローカル・ストレージまたはパラメーター・リストにある項目。 関数のローカル・ストレージまたはパラメーター・リストにある、関数およびレコードの入出力オブジェクト。項目名がこのカテゴリ内で固有でない場合は、項目名を修飾する必要があります。 プログラムの 1 次作業用ストレージ・レコード、呼び出されるパラメーター・リスト、テーブルおよび追加レコードのリスト、およびすべての入出力オブジェクト内にあるレコード、マップ、およびテーブル。項目名がこのカテゴリ内で固有でない場合は、項目名を修飾する必要があります。 項目名がプログラム内で見つからず、プログラムが暗黙項目を許容している場合、VisualAge Generator が項目の使用法に基づいてデータ項目定義を作成します。 	<p>項目修飾規則: フィールドが修飾されていない場合、または部分的に修飾されている場合は、EGL は以下の場所で順にそのフィールドを探します。</p> <ul style="list-style-type: none"> 関数のローカル・ストレージおよびパラメーター・リスト内の項目変数名およびレコード変数名を含む、関数レベルで宣言された変数。 入出力オブジェクト、その入出力オブジェクト内のフィールド、および関数のローカル・ストレージまたはパラメーター・リストで宣言されたすべてのレコード変数内のフィールド。 プログラム・レベルで宣言された、またはプログラムのパラメーター・リストで指定された、レコード変数名および項目変数名。 プログラムの use 書式ステートメントにリストされている書式。use ステートメントが FormGroup 名のみを指定している場合は、FormGroup 内のすべての書式が考慮されます。 プログラムの use 宣言で指定された DataTable 名。 プログラムに関して宣言されたすべてのレコード変数、書式、または DataTable 内のフィールド。use 書式ステートメントで FormGroup 名のみが指定されている場合は、FormGroup 内のすべての書式のすべてのフィールドが考慮されます。 プログラムに関する use 宣言で指定されたユーザー・ライブラリー内のフィールド。 システム・ライブラリー内のフィールド。 EGL は、暗黙項目を許容しません。すべての項目を明示的に定義する必要があります。 	<p>詳細と起こりうる問題については、87 ページの『レコード内のレベル 77 項目』、および 107 ページの『プログラム内の暗黙データ項目』を参照してください。</p>

表 120. ステートメント - 一般規則 - データ項目修飾と数値リテラル (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>数値リテラル:</p> <ul style="list-style-type: none"> 引用符で囲まれません。 各国語に応じて、ピリオド (.) またはコンマ (,) のどちらかを小数点として使用します。 浮動小数点リテラルはサポートされていません。 	<p>数値リテラル:</p> <ul style="list-style-type: none"> 引用符で囲まれません。 ピリオドを小数点として使用する必要があります。生成時に、decimalSymbol ビルド記述子オプションによって、生成される Java コードまたは COBOL コード内でピリオドまたはコンマのどちらを小数点として使用するかが決まります。 浮動小数点リテラルがサポートされています。 	<p>マイグレーション・ツールは、書式変数フィールドの初期値を除いて、小数点として使用されるコンマをピリオドに変換します。</p>
<p>文字リテラル、混合リテラル、および DBCS リテラル:</p> <ul style="list-style-type: none"> 引用符で囲む必要があります。 文字リテラルが単一引用符で囲まれている場合、そのリテラルは大文字に変換されます。 文字リテラルが二重引用符で囲まれている場合、そのリテラルは入力されたとおりに使用されます。 	<p>文字リテラル、混合リテラル、および DBCS リテラル:</p> <ul style="list-style-type: none"> 二重引用符で囲む必要があります。 リテラルは入力されたとおりに使用されます。 16 進リテラルおよびユニコード・リテラルもサポートされています。 オプションで、以下のような、リテラルのタイプを示す 1 文字の接頭部を指定することができます。 <ul style="list-style-type: none"> C - CHAR D - DBCHAR M - MBCHAR X - HEX オプションで、用紙送りなど、印刷不能文字を示す以下のような 2 文字の接頭部を指定することができます。 <ul style="list-style-type: none"> CX - CHAR DX - DBCHAR MX - MBCHAR UX - UNICODE 接頭部がない場合、リテラルは STRING データ型として扱われます。 	<p>マイグレーション・ツールは、リテラルについては 1 文字または 2 文字の接頭部を含めません。</p> <p>マイグレーション・ツールは、すべてのプログラムおよび VGUI レコードに関して textLiteralDefaultIsString プロパティを NO に設定します。</p>

表 121. ステートメント - 関数呼び出し

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
VAGen 構文の例: <code>functionName([argumentList]);</code> 注: 関数呼び出しは、単一の、完全なステートメントです。	EGL 構文の例: <code>functionName([argumentList]);</code> 注: 関数呼び出しは、別の関数または call ステートメントの引数として、または if や while ステートメントなど他のステートメントのオペランドとして使用できます。	同等な EGL のシステム・ライブラリー関数については、404 ページの『EZE ワード』を参照してください。 入出力エラー・ルーチンからの関数呼び出しについては、368 ページの表 113 を参照してください。
フロー・ステートメント内: <code>functionName();</code>	フロー・ステートメントはサポートされません。 <code>goto functionName;</code>	特別な考慮事項なし。

表 122. ステートメント - 割り当て、MOVE、および MOVEA

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
VAGen 構文の例: <code>target = functionName ([argumentList]);</code>	EGL 構文の例: <code>target = functionName ([argumentList]);</code>	同等な EGL のシステム・ライブラリー関数については、404 ページの『EZE ワード』を参照してください。
<code>target = numericExpression;</code> OR <code>target = numericExpression (R;</code>	<code>target = numericExpression ;</code> OR <code>target = mathLib.round (numericExpression, -mathlib.decimals(target));</code>	(R オプションが指定されている場合、マイグレーション・ツールはそのオプションを EGL mathlib.round() システム関数に変換します。

表 122. ステートメント - 割り当て、MOVE、および MOVEA (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>target = source;</p> <p>OR</p> <p>MOVE source [TO] target;</p> <p>注:</p> <ul style="list-style-type: none"> ターゲットは、レコード、マップ、データ項目、または一部の EZE データ・ワードのいずれかです。 ソースは、レコード、マップ、リテラル、データ項目、または一部の EZE データ・ワードのいずれかです。 ターゲットがレコードまたはマップである場合は、ソースもレコードまたはマップであることが必要です。move corresponding が実行されます。 	<p>target = source ;</p> <p>OR</p> <p>move source to target byName ;</p> <p>OR</p> <p>move source to target withV60Compat;</p> <p>注:</p> <ul style="list-style-type: none"> 代入ステートメントの場合: <ul style="list-style-type: none"> ターゲットは、レコード、項目、または一部のシステム変数のいずれかです。ターゲットがレコードである場合は、ソースもレコードであることが必要です。ソースは、バイトごとにターゲットに移動されます。 ソースは、レコード、リテラル、項目、または一部のシステム変数のいずれかです。 書式は、代入ステートメント内では使用できません。 代入ステートメントに対しては、move corresponding は実行されません。 move ステートメントの場合: <ul style="list-style-type: none"> ターゲットとソースは、VisualAge Generator の代入ステートメントまたは MOVE ステートメントの場合と同じです。 byName を指定した場合、EGL は move corresponding を実行します。 withV60Compat を指定した場合、移動は、ソースのパーツ型に応じて、項目間移動または move corresponding のいずれかになります。 <p>データ変換と切り捨ての規則は、VisualAge Generator の場合と同じです。</p> <p>EGL は VAGen よりも多くのデータ変換をサポートしています。例えば、CHAR フィールドに INT フィールドを割り当てることができます。</p>	<p>マイグレーション・ツールは、代入ステートメントと move ステートメントをマイグレーションする際に、以下の EGL 規則を考慮します。</p> <ul style="list-style-type: none"> EGL は、項目間移動の場合は代入ステートメントの使用を選択します。 VAGen の move corresponding の振る舞いを保持するために、レコードまたは書式が関係する移動には move byName ステートメントが必要です。 move withV60Compat ステートメントは容認されており、ソースのパーツ型に応じて、項目間移動または move corresponding として処理されます。 <p>このため、マイグレーション・ツールは以下の処理を行います。</p> <ul style="list-style-type: none"> 以下のいずれかの状態では、代入ステートメントに変換します。 <ul style="list-style-type: none"> ソースまたはターゲットが EZE データ・ワードである (例: EZEAPP)。 ソースがリテラルである。 ソースまたはターゲットが修飾項目または添え字付き項目である。 ソースまたはターゲットが、関数パラメーター・リストまたはローカル・ストレージ内の項目である。 ソースまたはターゲットが、関数パラメーター・リストまたはローカル・ストレージ内の関数またはレコードの入出力オブジェクトである場合は、move byName に変換します。 その他のすべての状態では、move withV60Compat に変換します。 <p>詳細と起こりうる問題については、131 ページの『代入ステートメント』を参照してください。</p>

表 122. ステートメント - 割り当て、MOVE、および MOVEA (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
MOVEA source [T0] target; OR MOVEA source [T0] target FOR occurrence; 注: <i>source</i> は、配列またはスカラーのどちらかです。	move source to target for all ; OR move source to target for occurrence ;	マイグレーション・ツールは MOVEA ステートメントを、 for 修飾子を持つ move ステートメントに変換します。マイグレーション・ツールは、以下の処理も行います。 <ul style="list-style-type: none"> FOR occurrence オプションが VisualAge Generator で指定されていなかった場合は、for all オプションを組み込みます。 FOR occurrence オプションが VisualAge Generator で指定されていた場合は、for occurrence オプションを組み込みます。 添え字が前に指定されていなかった場合は、ターゲットの添え字を 1 に設定します。 ソースは配列またはスカラー項目のどちらかなので、ソースに対しては添え字を 1 に設定しません。

表 123. ステートメント - SET

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
一般情報: • コンマまたはブランクを区切りに使用して、単一の SET ステートメントに複数のオプションを指定できます。	一般情報: • 単一の set ステートメントに複数のオプションを指定するには、コンマで区切る必要があります。	特別な考慮事項なし。
SET record SCAN; OR SET record EMPTY; 注: SET record EMPTY は、レベル 77 項目には影響しません。	set record position; OR set record empty;	マイグレーション・ツールは、レベル 77 レコードに対するステートメントを追加しません。
SET sqlItem NULL; 注: <i>sqlItem</i> は、SQL 行レコード内の項目、または関数の SQLITEM パラメーターです。	sqlItem = null; 注: <i>sqlItem</i> に指定できるのは、SQL 行レコード内の、 isSQLNullable プロパティが YES に設定された項目、または関数の nullable パラメーターです。	特別な考慮事項なし。

表 123. ステートメント - SET (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<pre>SET map [ALARM [CLEAR EMPTY]] ;</pre> <p>注:</p> <ul style="list-style-type: none"> CLEAR は EMPTY は相互に排他的です。 ALARM と、CLEAR または EMPTY は、PAGE オプションと組み合わせることができます。 	<pre>set form [alarm [initial empty]] ;</pre> <p>注:</p> <ul style="list-style-type: none"> initial および empty は、相互に排他的です。 PAGE オプションの置換表現は、他のオプションと組み合わせることはできません。 	<p>ALARM、CLEAR、または EMPTY が PAGE オプションと組み合わせて使用されている場合、マイグレーション・ツールは VAGen ステートメントを 2 つの EGL ステートメントに分割します。</p>
<pre>SET map PAGE ;</pre> <p>注: PAGE は、ALARM、および CLEAR または EMPTY のどちらかと組み合わせることができます。</p>	<pre>converseLib.clearScreen(); // display form</pre> <p>OR</p> <pre>converseLib.pageEject(); // printer form</pre> <p>注: PAGE オプションの置換表現は、他のオプションと組み合わせることはできません。</p>	<p>マイグレーション・ツールは以下のようして SET map PAGE をマイグレーションします。</p> <ul style="list-style-type: none"> SET map PAGE が他のいずれかのオプションと組み合わせて使用されている場合、マイグレーション・ツールは VAGen ステートメントを 2 つの EGL ステートメントに分割します。 マップが表示マップである場合、マイグレーション・ツールはステートメントを converseLib.clearScreen() に変換します。 マップがプリンター・マップである場合、マイグレーション・ツールはステートメントを converseLib.pageEject() に変換します。 マップを使用してマップ・タイプを判別できない場合、マイグレーション・ツールはステートメントをプレースホルダーとして無効関数名 converseLib.EZE_SETPAGE() に変換します。 <p>詳細と起こりうる問題については、133 ページの『SET map PAGE ステートメント』を参照してください。</p>

表 123. ステートメント - SET (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<pre>SET mapItem [CURSOR FULL [NORMAL DEFINED]] ;</pre> <p>注:</p> <ul style="list-style-type: none"> • <i>mapItem</i> は、マップのフィールド、または関数の MAPITEM パラメーターのどちらかです。 • NORMAL と DEFINED は相互に排他的です。 • CURSOR と FULL は、NORMAL または DEFINED のどちらかと組み合わせることができます。 • VisualAge Generator は、印刷マップに対して CURSOR、FULL、NORMAL、および DEFINED の設定を許容しますが、これらの設定は印刷出力に影響しません。 	<pre>set formField [cursor full [normal initialAttributes]] ;</pre> <p>注:</p> <ul style="list-style-type: none"> • <i>formField</i> に指定できるのは、書式の変数フィールド、または関数の field パラメーターのどちらかです。 • normal および initialAttributes は、相互に排他的です。 • cursor および full は、normal または initialAttributes のいずれかと組み合わせることができます。 • 印刷書式について、EGL は、cursor、full、normal、および initialAttributes の設定をサポートしません。 	<p>マイグレーション・ツールは、<i>formField</i> がテキスト書式と印刷書式のどちらにあるかに関係なく、それぞれのオプションと同等な EGL のオプションへのマイグレーションを行います。詳細と起こりうる問題については、134 ページの『SET mapItem 属性』を参照してください。</p>
<pre>SET mapItem [CURSOR FULL color extendedHighlight MODIFIED [BRIGHT DARK] [PROTECT AUTOSKIP]] ;</pre> <p>注:</p> <ul style="list-style-type: none"> • <i>mapItem</i> は、マップのフィールド、または関数の MAPITEM パラメーターのどちらかです。 • BRIGHT は DARK は相互に排他的です。 • PROTECT と AUTOSKIP は相互に排他的です。 • その他のオプションは、すべて組み合わせ可能です。 • VisualAge Generator は、印刷マップに対してこれらの属性の設定を許容します。ただし、印刷出力に影響を及ぼすものは USCORE の拡張強調表示オプションのみです。 	<pre>set formField [cursor full color extendedHighlight modified [bold invisible] [protect skip]] ;</pre> <p>注:</p> <ul style="list-style-type: none"> • <i>formField</i> に指定できるのは、書式の変数フィールド、または関数の field パラメーターのどちらかです。 • bold および invisible は、相互に排他的です。 • Protect および skip は、相互に排他的です。 • その他のオプションは、すべて組み合わせ可能です。 • underline の <i>extendedHighlight</i> オプションを除いて、EGL は印刷書式に対してはこれらの属性の設定をサポートしていません。 	<p>マイグレーション・ツールは、<i>formField</i> がテキスト書式と印刷書式のどちらにあるかに関係なく、それぞれのオプションと同等な EGL のオプションへのマイグレーションを行います。詳細と起こりうる問題については、134 ページの『SET mapItem 属性』を参照してください。色 情報と <i>extendedHighlight</i> 情報については、この表内の後の行を参照してください。</p>

表 123. ステートメント - SET (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
color: MONO BLUE GREEN PINK RED TURQ YELLOW WHITE	color: defaultColor blue green magenta red cyan yellow white	特別な考慮事項なし。
extendedHighlight: NOHILITE BLINK RVIDEO USCORE	extendedHighlight: noHighLight blink reverse underline	特別な考慮事項なし。

表 124. ステートメント - RETR および FIND

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
RETR DataItem1 table[.searchColumn] DataItem2 [returnColumn] ; 注: <ul style="list-style-type: none"> • <i>searchColumn</i> が指定されていない場合、デフォルトはテーブルの先頭列です。 • <i>returnColumn</i> が指定されていない場合、デフォルトはテーブルの 2 列目です。 	if (<i>DataItem1</i> in <i>DataTable.searchColumn</i>) <i>DataItem2</i> = <i>DataTable.returnColumn</i> [sysVar.arrayIndex]; end 注: <i>searchColumn</i> と <i>returnColumn</i> が必要です。	マイグレーション・ツールは、RETR ステートメントを if ステートメントと代入ステートメントに変換します。 マイグレーション時にテーブルが使用できない場合は、特別な考慮事項が適用されます。詳細と起こりうる問題については、132 ページの『RETR ステートメント』を参照してください。

表 124. ステートメント - RETR および FIND (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>FIND DataItem table[.searchColumn] trueStatement;</p> <p>OR</p> <p>FIND DataItem table[.searchColumn] , falseStatement ;</p> <p>OR</p> <p>FIND DataItem table[.searchColumn] trueStatement [,] falseStatement ;</p> <p>注:</p> <ul style="list-style-type: none"> • <i>searchColumn</i> が指定されていない場合、デフォルトはテーブルの先頭列です。 • FIND がプログラム・フローの中で使用されている場合、<i>trueStatement</i> と <i>falseStatement</i> は、main 関数の名前、または EZECLOS のどちらかです。 • FIND が関数の中で使用されている場合、<i>trueStatement</i> と <i>falseStatement</i> は、任意の関数の名前、EZECLOS、EZEFL0、または EZERTN のいずれかです。 	<p>if (<i>DataItem in DataTable.searchColumn</i>) <i>EGLtrueStatement</i> ; end</p> <p>または</p> <p>if (<i>DataItem in DataTable.searchColumn</i>) else <i>EGLfalseStatement</i> ; end</p> <p>または</p> <p>if (<i>DataItem in DataTable.searchColumn</i>) <i>EGLtrueStatement</i> ; else <i>EGLfalseStatement</i> ; end</p> <p>注: <i>searchColumn</i> が必要です。</p>	<p>マイグレーション・ツールは、FIND ステートメントを、if ステートメント、および <i>true</i> と <i>false</i> の両ステートメントと等価な EGL のステートメントに変換します。 <i>trueStatement</i> および <i>falseStatement</i> から対応する EGL ステートメントへの変換については、この表内の後の行を参照してください。</p>
<p>フロー内の <i>true/falseStatement</i></p> <ul style="list-style-type: none"> • <i>functionName()</i> (main のみ) • EZECLOS 	<p>対応する EGL の置き換え:</p> <ul style="list-style-type: none"> • goto <i>functionName</i>; • exit program; 	<p>特別な考慮事項なし。</p>
<p>関数内の <i>true/falseStatement</i>:</p> <ul style="list-style-type: none"> • <i>functionName</i> (任意の関数) • EZECLOS • EZEFL0 • EZERTN 	<p>対応する EGL の置き換え:</p> <ul style="list-style-type: none"> • <i>functionName</i>(); • exit program; • exit stack; • return; 	<p>特別な考慮事項なし。</p>

表 125. ステートメント - ZEZAID、EZESYS、および入出力エラー値を含む IF、WHILE および TEST

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<pre>IF logicalExpression ; { statement ; } [ELSE; { statement ; }] END;</pre>	<pre>if (EGLLogicalExpression) { EGLStatement ; } [else { EGLStatement ; }] end</pre>	VAGen 論理式と EGL 論理式の関係については、この表内の後の行を参照してください。
<pre>WHILE logicalExpression ; { statement ; } END;</pre>	<pre>while (EGLLogicalExpression) { EGLStatement ; } end</pre>	VAGen 論理式と EGL 論理式の関係については、この表内の後の行を参照してください。
<pre>TEST testCondition trueStatement ; TEST testCondition , falseStatement ; TEST testCondition trueStatement [,] falseStatement ;</pre> <p>注:</p> <ul style="list-style-type: none"> • TEST ステートメントは、IF ... IS ステートメントと同様です。ただし、TEST <i>mapItem</i> nnn、+nnn、および -nnn は例外であり、これと等価な IF ステートメントはありません。 • TEST がプログラム・フローの中で使用されている場合、<i>trueStatement</i> と <i>falseStatement</i> は、main 関数の名前、または EZECLOS のどちらかです。 • TEST が関数の中で使用されている場合、<i>trueStatement</i> と <i>falseStatement</i> は、任意の関数の名前、EZECLOS、EZEFLO、または EZERTN のいずれかです。 	<pre>if (EGLLogicalExpression) EGLtrueStatement ; end if (EGLLogicalExpression) else EGLfalseStatement ; end if (EGLLogicalExpression) EGLtrueStatement ; else EGLfalseStatement ; end</pre>	<p>以下の例外を除き、マイグレーション・ツールは TEST ステートメントを等価な if ... is ステートメントと、true および false ステートメントと等価な EGL ステートメントに変換します。例外は以下のとおりです。</p> <ul style="list-style-type: none"> • TEST <i>mapItem</i> nnn、+nnn、および -nnn。 • TEST <i>sqlItem</i> NULL <p>VAGen 論理式と EGL 論理式の関係については、この表内の後の行を参照してください。</p> <p>TEST <i>mapItem</i> nnn、+nnn、および -nnn の変換については、この表内の後の行を参照してください。</p> <p>TEST <i>sqlItem</i> NULL の変換については、この表内の後の行を参照してください。</p> <p><i>trueStatement</i> および <i>falseStatement</i> から対応する EGL ステートメントへの変換については、この表内の後の行を参照してください。</p>
<p>IF と WHILE 用の VisualAge Generator プール演算子:</p> <ul style="list-style-type: none"> • AND • OR 	<p>if および while に対応する EGL プール演算子:</p> <ul style="list-style-type: none"> • && または and • または or 	<p>「VAGen 論理演算 (AND および OR) の使用」マイグレーション設定を選択解除した場合、マイグレーション・ツールは && および を論理演算子として使用します。このマイグレーション設定を選択した場合、マイグレーション・ツールは and および or を論理演算子として使用します。</p>

表 125. ステートメント - EZEAD、EZESYS、および入出力エラー値を含む IF、WHILE および TEST (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>IF と WHILE 用の VisualAge Generator 関係演算子:</p> <ul style="list-style-type: none"> • EQ および = • NE および ^= • LE および <= および =< • LT および < • GE および >= および => • GT および > 	<p>if および while に対応する EGL の関係演算子:</p> <ul style="list-style-type: none"> • == • != • <= • < • >= • > 	<p>特別な考慮事項なし。</p>
<p>IF と WHILE 用の VisualAge Generator 状態演算子:</p> <ul style="list-style-type: none"> • IS • NOT 	<p>if および while に対応する EGL の状態演算子:</p> <ul style="list-style-type: none"> • is • not 	<p>ほとんどの場合、マイグレーション・ツールは以下の処理を行います。</p> <ul style="list-style-type: none"> • IS および NOT を示されたとおりにマイグレーションします。 • VAGen TEST ステートメントを EGL if ... is ステートメントにマイグレーションします。 <p>例外は以下のとおりです。</p> <ul style="list-style-type: none"> • TEST <i>mapItem</i> nnn、+nnn、および -nnn • TEST <i>mapItem</i> NULL <p>例外の詳細については、この表内の後の行を参照してください。</p>
<p>IF と WHILE 用の VisualAge Generator 配列演算子:</p> <ul style="list-style-type: none"> • IN <p>配列の特定の要素から検索を開始するには、X IN Y[Z] を使用します。ここで、Z は、配列内の検索を開始する索引です。配列に添え字が指定されていない場合、検索は配列の先頭から開始します。</p>	<p>if および while に対応する EGL の状態演算子:</p> <ul style="list-style-type: none"> • in <p>配列の特定の要素から検索を開始するには、X in Y from Z を使用します。ここで、Z は配列内の検索を開始する索引です。配列に添え字が指定されていない場合、検索は配列の先頭から開始します。</p>	<p>特別な考慮事項が適用されます。</p> <p>135 ページの『IN がリテラルかスカラーかの検査』を参照してください。</p>
<p>VisualAge Generator <i>mapItem</i> の状態条件:</p> <ul style="list-style-type: none"> • BLANK または BLANKS • CURSOR • DATA • MODIFIED • NULL または NULLS • NUMERIC 	<p>対応する EGL 書式フィールドの状態条件:</p> <ul style="list-style-type: none"> • blanks • cursor • data • modified • blanks • numeric 	<p>マイグレーション・ツールは、同等な EGL 状態条件への変換を行います。</p> <p><i>mapItem</i> NULL には特別な考慮事項が適用されます。詳細と起こりうる問題については、136 ページの『SQL 項目とマップ項目が NULL かどうかの検査』を参照してください。</p>

表 125. ステートメント - EZEID、EZESYS、および入出力エラー値を含む IF、WHILE および TEST (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>TEST ステートメントの特殊な mapItem の状態条件: nnn +nnn -nnn</p> <p>注: この条件は、ユーザーが入力したデータの長さを nnn と比較します。nnn、+nnn、または -nnn に対応して、=、>、または < のテストが行われます。</p>	<p>EGL は、この状態条件のサポートを直接は提供していません。ただし、以下の手順を実行することにより、等価な結果を得ることができます。</p> <ol style="list-style-type: none"> システム・ライブラリー関数 converseLib.fieldInputLength() を使用します。この関数は、ユーザーが入力したデータの長さを返します。 if ステートメントを使用して、得られた長さをそれぞれ nnn、+nnn、または -nnn に対応する ==、>、または < によって比較します。 	<p>プログラムをマイグレーションする際に、マイグレーション・ツールは常に次の宣言を組み込みます。</p> <pre><custPrefix>EZE_ITEMLEN</pre> <p>マイグレーション・ツールは、TEST nnn、+nnn、または -nnn に関して以下の処理を行います。</p> <ul style="list-style-type: none"> TEST ステートメントの直前にステートメントを追加して、次の設定を行います。 <pre><custPrefix>EZE_ITEMLEN</pre> <p>この設定には、システム・ライブラリー関数 converseLib.fieldInputLength() を使用します。</p> <ul style="list-style-type: none"> TEST ステートメントを if ステートメントに変更し、 <pre><custPrefix>EZE_ITEMLEN</pre> <p>を == nnn、> nnn、または < nnn と比較します。</p>
<p>VisualAge Generator マップの状態条件:</p> <ul style="list-style-type: none"> MODIFIED 	<p>対応する EGL 書式の状態条件:</p> <ul style="list-style-type: none"> modified 	<p>特別な考慮事項なし。</p>
<p>VisualAge Generator EZEID の状態条件:</p> <ul style="list-style-type: none"> ENTER BYPASS PAn (ここで、n = 1, 2, 3) PFn (ここで n は 1 から 24) PA PF 	<p>対応する EGL converseVar.eventKey の状態条件:</p> <ul style="list-style-type: none"> enter bypass pan (ここで、n = 1, 2, 3) pfn (ここで n は 1 から 24) pakey pfkey 	<p>特別な考慮事項なし。</p>

表 125. ステートメント - EZEAD、EZESYS、および入出力エラー値を含む IF、WHILE および TEST (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VisualAge Generator の sqlItem 状態条件:</p> <ul style="list-style-type: none"> • BLANK または BLANKS • NULL • NUMERIC • TRUNC <p>注: sqlItem に NULL がないかどうかを検査する場合は、IS または NOT 演算子を使用します。これは、他の状態を検査する場合と同じです。</p>	<p>EGL の対応する SQL 項目の状態条件:</p> <ul style="list-style-type: none"> • blanks • null • numeric • trunc <p>注: SQL 項目に null がないかどうかを検査する場合は、以下を使用します。</p> <p><code>sqlItem == null</code> または <code>sqlItem != null</code> <code>is</code> または <code>not</code> 演算子は使用しません。</p>	<p>マイグレーション・ツールは、同様な EGL 状態条件への変換を行います。</p> <p>sqlItem NULL には特別な考慮事項が適用されます。詳細と起こりうる問題については、136 ページの『SQL 項目とマップ項目が NULL かどうかの検査』を参照してください。</p>
<p>VisualAge Generator レコードの状態条件:</p> <ul style="list-style-type: none"> • DED • DUP • EOF • ERR • FMT • FNA • FNF • FUL • HRD • LOK • NRF • UNQ <p>注:</p> <ul style="list-style-type: none"> • DUP は、SQL レコードと非 SQL レコードの両方に対してサポートされます。 • SQL レコードの場合、DUP と UNQ は同等で、常にハード・エラーです。 • 非 SQL レコードの場合、DUP と UNQ は同等ではなく、両方ともソフト・エラーです。 • LOK は、OS/400 上でのみサポートされるソフト・エラーです。 	<p>対応する EGL レコードの状態条件:</p> <ul style="list-style-type: none"> • deadLock • duplicate または unique • endOfFile • ioError • invalidFormat • fileNotAvailable • fileNotFound • full • hardIOError • deadLock • noRecordFound • unique <p>注:</p> <ul style="list-style-type: none"> • duplicate は、非 SQL レコードに対してのみサポートされており、ソフト・エラーになります。 • unique は、SQL および非 SQL のどちらのレコードに対してもハード・エラーになります。 • LOK は deadlock に変換され、常にハード・エラーになります。 	<p>マイグレーション・ツールは、同様な EGL 状態条件への変換を行います。</p> <p>レコード・タイプに応じて、DUP のマイグレーションには特別な考慮事項が適用されます。詳細と起こりうる問題については、138 ページの『入出力エラー値 UNQ および DUP』を参照してください。</p> <p>LOK のマイグレーションにも、特別な考慮事項が適用されます。詳細と起こりうる問題については、141 ページの『入出力エラー値 LOK』を参照してください。</p>
<p>VisualAge Generator の UI レコードの状態条件:</p> <ul style="list-style-type: none"> • MODIFIED 	<p>EGL の対応する VGUI レコードの状態条件:</p> <ul style="list-style-type: none"> • modified 	<p>特別な考慮事項なし。</p>

表 125. ステートメント - *EZEAD*、*EZESYS*、および入出力エラー値を含む *IF*、*WHILE* および *TEST* (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
VisualAge Generator の DataItem の状態条件: <ul style="list-style-type: none"> • BLANK または BLANKS • NUMERIC 	対応する EGL のデータ項目の状態条件: <ul style="list-style-type: none"> • blanks • numeric 	特別な考慮事項なし。
VisualAge Generator EZESYS の状態条件: <ul style="list-style-type: none"> • AIX • AIXCICS • HP • IMSBMP • IMSVS • MVS BATCH • MVSCICS • NTCICS • OS2 • OS2CICS • OS2GUI • OS400 • SCO • SOLACICS • SOLARIS • TSO • VMCMS • VMBATCH • VSEBATCH • VSECICS • WINGUI • WINNT • ITF 	対応する sysVar.systemType の状態条件: <ul style="list-style-type: none"> • aix • AIXCICS • hpux • imsbmp • imsvs • zosbatch • zoscics • NTCICS • OS2 • OS2CICS • OS2GUI • iseriesc • SCO • SOLACICS • solaris • TSO • VMCMS • VMBATCH • vsebatch • vsecics • WINGUI • win • debug 	マイグレーション・ツールは、同等な EGL 状態条件への変換を行います。 EZESYS の状態の検査には、特別な考慮事項が適用されます。詳細と起こりうる問題については、142 ページの『EZESYS』を参照してください。 注: 一部の VAGen ランタイム環境はサポートされません。ただし、マイグレーション・ツールは常に、等価な値が EGL では有効でなくても、その値に変換します。マイグレーション・ツールは、サポートされていない値を保持します。これにより、TSO などのサポートされていない環境から、ZOSCICS などのサポートされている EGL 環境に変更した場合に、ロジックの更新が必要になる可能性のある場所が見つかりやすくなります。
フロー内の true/falseStatement: <ul style="list-style-type: none"> • functionName() (main のみ) • EZECLDS 	対応する EGL の置き換え: <ul style="list-style-type: none"> • goto functionName ; • exit program; 	特別な考慮事項なし。
関数内の true/falseStatement: <ul style="list-style-type: none"> • functionName (任意の関数) • EZECLDS • EZEFLD • EZERTN 	対応する EGL の置き換え: <ul style="list-style-type: none"> • functionName(); • exit program; • exit stack; • return; 	特別な考慮事項なし。

表 126. ステートメント - CALL

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>CALL <i>programName</i> <i>argument</i> [{ [,] <i>argument</i> }] [(<i>options</i>) ;</p> <p>または</p> <p>CALL <i>serviceRoutine</i> <i>argument</i> [{ [,] <i>argument</i> }] [(<i>options</i>) ;</p> <p>注:</p> <ul style="list-style-type: none"> 引数を区切るコンマはオプションです。 引数リストは括弧で囲まれません。 <i>programName</i> または <i>serviceRoutine</i> は、引用符で囲まれることはありません。 	<p>call <i>programName</i> (<i>argument</i> [{ , <i>argument</i> }]) [{ <i>options</i> }] ;</p> <p>注:</p> <ul style="list-style-type: none"> 引数を区切るコンマは必要です。 引数リストは括弧で囲む必要があります。 <i>programName</i> を予約語にすることはできません。プログラムが非 EGL プログラムの場合は、リンケージ・オプション要素を使用して実名を指定します。 プログラムがワークスペース内不在の場合は、<i>programName</i> を引用符で囲む必要があります。 	<p>オプションから、対応する EGL のステートメントまたはオプションへの変換については、この表内の後の行を参照してください。</p> <p>「CALL および DXFR プログラム名を引用符で囲む」マイグレーション設定を選択した場合、マイグレーション・ツールは <i>programName</i> を引用符で囲みます。この設定をクリアした場合、マイグレーション・ツールは <i>programName</i> を引用符で囲みません。</p> <p>マイグレーション・ツールは、設定に関係なく、以下の処理を行います。</p> <ul style="list-style-type: none"> (NONCSP オプションが指定された場合は、<i>programName</i> を引用符で囲みます。 <i>programName</i> が EZCHART の場合は、<i>programName</i> を引用符で囲み、isExternal プロパティを yes に設定します。 <p>サービス・ルーチン用の CALL ステートメントのマイグレーションについては、415 ページの『サービス・ルーチン』を参照してください。</p>
REPLY オプション	<p>REPLY オプションが VisualAge Generator で指定されている場合、マイグレーション・ツールはこのオプションを以下の EGL ステートメントに変換します。</p> <p>try call <i>programName</i> (<i>argument</i> [{ , <i>argument</i> }]) [{ <i>otherOptions</i> }] ; end</p>	<p>REPLY オプションが指定されている場合、マイグレーション・ツールは try ブロックを組み込みます。</p>
<p>otherOptions:</p> <ul style="list-style-type: none"> NOMAPS NONCSP 	<p>対応する EGL otherOptions:</p> <ul style="list-style-type: none"> isNoRefresh = YES isExternal = YES 	<p>特別な考慮事項なし。</p>

表 127. ステートメント - DXFR

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>DXFR target [recordName] [(NONCSP)] ;</p> <p>ここで、target は次のとおりです。</p> <ul style="list-style-type: none"> • programName • EZEAPP <p>注:</p> <ul style="list-style-type: none"> • 任意のレコードを渡すことができます。 • 作業用ストレージ・レコードが渡される場合に、レベル 77 項目は含まれません。 • programName は引用符で囲まれません。 	<p>transfer to program target [passing recordName] [isExternal = YES] ;</p> <p>ここで、target は次のとおりです。</p> <ul style="list-style-type: none"> • programName • sysVar.transferName <p>注:</p> <ul style="list-style-type: none"> • 任意のレコードを渡すことができます。 • programName を予約語にすることはできません。プログラムが非 EGL プログラムの場合は、リンケージ・オプション要素を使用して実名を指定します。 • プログラムがワークスペース内でない場合は、programName を引用符で囲む必要があります。 	<p>「CALL および DXFR プログラム名を引用符で囲む」マイグレーション設定を選択した場合、マイグレーション・ツールは programName を引用符で囲みます。この設定をクリアした場合、マイグレーション・ツールは programName を引用符で囲みません。</p> <p>マイグレーション・ツールは、設定に関係なく、以下の処理を行います。</p> <ul style="list-style-type: none"> • (NONCSP オプションが指定された場合は、programName を引用符で囲みます。 • sysVar.transferName は引用符で囲みません。

表 128. ステートメント - XFER

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>バリエーション 1 - transfer へのマイグレーション (マップまたは UI レコードの指定なし)</p> <p>XFER target [recordName] [(NONCSP)] ;</p> <p>ここで、target は次のとおりです。</p> <ul style="list-style-type: none"> • transactionName • EZEAPP <p>注:</p> <ul style="list-style-type: none"> • このフォーマットの XFER には、マップまたは UI レコードは指定されません。 • 任意のレコードを渡すことができます。作業用ストレージ・レコードが渡される場合に、レベル 77 項目は含まれません。 	<p>以下に、transfer ステートメントの EGL 構文を示します。</p> <p>transfer to transaction target [passing recordName] [isExternal = YES] ;</p> <p>ここで、target は次のとおりです。</p> <ul style="list-style-type: none"> • transactionName • sysVar.transferName <p>注: 任意のレコードを渡すことができます。</p>	<p>ステートメントにコンマが含まれていない場合、マイグレーション・ツールは XFER を EGL transfer to transaction ステートメントに変換します。</p> <p>transactionName について詳しくは、この表内の後の行を参照してください。</p>

表 128. ステートメント - XFER (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>バリエーション 2 - Show へのマイグレーション (マップを指定した XFER または UI レコードを指定した XFER)</p> <pre>XFER target [recordName] , mapName [(NONCSP) ;</pre> <p>または</p> <pre>XFER target [recordName] , UIRecordName ;</pre> <p>ここで、<i>target</i> は次のとおりです。</p> <ul style="list-style-type: none"> • <i>transactionName</i> • EZEAPP • UI レコードを指定した XFER の ‘ ‘ <p>注:</p> <ul style="list-style-type: none"> • 任意のレコードを渡すことができます。作業用ストレージ・レコードが渡される場合に、レベル 77 項目は含まれません。 • (NONCSP は、マップを指定した XFER でのみサポートされます。) 	<p>show ステートメントの場合、EGL 構文はフォームまたは VGUI レコードが使用されているかどうかによって異なります。</p> <pre>show formName returning to target [passing recordName] [isExternal = YES] ;</pre> <p>または</p> <pre>show UIRecordName [returning to target] [passing recordName] ;</pre> <p>ここで、<i>target</i> は次のとおりです。</p> <ul style="list-style-type: none"> • <i>transactionName</i> • sysVar.transferName <p>注:</p> <ul style="list-style-type: none"> • 任意のレコードを渡すことができます。 • show ステートメントに VGUI レコードを使用した場合、ターゲットが ‘ ‘ であれば、returning to target 文節は省略されます。 	<p>ステートメントにコンマが含まれている場合、マイグレーション・ツールは XFER ステートメントを EGL show ステートメントに変換します。</p> <p><i>transactionName</i> について詳しくは、この表内の後の行を参照してください。</p>
<p><i>transactionName</i></p> <p>注:</p> <ul style="list-style-type: none"> • <i>transactionName</i> は、非トランザクション環境ではプログラム名です。 • <i>transactionName</i> は引用符で囲まれません。 	<p><i>transactionName</i></p> <p>注:</p> <ul style="list-style-type: none"> • <i>transactionName</i> は、非トランザクション環境ではプログラム名です。<i>transactionName</i> を予約語にすることはできません。 • プログラムが非 EGL プログラムの場合は、リンケージ・オプション要素を使用して実名を指定します。 • プログラムがワークスペース内不在の場合は、<i>transactionName</i> を引用符で囲む必要があります。 	<p>「XFER プログラム名を引用符で囲む」マイグレーション設定を選択した場合、マイグレーション・ツールは <i>transactionName</i> を引用符で囲みます。この設定をクリアした場合、マイグレーション・ツールは <i>transactionName</i> を引用符で囲みません。</p> <p>マイグレーション・ツールは、設定に関係なく、以下の処理を行います。</p> <ul style="list-style-type: none"> • (NONCSP オプションが指定された場合は、<i>transactionName</i> を引用符で囲みます。 • sysVar.transferName は引用符で囲みません。

EZE ワード

EZE ワードに関するセクションは、次のセクションで構成されています。

- 404 ページの『プログラム・フローの EZE ワード』
- 405 ページの『SQL EZE ワード』
- 407 ページの『DL/I EZE ワード』
- 408 ページの『日付と時刻の EZE ワード』
- 408 ページの『その他のデータの EZE ワード』
- 411 ページの『一般的な関数の EZE ワード』
- 412 ページの『ストリング EZE ワード』
- 413 ページの『数学 EZE ワード』
- 415 ページの『ユーザー・インターフェースの EZE ワード』
- 415 ページの『オブジェクト・スクリプトの EZE ワード』

プログラム・フローの EZE ワード

表の左側の欄には、VisualAge Generator 4.5 の EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。

表 129. プログラム・フローの EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL
EZECLOS	<p>これは、ロケーションによって次のように異なります。</p> <ul style="list-style-type: none">• 入出力エラー・ルーチンで使用されている場合、マイグレーション・ツールは EZECLOS を try ブロック内で次の EGL コードに変換します。 <p>onException exit program;</p> <ul style="list-style-type: none">• 他の場所で使用されている場合 (TEST や FIND の true または false オペランドとして使用されている場合など)、マイグレーション・ツールは EZECLOS を次の EGL コードに変換します。 <p>exit program;</p> <p>注: exit program ステートメントのデフォルトの戻りコードは sysVar.returnValue です。これは、EZERCODE と同等です。このデフォルトは、VisualAge Generator と同じ機能を実現します。</p>

表 129. プログラム・フローの EZE ワード (続き)

VisualAge Generator 4.5 の EZE ワード	EGL
<p>EZEFLO</p> <p>注: EZEFLO は、フロー・ステートメント内では使用できません。</p>	<p>これは、ロケーションによって次のように異なります。</p> <ul style="list-style-type: none"> 入出力エラー・ルーチンで使用されている場合、マイグレーション・ツールは EZEFLO を try ブロック内で次の EGL コードに変換します。 <pre>onException exit stack;</pre> <ul style="list-style-type: none"> 他の場所で使用されている場合 (TEST や FIND の true または false オペランドとして使用されている場合など)、マイグレーション・ツールは EZEFLO を次の EGL コードに変換します。 <pre>exit stack;</pre>
<p>EZERTN または EZERTN(<i>return value</i>)</p> <p>注:</p> <ul style="list-style-type: none"> EZERTN は、フロー・ステートメント内では使用できません。 EZERTN(<i>return value</i>) は、入出力エラー・ルーチンとしては使用できません。 	<p>これは、ロケーションによって次のように異なります。</p> <ul style="list-style-type: none"> 入出力エラー・ルーチンで使用されている場合、マイグレーション・ツールは try ブロックを組み込みますが、onException ステートメントを省略します。 他の場所で使用されている場合、マイグレーション・ツールは EZERTN を次の EGL コードに変換します。 <pre>return;</pre> <p>OR</p> <pre>return(returnValue);</pre> <p>注: <i>returnValue</i> が EZESYS である場合は、他の考慮事項について、408 ページの『その他のデータの EZE ワード』に記載されている EZESYS に関する情報を参照してください。</p>

SQL EZE ワード

表の左側の欄には、VisualAge Generator 4.5 の EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。

表 130. SQL EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZECONCT	vgLib.connectionService() 引数は VAGen の場合と同じです。ただし、デバッグと Java 生成の場合、作業単位引数の値の一部はサポートされません。JDBC は、単一フェーズ・コミットのみをサポートします。詳しくは、277 ページの『SQL サポートに関する違い』を参照してください。
EZESQCOD	sysVar.sqlData.sqlcode
EZESQISL 注: <ul style="list-style-type: none"> VisualAge Generator 4.5 の場合、EZESQISL は ODBC 環境での使用がサポートされています。 それ以外のすべての環境では、EZESQISL はサポートされないか VisualAge Generator 内で無視されますが、互換性のために保持されます。 	vgVar.sqlIsolationLevel 注: EGL は次の使用を行う場合に、 vgVar.sqlIsolationLevel をサポートします。 <ul style="list-style-type: none"> VAGen 互換モードが選択されているかどうかに関わらず vgLib.connectionService() を使用する場合 VAGen 互換モードが選択されているときのみ sysLib.connect() を使用する場合
EZESQLCA	sysVar.sqlData.sqlca 注: sysVar.sqlData.sqlca は、EGL では部分的にのみサポートされます。デバッグおよび Java 生成の場合、EGL は sysVar.sqlData.sqlerrmc と sysVar.sqlData.sqlwarn[7] の値を格納するフィールドを sysVar.sqlData.sqlca 内に設定しません。
EZESQRD3	sysVar.sqlData.sqlerrd[3] 注: マイグレーション・ツールは、これを配列参照に変更します。
EZESQRRM	sysVar.sqlData.sqlerrmc 注: sysVar.sqlData.sqlerrmc は、デバッグまたは Java 生成の場合にはサポートされません。
EZESQWN1	sysVar.sqlData.sqlwarn[2] 注: マイグレーション・ツールは、これを配列参照に変更します。
EZESQWN6	sysVar.sqlData.sqlwarn[7] 注: マイグレーション・ツールは、これを配列参照に変更します。 sysVar.sqlData.sqlwarn[7] は、デバッグまたは Java 生成の場合にはサポートされません。
N/A	sysVar.sqlData.sqlState 注: これは EGL 用の新規のもので、VisualAge Generator 4.5 に同等なワードはありません。マイグレーション・ツールによってこれに変換されるワードはありません。

DL/I EZE ワード

表の左側の欄には、VisualAge Generator 4.5 EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。

表 131. DL/I EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL
EZEDLCER	dliVar.cicsError
EZEDLCON	dliVar.cicsCondition
EZEDLDBD	dliVar.dbName
EZEDLERR	dliVar.handleHardDLIErrors
EZEDLKEY	dliVar.keyArea[1:dliVar.keyAreaLen]
EZEDLKYL	dliVar.keyAreaLen
EZEDLLEV	dliVar.segmentLevel
EZEDLPCB 注: これは配列です。デフォルトの添え字は 1 です。	<p>マイグレーション・ツールは、プログラムの PSBRecord を宣言する変数を常に <i>psb</i> にセットします。そのため、ステートメント内では、マイグレーション・ツールが EZEDLPCB[n] を次の方法で変換します。</p> <ul style="list-style-type: none"> • EZEDLPCB[0] は <i>psb.iopcb</i> に変換されます。 • 1 はデフォルトの添え字であるため、EZEDLPCB は <i>psb.pcb1</i> に変換されます。 • EZEDLPCB[n] (ここで <i>n</i> は数値リテラル) は <i>psb.pcbn</i> に変換されます。 <p>プログラムの呼び出し済みパラメーターのリストには、特有の考慮事項があります。詳しくは、357 ページの表 107を参照してください。</p>
EZEDLPRO	dliVar.procOptions
EZEDLPSB	<p>call ステートメント以外のステートメントでは、EZEDLPSB は次のように変換されます。</p> <p>dliLib.psbData.psbName</p> <p>call ステートメントでは、EZEDLPSB は次のように変換されます。</p> <p>dliLib.psbData</p> <p>プログラムの呼び出し済みパラメーターのリストには、特有の考慮事項があります。詳しくは、357 ページの表 107を参照してください。</p>
EZEDLRST	dliVar.cicsRestart
EZEDLSEG	dliVar.segmentName
EZEDLSSG	dliVar.numSensitiveSegs
EZEDLSTC	dliVar.statusCode

表 131. DLI EZE ワード (続き)

VisualAge Generator 4.5 の EZE ワード	EGL
EZEDLTRM 注: EZEDLTRM は EZE CNVCM と同等です。マイグレーション・ツールは、どちらの EZE ワードも converseVar.commitOnConverse に変換します。	converseVar.commitOnConverse

日付と時刻の EZE ワード

表の左側の欄には、VisualAge Generator 4.5 の EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。

表 132. 日付と時刻の EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZEDAY	vgVar.currentShortJulianDate
EZEDAYL	vgVar.currentJulianDate
EZEDAYLC	vgVar.currentFormattedJulianDate
EZEDTE	vgVar.currentShortGregorianDate
EZEDTEL	vgVar.currentGregorianDate
EZEDTELC	vgVar.currentFormattedGregorianDate
EZETIM	vgVar.currentFormattedTime

その他のデータの EZE ワード

表の左側の欄には、VisualAge Generator 4.5 の EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。

表 133. その他のデータの EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZE AID	converseVar.eventKey
EZE APP	sysVar.transferName
EZE CNVCM	converseVar.commitOnConverse
EZE CONV T	sysVar.callConversionTable
<i>record</i> .EZEDEST	<i>record</i> . resourceAssociation 注: 修飾は引き続きレコード名です。
EZEDESTP	converseVar.printerAssociation
EZE FEC	vgVar.handleHardIOErrors
EZE LOC	sysVar.remoteSystemID

表 133. その他のデータの EZE ワード (続き)

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZELTERM	<p>テキスト・プログラム用の sysVar.terminalID。</p> <p>VGWebTransaction プログラム用の sysVar.conversationID。</p> <p>注:</p> <ul style="list-style-type: none"> • テキスト・プログラムでは、sysVar.terminalID と sysVar.conversationID の両方が terminalID 情報を提供します。 • VGWebTransaction プログラムでは、sysVar.terminalID と sysVar.conversationID の両方が conversationID 情報を提供します。
EZEMNO	<ul style="list-style-type: none"> • EZEMNO が MOVE または割り当てのターゲットとして使用されている場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> – EZEMNO が 9999 以外の数値リテラルから設定されている場合、EZEMNO は次のようになります。 <pre>converseLib.validationFailed(numericLiteral);</pre> – EZEMNO が数値リテラル 9999 から設定されている場合、EZEMNO は次のようになります。 <pre>converseLib.validationFailed();</pre> – EZEMNO が項目から設定されている場合、EZEMNO は次のようになります。 <pre>if (itemName == 9999) converseLib.validationFailed(); else converseLib.validationFailed(itemName); end</pre> • EZEMNO がその他の場所で使用されている場合は、次のように置き換えられます。 <pre>converseVar.validationMsgNum</pre>

表 133. その他のデータの EZE ワード (続き)

VisualAge Generator 4.5 の EZE ワード	EGL の定義
<p>EZEMSG</p> <p>注: データ項目としての EZEMSG は、マップに配置されている場合のみ存在します。複数のマップに配置されている場合は、EZEMSG を修飾する必要があります。</p>	<p><i>custPrefix</i>EZEMSG</p> <p>注:</p> <ul style="list-style-type: none"> • <i>custPrefix</i> は、ステージ 2 マイグレーション時に指定した RenamingPrefix です。 • <i>custPrefix</i> と EZEMSG の間にドットは入りません。 • EZEMSG が関数の中で使用されている場合、マイグレーション・ツールは、これらの関数の中で EZEMSG によって使用されていたものと同じ修飾を <i>custPrefix</i>EZEMSG にも使用します。例えば、xxxx.EZEMSG は <i>xxxx.custPrefix</i>EZEMSG になります。 • EZEMSG がマップ内で使用されている場合、マイグレーション・ツールは次の処理を行います。 <ul style="list-style-type: none"> – フィールド名を <i>custPrefix</i>EZEMSG に変更します。 – フォームの msgField プロパティを <i>custPrefix</i>EZEMSG に設定します。
EZEORDER	vgVar.handleOverflow
EZEORDER	sysVar.overflowIndicator
EZERCODE	sysVar.returnValue
<p>注: VisualAge Generator では、EZERCODE に対して負の値や 512 を超える値も指定できます。</p>	<p>注: EGL では、sysVar.returnValue に対して負の値や 512 を超える値は指定できません。</p>
EZEREPY	vgVar.handleSysLibraryErrors
EZERT2	vgVar.mqConditionCode
<p>注: VisualAge Generator 4.5 では、EZERT2 は MQ Series アクセス用の条件コードとしてのみ使用されます。</p>	
EZERT8	sysVar.errorCode
<p>注: EZERT8 が設定される対象は、次のとおりです。</p> <ul style="list-style-type: none"> • REPLY オプションが指定されている場合は、CALL ステートメント。 • EZEREPY が 1 に設定されている場合は、EZE システム関数呼び出し。 • シリアル・レコード、索引付きレコード、相対レコード、およびメッセージ・キュー・レコードの入出力オプション。 	<p>注: sysVar.errorCode が設定される状況は、次のとおりです。</p> <ul style="list-style-type: none"> • すべての call ステートメント。 • すべての sysLib システム関数呼び出し。 • 一部の strLib、mathLib、および vgLib システム関数呼び出し • シリアル・レコード、索引付きレコード、相対レコード、およびメッセージ・キュー・レコードの入出力オプション。 <p>EGL では、sysVar.errorCode の値が変更される頻度が VisualAge Generator の場合より高くなります。</p>

表 133. その他のデータの EZE ワード (続き)

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZESEGM	converseVar.segmentedMode
EZESEGTR	sysVar.transactionID
EZESYS	<p>if ステートメントまたは while ステートメント内で EGL の値を使用するには、以下を使用します。</p> <p>sysVar.systemType</p> <p>他のステートメント内で使用するために以前の VAGen の値を取得するには、以下を使用します。</p> <pre>myItem = vgLib.getVAGSysType();</pre> <p>その後、ステートメント内で <i>myItem</i> を使用します。</p> <p>マイグレーション済みの VAGen プログラム内で以前の VAGen の値を使用する必要がある場合は、以下を使用します。</p> <pre>custPrefixEZESYS</pre> <p>ここで、<i>custPrefix</i> はマイグレーションのステージ 2 で指定した名前変更接頭部です。マイグレーション設定「以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)」に基づいて、マイグレーション・ツールは、<i>custPrefixEZESYS</i> のデータ宣言、および以前の VAGen 値にそれを初期化するステートメントを組み込んだり、省略したりします。</p> <p>詳細と起こりうる問題については、142 ページの『EZESYS』を参照してください。</p>
EZETST 注: IF...IN、および MOVEA に対して設定されます。 EZETST は 2 バイトのバイナリーです。	sysVar.arrayIndex 注: arrayIndex は INT (4 バイトのバイナリー) です。
EZEUSR	sysVar.sessionID
EZEUSRID	sysVar.userID

一般的な関数の EZE ワード

表の左側の欄には、VisualAge Generator 4.5 の EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。注記のある場合を除き、引数リストは VisualAge Generator と EGL のどちらでも同じなので、表からは省略しています。

表 134. 一般的な関数の EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL の定義
<p>result = EZEBYTES (<i>itemOrRecord</i>)</p> <p>注: VisualAge Generator の資料には、項目とレコードのみが EZEBYTES の引数として使用できると記述されています。ただし、VisualAge Generator は EZEBYTES の引数としてマップを許容します。</p>	<p>result = sysLib.bytes (<i>itemOrRecordOrForm</i>)</p> <p>注: EGL は、sysLib.bytes() の引数としてのフォームをサポートしています。マイグレーション・ツールは、引数が項目、レコード、またはマップのどれであるかに関係なく、引数を変換します。</p>
EZECOMIT()	<p>sysLib.commit()</p> <p>注: デバッグおよび Java 生成では、SQL をサポートするために JDBC を使用します。JDBC は、単一フェーズ・コミットのみをサポートします。詳しくは、277 ページの『SQL サポートに関する違い』を参照してください。</p>
EZECONV(target,direction,conversionTable)	<p>sysLib.convert()</p> <p>注: 方向は、ConvertDirection 列挙の値 (ConvertDirection.local または ConvertDirection.remote のいずれか) で指定する必要があります。</p>
EZEC10(xxx, yyy, zzz)	sysLib.verifyChkDigitMod10()
EZEC11(xxx, yyy, zzz)	sysLib.verifyChkDigitMod11()
EZEG10(xxx, yyy, zzz)	sysLib.calculateChkDigitMod10()
EZEG11(xxx, yyy, zzz)	sysLib.calculateChkDigitMod11()
EZEPURGE(queueName)	sysLib.purge()
EZEROLLB()	sysLib.rollback()
EZEWAIT(variableName)	<p>sysLib.wait (<i>variableName</i>);</p> <p>注:</p> <ul style="list-style-type: none"> <i>variableName</i> は、秒単位で時間を指定します。 マイグレーション・ツールは、時間を秒数に変換します。詳細と起こりうる問題については、144 ページの『EZEWAIT』を参照してください。

ストリング EZE ワード

表の左側の欄には、VisualAge Generator 4.5 の EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。引数リストは VisualAge Generator と EGL のどちらでも同じなので、表からは省略しています。

表 135. スtring EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZESBLKT	<code>strLib.setBlankTerminator()</code>
EZESCCWS	<code>vgLib.concatenateWithSeparator()</code>
EZESCMPR	<code>vgLib.compareBytes()</code>
EZESCNCT	<code>vgLib.concatenateBytes()</code>
EZESCOPY	<code>vgLib.copyBytes()</code>
EZESFIND	<code>vgLib.findStr()</code>
EZESNULT	<code>strLib.setNullTerminator()</code>
EZESSET	<code>vgLib.setSubStr()</code>
EZESTLEN	<code>strLib.byteLen()</code>
EZESTOKN	<code>strLib.getNextToken()</code>

数学 EZE ワード

表の左側の欄には、VisualAge Generator 4.5 の EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。VisualAgeGenerator では、EZE 数学関数は、次の書式の完全なステートメントとして作成されます。

```
result = EZEMathFunction(argument1, argument2, ... argumentN);
```

EGL では、多くの数学関数が、ステートメントの一部（例えば、**if** または **while** ステートメントのオペランドなど）で使用可能です。したがって、その結果は次のように **mathLib.assign()** 関数を使用して割り当てる必要があります。

```
mathLib.assign(EGLMathFunction(argument1, argument2, ... argumentN), result);
```

次の表は、**mathLib.assign()** 関数を使用して結果を取得する必要がある EGL 数学関数を示しています。その他の場合は、特に注意書きがない限り、対応する数学関数の引数リストは VisualAge Generator と EGL のどちらでも同じなので、表からは省略しています。

表 136. 数学 EZE ワード - 一般的な数学関数

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZEABS	<code>mathLib.assign(mathLib.abs(...), result)</code>
EZECEIL	<code>mathLib.ceiling()</code>
EZEEXP	<code>mathLib.assign(mathLib.exp(...), result)</code>
EZEFLOOR	<code>mathLib.floor()</code>
EZEFREXP	<code>mathLib.assign(mathLib.frexp(...), result)</code>
EZELDEXP	<code>mathLib.assign(mathLib.ldexp(...), result)</code>
EZELOG	<code>mathLib.assign(mathLib.log(...), result)</code>
EZELOG10	<code>mathLib.assign(mathLib.log10(...), result)</code>
EZEMAX	<code>mathLib.assign (mathLib.max (...), result)</code>
EZEMIN	<code>mathLib.assign (mathLib.min(...), result)</code>
EZEMODF	<code>mathLib.assign (mathLib.modf(...), result)</code>

表 136. 数学 EZE ワード - 一般的な数学関数 (続き)

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZENCMPR	<code>vgLib.compareNum()</code>
EZEPOW	<code>mathLib.assign (mathLib.pow(...), result)</code>
EZEPRSCN	<code>mathLib.precision()</code>
EZEROUND	<code>mathLib.round()</code> 注: <code>mathLib.round()</code> は、(R オプションを指定した VAGen assignment ステートメントを置き換えるためにも使用されます。マイグレーション・ツールは、このタイプの assignment ステートメントを次の構文に変換します。 <pre>result = mathLib.round(numericExpression, -mathLib.decimals(result));</pre>
EZESQRT	<code>mathLib.assign(mathLib.sqrt(...), result)</code>

表 137. 数学 EZE ワード - 三角関数

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZEACOS	<code>mathLib.assign(mathLib.acos(...), result)</code>
EZEASIN	<code>mathLib.assign(mathLib.asin(...), result)</code>
EZEATAN	<code>mathLib.assign(mathLib.atan(...), result)</code>
EZEATAN2	<code>mathLib.assign(mathLib.atan2(...), result)</code>
EZECOS	<code>mathLib.assign(mathLib.cos(...), result)</code>
EZECOSH	<code>mathLib.assign(mathLib.cosh(...), result)</code>
EZESIN	<code>mathLib.assign(mathLib.sin(...), result)</code>
EZESINH	<code>mathLib.assign(mathLib.sinh(...), result)</code>
EZETAN	<code>mathLib.assign(mathLib.tan(...), result)</code>
EZETANH	<code>mathLib.assign(mathLib.tanh(...), result)</code>

表 138. 数学 EZE ワード - 浮動小数点数学関数

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZEFLADD	<code>mathLib.assign(vgLib.floatingSum(...), result)</code>
EZEFLDIV	<code>mathLib.assign(vgLib.floatingQuotient(...), result)</code>
EZEFLMOD	<code>mathLib.assign(vgLib.floatingMod(...), result)</code>
EZEFLMUL	<code>mathLib.assign(vgLib.floatingProduct(...), result)</code>
EZEFLSET	<code>mathLib.assign(..., result)</code>
EZEFLSUB	<code>mathLib.assign(vgLib.floatingDifference(...), result)</code>

ユーザー・インターフェースの EZE ワード

表の左側の欄には、VisualAge Generator 4.5 EZE ワードを示します。右側の欄には、マイグレーション・ツールが EZE ワードを EGL に変換した結果を示します。引数リストは VisualAge Generator でも EGL でも同じであるため、表からは省略しています。

表 139. ユーザー・インターフェースの EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZEUIERR	<code>sysLib.setError</code>
EZEUILOC	<code>sysLib.setLocale</code>

オブジェクト・スクリプトの EZE ワード

表 140. オブジェクト・スクリプトの EZE ワード

VisualAge Generator 4.5 の EZE ワード	EGL の定義
EZESCRPT(<i>targetScriptName</i>)	<code>EZE_SCRPT(<i>targetScriptName</i>)</code> EZESCRPT を置換する、対応する EGL 関数はありません。マイグレーション・ツールは意図的に無効な構文を作成し、エラー・メッセージを出します。

サービス・ルーチン

サービス・ルーチンに関するセクションは、次の表で構成されています。

- サービス・ルーチン - 一般構文 (415 ページの表 141)
- サービス・ルーチン - VisualAge Generator ルーチンおよび同等な EGL ルーチン (416 ページの表 142)

表 141. サービス・ルーチン - 一般構文

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<code>CALL serviceRoutine [argumentList] ;</code>	<code>eglSystemLibrary.EGLSystemFunction ([argumentList]);</code> 注: EGL システム関数は、VisualAge Generator の場合と同じ引数リストを使用します。	特別な考慮事項なし。
<code>CALL serviceRoutine [argumentList] (REPLY ;</code>	<code>try eglSystemLibrary.EGLSystemFunction ([argumentList]); end;</code> 注: EGL システム関数は、VisualAge Generator の場合と同じ引数リストを使用します。	VisualAge Generator で (REPLY オプションが組み込まれていた場合、マイグレーション・ツールは try ブロックを組み込みます。

表 142. サービス・ルーチン - VisualAge Generator ルーチンおよび同等な EGL ルーチン

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
CALL AUDIT	<code>sysLib.audit()</code>	特別な考慮事項なし。
CALL COMMIT	<code>sysLib.commit()</code>	特別な考慮事項なし。
CALL CREATX	<code>vgLib.startTransaction()</code>	特別な考慮事項なし。
CALL CSPTDLI	<code>vgLib.VGTDLI()</code> 注: EGL は EGLTDLI および AIBTDLI もサポートしています。	特別な考慮事項なし。
CALL EZCHART	<code>call "EZCHART" [arguments] { isExternal = YES } ;</code> 注: EGL に EZCHART の置換表現はありません。	VAGen マイグレーション・ツールは、EZCHART を外部定義プログラムへの呼び出しに変換します。 VisualAge Generator で REPLY オプションが指定されていた場合、マイグレーション・ツールは try ブロック内に call ステートメントをネストします。
CALL RESET	<code>sysLib.rollback()</code>	特別な考慮事項なし。

PSB

VisualAge Generator では、PSB はパーツ型です。PSB は、IMS または DL/I PSB の情報のサブセットを含みます。TP PCB に関連付けされた名前はありません。DB と GSAM PCB に関連付けするデータベース名は、固有である必要はありません。DL/I I/O 関数は、データベース名か、PCB 番号のいずれかによって、PSB 内の特定の PCB を参照することができます。ステートメントおよびプログラムの呼び出し先パラメーター・リストで、EZEDLPCB 特殊機能語を使用して番号で PCB を参照することができます。I/O PCB は VAGen PSB に明示的にインクルードされていませんが、IMSVS、IMSBMP、および MVS Batch ターゲット環境では常に存在します。I/O PCB は PCB 番号 0 と見なされます。

EGL では、PSB はレコード・パーツ型のサブタイプです。PSBRecord は、非構造化レコードです。PSBRecord 内のそれぞれの PCB 変数は固有でなければなりません。DL/I I/O 関数は、PSBRecord 内で PCB 変数に与えられた名前を使用して、特定の PCB を参照することができます。同様に、ステートメントおよびプログラムのパラメーター・リストでは、PSB 内で PCB 変数に与えられた名前を使用します。

マイグレーション・ツールは、VAGen PSB 内の数値位置を基に、各 TP PCB の変数名を作成します。このツールは、VAGen PSB、PCB のタイプを指示するカスタマー指定サフィックス、および、必要な場合、固有変数名を作成する番号にあるデータベース名の組み合わせを使用して、それぞれの DB または GSAM PCB ごとに変数名を作成します。このツールはまた、DB および GSAM PCB を再定義する変数を作成します。再定義変数は、VAGen PSB 内の PCB の数値位置を基にしています。これによりマイグレーション・ツールは、マイグレーション実行中にどちらの変数 (データベース名または PCB 番号) も使用可能になります。

表 143. PSB

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>PSB 情報:</p> <ul style="list-style-type: none"> 名前 PCB 情報は PCB タイプにより異なります <ul style="list-style-type: none"> 数値 型 <ul style="list-style-type: none"> TP DB GSAM データベース セグメント 親 索引キー <p>注:</p> <ul style="list-style-type: none"> PSB は分離パーツ型です。 I/O PCB は明示的に指定されませんが、IMSVS、IMS BMP、および MVS Batch 環境では DL/I PSB 内になければなりません。 	<p>EGL 構文の例:</p> <pre>Record psbName type PSBRecord {defaultPSBName = "originalPSBName"} iopcb IO_PCBRecord; { @PCB { pcbType=PCBKind.TP }}; pcb0 IO_PCBRecord { redefines=iopcb }; [otherPCBInformation] end // end psbName</pre> <p>注:</p> <ul style="list-style-type: none"> PSBRecord はレコード・ステレオタイプです。 I/O PCB は明示的に指定しなければなりません。 EGL は @PCB 複合プロパティを使用して、PCB タイプを指定します。 EGL は次のレコードに関するレコード定義を提供します。 <ul style="list-style-type: none"> IO_PCBRecord ALT_PCBRecord DB_PCBRecord GSAM_PCBRecord EGL は、CICS 環境用の生成中には、I/O PCB 変数を見捨て (除去) します。 	<p>予約語と競合すること、または名前の先頭が # または @ シンボルであることが原因で、PSB の名前変更が必要な場合、マイグレーション・ツールは defaultPSBName プロパティのみをインクルードします。</p> <p>マイグレーション・ツールは常にすべての PSBRecord に変数 iopcb と pcb0 再定義を追加します。</p>
<p>PCB タイプ - TP</p> <ul style="list-style-type: none"> 数値 	<pre>ELAALT ALT_PCBRecord {@PCB { pcbType = PCBKind.TP }}; pcb1 ALT_PCBRecord { redefines = ELAALT }; ELAEXP ALT_PCBRecord {@PCB { pcbType = PCBKind.TP }}; pcb2 ALT_PCBRecord { redefines = ELAEXP }; pcbn ALT_PCBRecord { @PCB { pcbType = PCBKind.TP }};</pre> <p>注:</p> <ul style="list-style-type: none"> EGL は CICS 環境用の生成中には、代替 PCB 変数を見捨て (除去) します。 	<p>マイグレーション・ツールは、VAGen PSB の最初の 2 つの TP PCB の名前として、ELAALT と ELAEXP を使用します。マイグレーション・ツールはまた、番号で参照ができるようにこれらの 2 つの TP PCB に関する再定義を作成します。</p> <p>追加の TP PCB がある場合、マイグレーション・ツールは、PSB 内の PCB 番号を基に TP PCB の変数名を作成します。これによりマイグレーション・ツールは、EZEDLPCB[n] の置き換えとして pcbn を使用できるようになります。</p>

表 143. PSB (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>PCB タイプ - DB</p> <ul style="list-style-type: none"> 数値 データベース セグメント 親 索引キー <p>注:</p> <ul style="list-style-type: none"> PSB の複数の PCB に同じデータベース名を使用できます。 	<pre>DBName_dbSuffix DB_PCBRecord { @PCB { pcbType = PCBKind.DB, pcbName = "DBName", secondaryIndex = "indexKeyName", secondaryIndexItem = "renamedIndexKey", hierarchy = [@Relationship { segmentRecord = segmentName, parentRecord = parentSegmentName }] } }; pcbn DB_PCBRecord { redefines = DBName_dbSuffix };</pre> <p>注:</p> <ul style="list-style-type: none"> PSB はレコードであり、それぞれのデータベース名がレコード内でフィールドになるため、それぞれのデータベース名は固有でなければなりません。 	<p>マイグレーション・ツールは、サフィックスのついた VAGen データベース名に基づき、DB PCB の変数名を作成します。サフィックスは、ステージ 2 の「VAGen マイグレーション・データベースの I/O 設定」によって指定できます。</p> <p>必要な場合、マイグレーション・ツールは、固有の DBName を作成するために番号をインクルードします。例: DBName_n_dbSuffix。ここでは <i>n</i> は <i>pcbn</i> 再定義の番号と同じです。</p> <p>マイグレーション・ツールは、pcbName プロパティを、対応する VAGen PCB のデータベース名に設定します。IMS または DL/I PSB で VAGen データベース名を PCBNAME として使用している場合は、これによって、EGL デバッガーを使用したテストが容易になります。</p> <p>EGL 予約語との競合、または名前の先頭が # または @ シンボルであることが原因で、VAGen 索引キーの名前変更が必要な場合、マイグレーション・ツールは secondaryIndexItem プロパティのみをインクルードします。</p> <p>マイグレーション・ツールはまた、VAGen PSB 内の PCB 番号に基づいて DB PCB の再定義を作成します。</p>

表 143. PSB (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
PCB タイプ - GSAM • 数値 • データベース	<pre>DBName_gsamSuffix GSAM_PCBRecord { @PCB { pcbType = PCBKind.GSAM }}; pcbn GSAM_PCBRecord { redefines = DBName_gsamSuffix };</pre> <p>注:</p> <ul style="list-style-type: none"> PSB はレコードであり、それぞれのデータベース名がレコード内でフィールドになるため、それぞれの GSAM データベース名は固有でなければなりません。 EGL は CICS 環境用の生成中には、GSAM PCB 変数を無視 (除去) します。 	<p>マイグレーション・ツールは、サフィックスのついた VAGen データベース名に基づき、GSAM PCB の変数名を作成します。サフィックスは、ステージ 2 の「VAGen マイグレーション・データベースの I/O 設定」によって指定できます。</p> <p>必要な場合、マイグレーション・ツールは、固有の DBName を作成するために番号をインクルードします。例: DBName_n_gsamSuffix。ここでは <i>n</i> は <i>pcbn</i> 再定義の番号と同じです。</p> <p>マイグレーション・ツールはまた、VAGen PSB 内の PCB 番号に基づいて GSAM PCB の再定義を作成します。</p>

制御パーツ

VisualAge Generator では、制御パーツはフリー・フォームのテキスト・エディターを使用して入力されます。制御パーツは、生成時に実際に使用されるまでは検証されません。大文字または小文字は区別されません。

EGL では、制御パーツは .eglbld ファイルに XML 表記で格納され、制御パーツのタイプごとに専用のエディターを使用して作成されます。EGL の場合、大文字と小文字は区別されます。

このセクションの表では、VisualAge Generator のフリー・フォームのテキスト・エディターに入力する情報と、EGL で使用される XML タグまたは属性値を比較しています。表にはタグまたは属性値のみが示されており、実際の XML 構文は示されていません。

注:

- マイグレーション・ツールは、対応する EGL 置換表現は存在しないが EGL に必要な関連情報の判別に役立つ可能性がある生成オプション、リンクージ・テーブル・オプション、およびリソース関連オプションを、コメントとして組み込みます。例えば、マイグレーション・ツールは、生成オプション /jspreldir をコメントとして組み込みます。これらのコメントは、通常の EGL ビルド・パーツ・エディターを使用する際は表示されません。テキスト・エディターを使用してファイルを開くと、これらのコメントを表示できます。
- マイグレーション・ツールは、情報が現在または将来の EGL オプションの判別に使用できない場合は、対応する EGL の置換表現がない生成オプションを除去します。例えば、/lineinfo の置換表現は存在しません。このオプションは、IBM サポートによる VAGen 生成プログラムのデバッグを支援する

ためのものでした。このオプションは、EGL 生成プログラムには使用できないので、マイグレーション・ツールはこのオプションをコメントとして組み込みません。

- マイグレーション・ツールは、以下の場合を除いて制御パーツの名前を変更しません。
 - マイグレーション・ツールは、バインド制御パーツ名の末尾から .BND 接尾部を除去します。
 - マイグレーション・ツールは、リンク・エディット・パーツ名の末尾から .LKG 接尾部を除去します。
 - マイグレーション・ツールは、制御パーツ名に含まれるその他のドットを下線に変更します。
 - さらにこのツールは、/OPTIONS、/RESOURCE、および /LINKEDIT 生成オプション内で参照されている制御パーツ名のドットを下線に変更します。
 - マイグレーション・ツールは、パーツ名が EGL 予約語と競合している場合は、エラー・メッセージを出します。

制御パーツに関するセクションは、次の表で構成されています。

- 制御パーツの一般情報 (421 ページの表 144)
- 生成オプション (421 ページの表 145)
- 生成オプション: 変換テーブルの値 (441 ページの表 146)
- リンケージおよびリソース関連: 変換テーブル名、443 ページの表 147
- :callLink のリンケージ・テーブル・オプション (444 ページの表 148)
- :filelink のリンケージ・テーブル・オプション (448 ページの表 149)
- :crtxlink のリンケージ・テーブル・オプション (449 ページの表 150)
- :dxfrlink のリンケージ・テーブル・オプション (450 ページの表 151)
- リソース関連 (452 ページの表 152)
- リンク・エディット・オプション (457 ページの表 153)
- バインド制御 (458 ページの表 154)
- パーツ関連のシンボリック・パラメーター (458 ページの表 155)
- ファイル関連のシンボリック・パラメーター (460 ページの表 156)
- ユーザー定義のシンボリック・パラメーター (460 ページの表 157)

表 144. 制御パーツの一般情報

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VAGen 制御パーツ名:</p> <ul style="list-style-type: none"> • 名前にピリオド (.) を含むことができます。 • バインド・パーツとリンク・エディット・パーツの場合、名前のうち最初のピリオドの後にある部分はすべて接尾部として扱われます。接尾部は、/bind と /link edit の生成オプションの後に指定できます。 	<p>EGL ビルド・パーツ:</p> <ul style="list-style-type: none"> • ピリオド (.) は、ビルド・パーツ名の中では無効です。 	<p>マイグレーション・ツールは、ピリオド (.) を下線 (_) に変更します。</p>
<p>VAGen 制御パーツのタグと値の中では、大文字と小文字は区別されません。</p>	<p>EGL 制御パーツのタグと値の中では、大文字と小文字が区別されます。</p>	<p>マイグレーション・ツールは、制御パーツのタグと値を EGL に必要な正しい大/小文字に変換します。</p>

生成オプション・パーツ

表 145. 生成オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VAGen 生成オプション・パーツ:</p> <ul style="list-style-type: none"> • 1 つ以上の生成オプションを指定します。 • /options 生成オプションを使用してチェーニングできます。 • 生成時に、ワークスペースに含まれる他の制御パーツを参照できます。参照される制御パーツは、生成オプション・パーツの関連パーツとは見なされません。 	<p>EGL ビルド記述子パーツ:</p> <ul style="list-style-type: none"> • 1 つ以上のビルド記述子オプションを指定します。 • nextBuildDescriptor ビルド記述子オプションを使用してチェーニングできます。 • 以下のいずれかの基準が満たされる場合に限り、他のビルド・パーツを参照できます。 <ul style="list-style-type: none"> – ビルド・パーツが、同じ .eglbld ファイルに含まれている。 – ビルド・パーツが、.eglbld ファイルによってインポートされるファイル内にある。 	<p>VAGen 制御パーツがすべて同じ VisualAge Java パッケージ、または VisualAge Smalltalk アプリケーション内にある場合、制御パーツは同じ .eglbld ファイルに配置されます。この場合、import ステートメントは必要ありません。</p> <p>VAGen 制御パーツが異なる VisualAge Java パッケージまたは VisualAge Smalltalk アプリケーション内にある場合、マイグレーション・ツールは import ステートメントを作成しません。 import ステートメントを追加する必要があります。他の制御パーツへの参照を EGL が解決できない場合は、EGL 検査によって「問題」ビューにエラー・メッセージが表示されます。</p>
<p>VAGen 生成オプションの値は、ディレクトリー名またはファイル名に特殊文字が含まれている場合に限り、二重引用符で囲まれます。</p>	<p>EGL ビルド記述子オプションの値は、二重引用符で囲む必要があります。ただし、EGL ビルド・パーツ・エディターを使用する場合、エディターは引用符を XML ソースに自動的に挿入します。エディターに引用符は表示されません。</p>	<p>マイグレーション・ツールは、.eglbld ファイルの XML ソースを作成する際に、引用符を自動的に組み込みます。</p>

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>多くの VAGen 生成オプションは、生成オプションの肯定または否定を反映して /xxxx または /noxxxx のように指定できます。次に例を示します。</p> <ul style="list-style-type: none"> • /prep は、準備ステップの生成直後に自動的に開始するように指示します。 • /noprep は、準備ステップを後で実行する予定であるため、そのステップを自動的に開始しないように指示します。 	<p>多くの EGL ビルド記述子オプションは、ビルド記述子オプションの肯定または否定を反映して xxxx="YES" または xxxx="NO" のように指定できます。次に例を示します。</p> <ul style="list-style-type: none"> • prep = "YES" は、準備ステップの生成直後に自動的に開始するように指示します。 • prep = "NO" は、準備ステップを後で実行する予定であるため、そのステップを自動的に開始しないように指示します。 	<p>マイグレーション・ツールは、オプションを以下のように処理します。</p> <ul style="list-style-type: none"> • 他に指示のない限り、マイグレーション・ツールは /xxxx を対応する xxxx="YES" オプションに変換します。 • 他に指示のない限り、マイグレーション・ツールは /noxxxx を対応する xxxx="NO" オプションに変換します。
/ansisql	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/bidicontable=xxxx	bidConversionTable ="xxxx"	特別な考慮事項なし。
<p>/bind=xxxx</p> <p>VisualAge Generator の場合、xxxx はバインド・パーツの接尾部です。プログラムのバインド・パーツの名前は、pgmname.xxxx です。ここで、xxxx は /bind オプションによって指定される接尾部です。/bind=suffix は、以下のいずれかの理由で指定されることがあります。</p> <ul style="list-style-type: none"> • プログラムを複数の DB2 プランにバインドするため、プログラムに特殊なバインドが必要。 • VisualAge Generator では、プログラムとまったく同じ名前のバインド・パーツを容易に作成できない。 	<p>bind オプションの意味は、VisualAge Generator とは異なります。EGL の場合、xxxx はバインド・パーツのフルネームです。bind オプションは、バインド・パーツ名がプログラムと異なる場合にのみ指定する必要があります。ほとんどの場合、プログラムとバインド・パーツの名前は同じなので、bind オプションを組み込む必要はありません。</p> <p>複数のランタイム環境用に同じプログラムを生成し、それぞれの環境ごとに特殊なバインド・コマンドが必要な場合のみ、bind オプションが必要です。</p> <p>bind オプションのもう 1 つの用途は、バインド・コマンドのテンプレートを含むパーツ名の指定です。プロジェクト管理担当者または DBA は、メンバー固有のパラメーター用に置換可能な SYMPARMS を組み込んで、バインド・パーツを定義できます。EGL の bind オプションを使用して、このテンプレート・パーツを指定できます。それぞれのプログラムごとにパッケージをバインドする場合は、この技法が役に立ちます。</p>	<p>VisualAge Generator と EGL では bind オプションの値の意味が異なるので、マイグレーション・ツールはこのオプションをマイグレーションできません。マイグレーション・ツールは、/bind をコメントとして組み込みます。</p> <p>バインド・ビルド記述子のオプションの設定方法について詳しくは、以下のセクションを参照してください。</p> <ul style="list-style-type: none"> • 254 ページの『テンプレートとして使用するバインド制御パーツの設定』。 • 256 ページの『プログラム固有のバインド制御パーツの設定』。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/checktype=xxx xxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • none • low • all 	checkType="xxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> • NONE • LOW • ALL 	特別な考慮事項なし。
/cicsdbcs	サポートされていません。	現在、サポートされるすべての CICS 変換プログラムに DBCS のサポートが組み込まれているので、マイグレーション・ツールはこのオプションをコメントとして組み込みません。
/cicsentries=xxx xxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • none • rdo • macro 	cicsEntries="xxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> • NONE • RDO • MACRO 	特別な考慮事項なし。
/cobollevel=le vs	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p><code>commentlevel=n</code> または <code>/commentlevel=commentText</code></p> <p><code>n</code> または <code>commentText</code> は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • 0 または minimum • 1 または info • 2 または logic • 3 または data • 4 または statements <p>注:</p> <ul style="list-style-type: none"> • 数値または同等な <code>commentText</code> のどちらかを指定できます。 • 0 = 生成オプションのコメントのみ • 1 = 別名、標準的な生成情報 • 2 = プログラムとテーブルのプロローグ、および関数の説明 • 3 = レコードのプロローグとデータ項目の説明 • 4 = ソース・ステートメントとコメント • C++ の場合、有効な値は 0 = none と 1 = comments のみです。 	<p><code>commentLevel="n"</code></p> <p>以下の EGL 値は、VAGen 値に対応しています。</p> <ul style="list-style-type: none"> • 0 • 1 • 1 • 1 • 1 <p>注:</p> <ul style="list-style-type: none"> • 0 = コメントなし • 1 = コメントが含まれる 	<p>マイグレーション・ツールは、<code>/commentlevel=0</code> または minimum を 0 にマイグレーションし、他のすべての値を 1 にマイグレーションします。</p>
<p><code>/configmapname="xxxx"</code></p> <p><code>xxxx</code> は、VisualAge Smalltalk 構成マップの名前です。</p>	<p>サポートされていません。</p>	<p>マイグレーション・ツールは、このオプションをコメントとして組み込みます。これは、このオプションが、ソース・コード・リポジトリに 1 単位としてチェックインする必要がある、関連した EGL プロジェクトのグループの判別に役立つ可能性があるからです。</p>
<p><code>/configmapversion="xxxx"</code></p> <p><code>xxxx</code> は、<code>/configmapname</code> によって指定される、VisualAge Smalltalk 構成マップのバージョン名です。</p>	<p>サポートされていません。</p>	<p>マイグレーション・ツールは、このオプションをコメントとして組み込みます。これは、このオプションが、ソース・コード・リポジトリに 1 単位としてチェックインする必要がある、関連した EGL プロジェクトのグループの判別に役立つ可能性があるからです。</p>

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/contable=xxxx xxxx は、変換テーブルの名前です。	clientCodeSet = "yyyy" serverCodeSet = "zzzz" yyyy と zzzz は、それぞれクライアントとサーバーの変換テーブルの名前です。	マイグレーション・ツールは、VAGen /contable 生成オプションから clientCodeSet および serverCodeSet オプションの両方を設定します。VAGen と EGL の値の対応については、441 ページの表 146 を参照してください。/contable=xxxx の値が 441 ページの表 146 に示されていない場合、マイグレーション・ツールは clientCodeSet と serverCode の両方を xxxx に設定します。
/createdds	genDDSFile = "YES" "NO" 注: これは ISERIESC 用です。	特別な考慮事項なし。
/currency=xxx (1 文字から 3 文字)	currencySymbol = "xxx"	特別な考慮事項なし。
/data = 24 31	data = "24" "31"	特別な考慮事項なし。
/dbms=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • db2 • oracle • odbc 注: VisualAge Generator では、Oracle および ODBC は特定のワークステーションのプラットフォームでのみサポートされます。	dbms = "xxxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> • DB2 • ORACLE • DB2 注: <ul style="list-style-type: none"> • EGL では、Oracle は Java 生成を使用する場合にのみサポートされます。 • EGL の Java 生成機能は、ODBC ではなく JDBC をサポートします。 • EGL は、以下のデータベースもサポートします。 <ul style="list-style-type: none"> – CLOUDSCAPE – DERBY – INFORMIX – SQLSERVER 	マイグレーション・ツールは、odbc を DB2 に変更し、警告メッセージを出します。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/dbpassword=xxxx	sqlPassword = "xxxx"	マイグレーション・ツールは、VAGen の /dbpassword オプションと /sqlpassword オプションを、EGL の sqlPassword オプションにマージします。生成オプション・パーツに /dbpassword と /sqlpassword の両方が含まれている場合、マイグレーション・ツールは sqlPassword を 2 回組み込みます。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。
/dbuser=xxxx	sqlIID = "xxxx"	マイグレーション・ツールは、VAGen の /dbuser オプションと /sqlIID オプションを、EGL の sqlIID オプションにマージします。生成オプション・パーツに /dbuser と /sqlIID の両方が含まれている場合、マイグレーション・ツールは sqlIID を 2 回組み込みます。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。
/debugtrace	debugTrace = "YES" "NO"	特別な考慮事項なし。
/destaccount=xxxx	非サポート	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/destdir=xxxx	destDirectory = "xxxx"	特別な考慮事項なし。
/desthost=xxxx	destHost = "xxxx"	特別な考慮事項なし。
/destlib=xxxx	destLibrary = "xxxx" 注: これは ISERIESC 用です。	特別な考慮事項なし。
/destpassword=xxxx	destPassword = "xxxx"	特別な考慮事項なし。
/destuid=xxxx	destUserID = "xxxx"	特別な考慮事項なし。
/dxfrcancel	cancelAfterTransfer = "YES" "NO"	特別な考慮事項なし。
/dxfrxctl	useXctlForTransfer = "YES" "NO"	特別な考慮事項なし。
/ejbgroup=xxxx	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/endcommarea	endCommarea = "YES" "NO"	特別な考慮事項なし。
/errdest=xxxx	errorDestination = "xxxx" 注: これは IMS 用です。	特別な考慮事項なし。
/fastpath	imsFastPath = "YES" "NO" 注: これは IMS 用です。	特別な考慮事項なし。
/fold	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/ftptranslationcmddbs=xxxx	非サポート。EGL は、ホストへのファイル転送用に TCP/IP のみをサポートしています。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/ftptranslationcmdsbc=xxxx	非サポート。EGL は、ホストへのファイル転送用に TCP/IP のみをサポートしています。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/genauthortimevalues /nogenauthortimevalues	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/genhelpmaps	genHelpFormGroup = "YES" "NO"	特別な考慮事項なし。
/genmaps	genFormGroup = "YES" "NO"	特別な考慮事項なし。
/genout=xxxx	genDirectory = "xxxx"	<p>マイグレーション・ツールは、/genout を変換してその結果をオリジナルのビルド記述子と、secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツの両方に置きます。</p> <p>Java を生成する場合は、genProject ビルド記述子オプションを genDirectory オプションに追加して、あるいはその代わりとして指定することが必要になる場合があります。</p> <p>genProject は、次の場合は必須です。</p> <ul style="list-style-type: none"> • HP-UX または SOLARIS 向けに生成する場合 • VGWebTransactions または VGUI レコードを生成する場合
/genproperties /nogenproperties	<p>genProperties = "GLOBAL" genProperties = "NO"</p> <p>EGL では、genProperties = "PROGRAM" も使用できます。</p>	マイグレーション・ツールは、/genproperties を EGL の genProperties = "GLOBAL" に変換します。これは、生成結果の点でこれが最も近い値だからです。
/genresourcebundle	genResourceBundle = "YES" "NO"	マイグレーション・ツールは、/genresourcebundle を変換してその結果をオリジナルのビルド記述子と、 secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツの両方に置きます。
/genret	genReturnImmediate = "YES" "NO"	特別な考慮事項なし。
/gentables	genDataTables = "YES" "NO"	特別な考慮事項なし。
/genuirecords	genVGUIRecords = "YES" "NO"	特別な考慮事項なし。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/groupname=xxxx	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/inedit=all /inedit=inonly	validateOnlyIfModified = "NO" validateOnlyIfModified = "YES"	特別な考慮事項なし。
/initaddws VisualAge Generator の場合: <ul style="list-style-type: none"> このオプションは、メイン・プログラムと呼び出し先プログラムの両方に適用されます。 1 次作業用ストレージ・レコードは、常に初期化されます。/initaddws 生成オプションは、テーブルおよび追加レコードのリストに指定されている、その他の作業用ストレージ・レコードの初期化を行います。 	initNonIODataOnCall = "YES" "NO" EGL では、このオプションは呼び出し先プログラムにのみ適用されます。	特別な考慮事項なし。
/initrecd VisualAge Generator では、このオプションは、メイン・プログラムと呼び出し先プログラムの両方に適用されます。	initIORecordsOnCall = "YES" "NO" EGL では、このオプションは呼び出し先プログラムにのみ適用されます。	特別な考慮事項なし。
/javadestdir=xxxx	destDirectory = "xxxx"	マイグレーション・ツールは、/javadestdir を変換してその結果を secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツに置きます。
/javadesthost=xxxx	destHost = "xxxx"	マイグレーション・ツールは、/javadesthost を変換してその結果を secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツに置きます。
/javadestpassword=xxxx	destPassword = "xxxx"	マイグレーション・ツールは、/javadestpassword を変換してその結果を secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツに置きます。
/javadestuid=xxxx	destUserID = "xxxx"	マイグレーション・ツールは、/javadestuid を変換してその結果を secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツに置きます。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/jvasystem=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • AIX • LINUX • OS2 • OS390 • OS400 • SOLARIS • WINNT 	system = "xxxx" 以下の EGL 値は、VAGen 値に対応します。 <ul style="list-style-type: none"> • AIX • LINUX • 非サポート • USS • ISERIESJ • SOLARIS • WIN 	マイグレーション・ツールは、サポートされている /jvasystem の値を変換してその結果を secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツに置きます。 このツールは、非サポートの値をコメントとしてオリジナルのビルド記述子パーツに組み込みます。
/jobcard=xxxx	非サポート以下のような、同等の機能が提供されています。 <ul style="list-style-type: none"> • z/OS および iSeries ビルド・サーバーが jobcard を処理します。これらの環境では、JOB CARD シンボリック・パラメーターは無視されます。 • VSE は、JOB CARD シンボリック・パラメーターをサポートします。 	マイグレーション・ツールは、このオプションを JOBCARD シンボリック・パラメーターに変換します。
/jobname=xxxx	非サポート以下のような、同等の機能が提供されています。 <ul style="list-style-type: none"> • z/OS および iSeries では、\$USERID をビルド・スクリプトのジョブ名として使用できます。EGL 生成機能は、destUserID ビルド記述子オプションの値と番号を連結して得られる固有のジョブ名を使用して、\$USERID を置き換えます。これらの環境では、JOBNAME シンボリック・パラメーターは無視されます。 • VSE は、JOBNAME シンボリック・パラメーターをサポートします。 	マイグレーション・ツールは、このオプションを JOBNAM シンボリック・パラメーターに変換します。
/jspreldir="xxxx"	非サポート	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/leftjust	leftAlign = "YES" "NO"	特別な考慮事項なし。
/lineinfo	サポートされていません。	このオプションは IBM サポートが VAGen 生成プログラムをデバッグする場合にのみ意味があるので、マイグレーション・ツールはこのオプションをコメントとして組み込みません。生成される COBOL に影響はありません。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/lines=nn	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/linkage=xxxx xxxx は、VAGen リンケージ・テーブル・パーツの名前です。	linkage = "xxxx" xxxx は、EGL リンケージ・オプション・パーツの名前です。	特別な考慮事項なし。
/linkedit=xxxx VisualAge Generator の場合、xxxx はリンク・エディット・パーツの接尾部です。プログラムのリンク・エディット・パーツの名前は、pgmname.xxxx です。ここで、xxxx は /linkedit オプションによって指定される接尾部です。以下のいずれかの理由で、/linkedit=suffix を指定している場合があります。 <ul style="list-style-type: none"> PL/I への静的リンク・エディットを行う場合など、特殊なリンク・エディットがプログラムに必要。 VisualAge Generator では、プログラムとまったく同じ名前のリンク・エディット・パーツを容易に作成できない。 	linkEdit = "xxxx" linkEdit オプションの意味は、VisualAge Generator とは異なります。EGL の場合、xxxx はリンク・エディット・パーツのフルネームです。 linkEdit オプションは、リンク・エディット・パーツ名がプログラムと異なる場合にのみ指定する必要があります。ほとんどの場合、プログラムとリンク・エディット・パーツの名前は同じなので、 linkEdit オプションを組み込む必要はありません。 複数のランタイム環境用に同じプログラムを生成し、それぞれの環境ごとに特殊なリンク・エディット・コマンドが必要な場合にのみ、 linkEdit オプションが必要です。	VisualAge Generator と EGL ではリンク・エディット・オプションの値の意味が異なるので、マイグレーション・ツールはこのオプションをマイグレーションできません。マイグレーション・ツールは、/linkedit をコメントとして組み込みます。 詳しくは、256 ページの『リンク・エディット・コマンドの検討』を参照してください。
/listing /listingonerror /nolisting 注: これは 3 者択一のスイッチです。	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/locvalid	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/log=xx OR /nolog	imsLogID = "xx" OR include /nolog as a comment 注: これは IMS 用です。	マイグレーション・ツールは、このオプションを以下のように処理します。 <ul style="list-style-type: none"> /log=xx は imsLogID = "xx" に変換されます。 /nolog はコメントに変換されます。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<code>/math=xxxxx</code> xxxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> cobol cspae 	<code>math = "xxxxx"</code> 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> COBOL CSPAЕ 	特別な考慮事項なし。
<code>/mfsdev = ('deviceName', 'MFSInfo', 'eAI')</code> 注: <ul style="list-style-type: none"> VAGen <i>deviceName</i> が使用されます。 <i>MFSInfo</i> は、VAGen 装置で使用するのに対応する MFS 情報を提供します。 <i>eAI</i> は、拡張属性情報を提供します。 <i>eAI</i> の値は、EATTR、NOEATTR、および NCD です。 複数の <i>MFSInfo</i> および <i>eAI</i> の値を、単一の <i>deviceName</i> に提供することができます。 この生成オプションについては詳しくは、「<i>VisualAge Generator Server Guide for MVS, VSE, and VM</i>」を参照してください。 	<code><mfsDevice width="nn", height="nn", devStmntParms="MFSInfo", extendedAttributes="eAI" /></code> 注: <ul style="list-style-type: none"> EGL 装置サイズ (width および height) を使用します。 <i>MFSInfo</i> および <i>eAI</i> は、VisualAge Generator の場合と同じ情報を提供します。 <i>eAI</i> の値は、YES、NO、および NCD です。 複数の <i>MFSInfo</i> および <i>eAI</i> の値を、単一の装置サイズに提供することができます。 このビルド記述子オプションについては詳しくは、「<i>EGL 生成ガイド</i>」を参照してください。 	マイグレーション・ツールは、VAGen <i>deviceName</i> を対応する width および height に変換します。装置名とサイズの関係については、338 ページの表 97 を参照してください。2 つの <i>deviceName</i> が同一の width および height に変換され、 <i>MFSInfo</i> および <i>eAI</i> に同一の値が指定されている場合、マイグレーション・ツールは 1 つのエントリのみを組み込みます。 マイグレーション・ツールは、 <i>MFSInfo</i> の値を変更しません。 マイグレーション・ツールは、 <i>eAI</i> の値を対応する EGL 値に変換します。
<code>/mfseattr /nomfseattr /mfseattrncd</code> VisualAge Generator の場合、これら 3 つのオプションは、MFS フォーマットのマップの拡張属性サポートを生成するために必要な情報を提供する、3 者択一のスイッチです。	<code>mfsExtendedAttr = "YES" mfsExtendedAttr = "NO" mfsExtendedAttr = "NCD"</code> 注: これは IMS 用です。	特別な考慮事項なし。
<code>/mfsignore</code>	<code>mfsIgnore = "YES" "NO"</code> 注: これは IMS 用です。	特別な考慮事項なし。
<code>/mfstest</code>	<code>mfsUseTestLibrary = "YES" "NO"</code> 注: これは IMS 用です。	特別な考慮事項なし。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/msgtableprefix=xxxx VisualAge Generator の場合、メッセージ・テーブル接頭部はプログラム上で指定されます。UI レコードを単独で生成する場合は、生成時にメッセージ・テーブル接頭部を指定する必要があります。	msgTablePrefix = "xxxx" EGL では、 msgTablePrefix に関して VisualAge Generator と同じ考慮事項が適用されます。	マイグレーション・ツールは、/msgtableprefix を変換して、その結果をオリジナルのビルド記述子と、 secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツの両方に置きます。 VGUI レコードを使用するプログラムを生成しないで、VGUI レコードのみを生成する場合は、メッセージ・テーブルの接頭部が付いたパッケージ名を組み込む必要があります (例えば、 msgTablePrefix = "packageName.prefixID")。
/msp=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • all • gsam • mfs • seq 	formServicePgmType = "xxxx" 以下の EGL 値は、VAGen 値に対応します。 <ul style="list-style-type: none"> • ALL • GSAM • MFS • SEQ 注: IMS BMP、および z/OS パッチ用です。	特別な考慮事項なし。
/nullfill	fillWithNulls = "YES" "NO"	特別な考慮事項なし。
/numovfl	checkNumericOverflow = "YES" "NO"	特別な考慮事項なし。
/options=xxxx xxxx は、別の VAGen 生成オプション・パーツの名前です。	nextBuildDescriptor = "xxxx" xxxx は、別の EGL ビルド記述子パーツの名前です。	特別な考慮事項なし。
/packagename=xxxx VisualAge Generator では、/packagename 生成オプションは、Java、Java ラッパー、またはウェブ・トランザクション用の Java コンポーネントを生成する場合に使用されます。	wrapperPackageName = "xxxx" EGL では、 wrapperPackageName は enableJavaWrapperGen ビルド記述子オプションが "ONLY" または "YES" に設定されている場合にのみ使用されます。このように設定されていない場合、 wrapperPackageName は無視されます。	特別な考慮事項なし。
/possign=x x は、以下のいずれかの値です。 <ul style="list-style-type: none"> • f • c 	positiveSignIndicator = "x" 以下の EGL 値は、VAGen 値に対応します。 <ul style="list-style-type: none"> • F • C 注: これは ISERIESC 用です。	特別な考慮事項なし。
/prep	prep = "YES" "NO"	特別な考慮事項なし。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/prepfiler	buildPlan = "YES" "NO"	特別な考慮事項なし。
/printdest=xxxx xxxx は、以下のいずれかの値です。 • ezeprint • termid	printDestination = "xxxx" 以下の EGL 値は、VAGen 値に対応します。 • PROGRAMCONTROLLED • TERMINALID	特別な考慮事項なし。
/project="xxxx"["version"] xxxx は VisualAge for Java プロジェクトの名前、version は指定したプロジェクトのバージョン名です。	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。これは、このオプションが、ソース・コード・リポジトリに 1 単位としてチェックインする必要がある、関連した EGL プロジェクトのグループの判別に役立つ可能性があるからです。
/projectid=xxxx	projectID = "xxxx"	特別な考慮事項なし。
/recovery	restoreCurrentMsgOnError = "YES" "NO" 注: これは IMS 用です。	特別な考慮事項なし。
/resource=xxxx xxxx は、VAGen リソース関連パーツの名前です。	resourceAssociations = "xxxx" xxxx は、EGL リソース関連パーツの名前です。	特別な考慮事項なし。
/resourcebundlelocale=xxxx	resourceBundleLocale = "xxxx"	マイグレーション・ツールは、/resourcebundlelocale を変換して、その結果をオリジナルのビルド記述子と、 secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツの両方に置きます。
/resvword=xxxx	reservedWord = "xxxx"	特別な考慮事項なし。
/rt=xxxx	returnTransaction = "xxxx"	特別な考慮事項なし。
/runfile	genRunFile = "YES" "NO"	特別な考慮事項なし。
/sendtranslationcmddbcs=xxxx	サポートされていません。 注: EGL は、ホストへのファイル転送用に TCP/IP のみをサポートしています。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/session=xxxx	サポートされていません。 注: EGL は、ホストへのファイル転送用に TCP/IP のみをサポートしています。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/setfull	setFormItemFull = "YES" "NO"	特別な考慮事項なし。
/sp	checkToTransaction = "YES" "NO"	特別な考慮事項なし。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/spa=xxxx,ADF,yyyy 注: ADF はオプションです。 yyyy はオプションであるため、 VisualAge Generator では以下の オプションの組み合わせが有効で す。 /spa=xxxx /spa=xxxx,ADF,yyyy /spa=xxxx,,yyyy	EGL では、次の 3 つの別個のオプションがあります。 spaSize = "xxxx" spaADF = "YES" "NO" spaStatusBytePosition = "yyyy"	マイグレーション・ツールは、 /spa オプションを 3 つの EGL オプション に分割します。 マイグレーション・ツールは、値が YES の場合にのみ spaADF を組み込 みます。
/spzero 注: このオプションは、COBOL 生成でのみサポートされます。	spacesZero = "YES" "NO" 注: このオプションは、COBOL 生成、 Java 生成、およびデバッグでサポートさ れます。	特別な考慮事項なし。
/sqldb=xxxx	sqlDB = "xxxx"	特別な考慮事項なし。
/sqlid=xxxx	sqlID = "xxxx"	マイグレーション・ツールは、VAGen の /dbuser オプションと /sqlID オプ ションを、EGL の sqlID オプション にマージします。生成オプション・パ ーツに /dbuser と /sqlID の両方が含 まれている場合、マイグレーション・ ツールは sqlID を 2 回組み込みま す。EGL 検証により、「問題」ビュ ーにエラー・メッセージが表示されま す。
/sqlpassword=xxxx	sqlPassword = "xxxx"	マイグレーション・ツールは、VAGen の /dbpassword オプションと /sqlpassword オプションを、EGL の sqlPassword オプションにマージしま す。生成オプション・パーツに /dbpassword と /sqlpassword の両方が 含まれている場合、マイグレーショ ン・ツールは sqlPassword を 2 回組 み込みます。EGL 検証により、「問 題」ビューにエラー・メッセージが表 示されます。
/sqlvalid	validateSQLStatements = "YES" "NO"	特別な考慮事項なし。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/symparm=pppppppp,'vvvv' <ul style="list-style-type: none"> • pppppppp は、シンボリック・パラメーターの名前です。 pppppppp は 1 文字から 8 文字までです。 • vvvv は値です。値の中で、2 つの連続した単一引用符は、1 つの単一引用符を表します。 	EGL は、VisualAge Generator と同じ事前定義シンボリック・パラメーターを多数サポートします。新しい EGL のシンボリック・パラメーターと競合しない限り、任意のユーザー定義シンボリック・パラメーターを使用することもできます。	マイグレーション・ツールは、シンボリック・パラメーターを以下のように処理します。 <ul style="list-style-type: none"> • マイグレーション・ツールは、VAGen 定義のシンボリック・パラメーターを、対応する EGL のシンボリック・パラメーターに変換します。 • 対応する EGL のシンボリック・パラメーターが存在しない場合、マイグレーション・ツールは VAGen 定義のシンボリック・パラメーターを EGL のシンボリック・パラメーターの構文に変換し、パラメーターの名前や値は変更しません。また、マイグレーション・ツールはエラー・メッセージを出します。 • マイグレーション・ツールは、ユーザー定義のシンボリック・パラメーターを EGL のシンボリック・パラメーターの構文に変換し、パラメーターの名前や値は変更しません。
/SYMPARM=EZALTXTR,'xxxx'	transferErrorTransaction = "xxxx"	特別な考慮事項なし。
/SYMPARM=EZONEAS2,'xxxx'	oneFormItemCopybook = "YES"	特別な考慮事項なし。
/syncdxfr	synchOnPgmTransfer = "YES" "NO" 注: これは CICS 環境の DL/I 用です。	特別な考慮事項なし。
/syncxfer	synchOnTrxTransfer = "YES" "NO"	特別な考慮事項なし。
/syscodes	sysCodes = "YES" "NO"	特別な考慮事項なし。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>/system=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • MVSBATCH • MVSCICS • IMSBMP • IMSVS • AIX • JAVALINUX • JAVAOS390 • JAVAOS400 • JAVAWINNT • JAVAWRAPPER • WINNT • LINUX • OS400 • HP • SOLARIS • VSEBATCH • VSECICS <p>VAGen では次に示す環境も指定できますが、これらの環境はマイグレーション・ツールによって変換されません。JAVA、JAVAGUI、WINGUI、OS2GUI、OS2、OS2CICS、AIXCICS、NTCICS、SOLACICS、TSO、VMCMS、VMBATCH</p>	<p>system = "xxxx"</p> <p>以下の EGL 値は、VAGen 値に対応しています。</p> <ul style="list-style-type: none"> • ZOSBATCH • CICS for z/OS • IMSBMP • IMSVS • AIX • LINUX • USS • ISERIESJ • WIN • WIN • WIN • LINUX • ISERIESC • HPUX • SOLARIS • VSEBATCH • VSECICS 	<p>マイグレーション・ツールは、このオプションを以下のように処理します。</p> <ul style="list-style-type: none"> • /system=xxxx に対応する値が EGL に存在する場合、マイグレーション・ツールは対応する EGL の値にマイグレーションします。 • /system=xxxx に対応する値が EGL に存在しなければ、マイグレーション・ツールは /system=xxxx をコメントとして組み込みます。 • /system=JAVAWRAPPER の場合、マイグレーション・ツールは以下の EGL のビルド記述子オプションも組み込みます。 <ul style="list-style-type: none"> – enableJavaWrapperGen = "ONLY". これは、プログラムのために Java ラッパーのみを生成するように指定します。 – wrapperCompatibility = "V4". このオプションは、Java ラッパーに VisualAge Generator との互換性が必要であることを指定します。 • COBOL 環境の場合、マイグレーション・ツールは、destPort ビルド記述子オプションの指定が必要であることを示す警告メッセージを出します。
<p>/targnls=xxx</p> <p>xxx は、3 文字の各国語コードです。</p>	<p>targetNLS="xxx"</p> <p>xxx は、3 文字の各国語コードです。ENP (大文字英語) を除くすべての値は、VisualAge Generator と EGL で同一です。ENP に相当するものは EGL には存在しません。</p>	<p>マイグレーション・ツールは、/targnls を変換して、その結果をオリジナルのビルド記述子と、secondaryTargetBuildDescriptor オプションによって参照される新規のビルド記述子パーツの両方に置きます。</p> <p>マイグレーション・ツールは、VAGen 値を targetNLS 値として使用します。値が ENP である場合は、EGL 検査によって「問題」ビューにエラー・メッセージが示されます。.eglbld ファイルを編集して、値を変更できます。ENP の置換値として、ENU (大/小文字混合英語) を使用できます。</p>

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>/templates=xxxx</p> <p>VisualAge Generator の場合は、テンプレートを使用して準備 JCL とランタイム JCL が生成され、また CICS トランザクションとプログラムのエントリが生成されます。</p>	<p>templateDir = "xxxx"</p> <p>EGL の場合、z/OS および iSeries 用の準備テンプレートの代わりにビルド・スクリプトが使用されます。テンプレートが使用されるのは、ZOSBATCH、IMSBMP、および VSEBATCH のランタイム環境用のランタイム JCL、ISERIESC ターゲット環境用のランタイム CL、および VSE ランタイム環境用の準備 JCL を生成する場合のみです。</p>	<p>特別な考慮事項なし。</p>
<p>/trace=xxxx,yyyy</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • none • sqlerr • sqlio <p>yyyy はオプションです。yyyy が存在する場合は、<i>stmt</i> に設定されます。</p> <p>none、sqlerr、または sqlio (、<i>stmt</i> の指定あり、または指定なし) の任意の組み合わせが有効です。</p>	<p>/trace は、次のように複数のビルド記述子オプションに分割されます。</p> <ul style="list-style-type: none"> • sqlerr が組み込まれている場合、sqlErrorTrace = "YES" • sqlio が組み込まれている場合、sqlIOTrace = "YES" • stmt が組み込まれている場合、statementTrace = "YES" 	<p>特別な考慮事項なし。</p>
<p>/transfertype=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • tcpip • sna 	<p>サポートされていません。</p> <p>注: EGL は、ホストへのファイル転送用に TCP/IP のみをサポートしています。</p>	<p>マイグレーション・ツールは、このオプションをコメントとして組み込みます。</p>
<p>/transid=primaryID,restartID</p> <p>VisualAge Generator の場合は、/transid=、restartID が有効で、1 次トランザクションのデフォルトはプログラム名の先頭 4 文字です。</p>	<p>/transid は、次のように複数のビルド記述子オプションに分割されます。</p> <ul style="list-style-type: none"> • primaryID が組み込まれている場合、startTransactionID = "primaryID" • restartID が組み込まれている場合、restartTransactionID = "restartID" 	<p>特別な考慮事項なし。</p>
<p>/twaoff=nnnn</p>	<p>twaOffset = "nnnn"</p>	<p>特別な考慮事項なし。</p>

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/unload VisualAge Generator の場合、/unload を指定すると、現行生成プロセスのために要求されているプロジェクトまたは生成マップのロード前に、VAGen パーツを含むすべての VisualAge Java プロジェクト、または VisualAge Smalltalk 構成マップが、バッチ生成機能によってアンロードされていました。	サポートされていません。	マイグレーション・ツールは、このオプションに関するコメントを組み込みません。
/validmix	validateMixedItems = "YES" "NO"	特別な考慮事項なし。
/vmloadlib=xxxx	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
/vselib=xxxx	vseLibrary = "xxxx"	特別な考慮事項なし。
/workdb=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • aux • main • dli • sql 	workDBType = "xxxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> • AUX • MAIN • DLI • SQL 	特別な考慮事項なし。
使用されません。	vagCompatibility = "YES"	「互換モードを設定しない (Do not set compatibility mode)」マイグレーション設定に基づいて、マイグレーション・ツールはそれぞれのビルド記述子パーツについて、このオプションの追加あるいは省略を行います。
使用されません。	truncateExtraDecimals = "YES"	マイグレーション・ツールは、このオプションを常にすべてのビルド記述子パーツに追加します。この手法は、VAGen の振る舞いを保持するために、デフォルトの YES を使用する必要があることを明示的に示すために使用されます。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>COBOL 生成では、10 進記号および数字分離記号は、ランタイム時に使用される言語依存オプション・モジュールに基づいて設定されます。</p> <p>Java 生成では、decimalSymbol はランタイム・プロパティーです。数字分離記号は、反対になります (例えば、decimalSymbol がピリオドの場合、数字分離記号はコンマです)。</p>	<p>decimalSymbol = "x" separatorSymbol = "y"</p> <p><i>x</i> および <i>y</i> は、以下の記号のいずれかです。</p> <ul style="list-style-type: none"> • ピリオド (.) • コンマ (,) <p><i>x</i> と <i>y</i> を同じ値にすることはできません。</p> <p>COBOL 生成では、この情報を生成時に指定した場合、生成情報が優先されます。生成時に指定しない場合、記号は VisualAge Generator と同じように設定されます。</p> <p>Java 生成では、この情報を生成時に指定した場合、ランタイム・プロパティーの設定に使用されます。生成時に指定しない場合、ランタイム時に使用されるプロパティー・ファイルで設定できます。</p>	<p>マイグレーション・ツールは、decimalSymbol または separatorSymbol を設定しません。これらのオプションをビルド記述子パーツに追加できます。または、以下の技法を使用します。</p> <ul style="list-style-type: none"> • COBOL 生成では、VisualAge Generator に類似した言語依存オプション・モジュールを継続して使用できます。 • Java 生成では、ランタイム時に使用するプロパティー・ファイルで情報を直接設定できます。
使用されません。	<p>destPort = "xxxx"</p> <p>EGL の場合、destPort は、生成出力を実行準備のためにホスト・システムに転送するときに使用するポートを指定します。destPort ビルド記述子オプションは、COBOL 生成のターゲット環境に必要です。</p>	<p>マイグレーション・ツールは、destPort を設定しません。デフォルト値は、以下に示すようにターゲット環境によって異なります。</p> <ul style="list-style-type: none"> • z/OS 環境の場合、destPort のデフォルト値はありません。destPort ビルド記述子オプションを追加する必要があります、その値が z/OS ビルド・サーバーを開始する JCL で使用する値と一致しなければなりません。z/OS ビルド・サーバーを開始するサンプル JCL (ジョブ制御言語) はポート 5555 を使用します。 • iSeries 環境の場合、destPort のデフォルト値はありません。destPort ビルド記述子オプションを追加する必要があります。値は、iSeries ビルド・サーバーによって使用される値と一致しなければなりません。 • VSE 環境の場合、destPort のデフォルト値は 21 です。destPort ビルド記述子オプションを指定する必要があるのは、値が 21 と異なる場合のみです。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
使用されません。	genProject = "xxxx"	Java を生成する場合は、 genProject ビルド記述子オプションを genDirectory オプションに追加して、あるいはその代わりとして指定することが必要になる場合があります。 genProject は、次の場合に必要です。 <ul style="list-style-type: none"> • HP-UX または SOLARIS 向けに生成する場合、あるいは • VGWebTransactions または VGUI レコードを生成する場合
サポートされていません。	tempDirectory = "xxxx"	VGUI レコードを生成する場合、 genProject ディレクトリーに同名の JSP が既に存在するときには、 tempDirectory オプションを使用することにより、生成された JSP を置くディレクトリーを指定することができます。 tempDirectory を指定しない場合は、JSP は genProject ディレクトリーに生成されますが、その名前は newxxxx.JSP になります。ここで、xxxx は VGUI レコードの名前です。
サポートされていません。 VisualAge Generator では、プログラムが EZESYS の値を検査する場合であっても、プログラムを生成する対象であるすべてのターゲット環境に対して、すべての VAGen ソース・コードが有効でなければなりません。	eliminateSystemDependentCode = "YES" "NO" EGL では、プログラムが sysVar.systemType の値を検査する場合は、現行ターゲット生成環境では実行される可能性がないソース・コードを省略できます。これにより、得られる COBOL または Java のソース・コードを小さくすることができます。	マイグレーション・ツールは、 eliminateSystemDependentCode を設定しません。デフォルト値は "YES" です。
使用されません。	sessionBeanID = "xxxx"	マイグレーション・ツールは、 sessionBeanID を設定しません。Java または Java ラッパーを生成する場合、 sessionBeanID ビルド記述子オプションを設定する必要があるかどうか判別するには、EGL のオンライン・ヘルプを参照してください。

表 145. 生成オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
VisualAge Generator の場合は、SQL JDBC ドライバー・クラス、JNDI 名、および接続 URL 情報を、実行時に使用されるプロパティー・ファイルに組み込みます。	sqlJDBCDriverClass = "xxxx" sqlValidationConnectionURL = "xx" EGL の場合、この情報は生成時または実行時に指定できます。	マイグレーション・ツールは、 sqlJDBCDriverClass または sqlValidationConnectionURL ビルド記述子オプションを設定しません。これらの値を生成時に指定する必要がある場合は、以下の技法のいずれかを使用します。 <ul style="list-style-type: none"> ワークスペース設定を指定する。この技法は、Eclipse 環境で生成を行う場合のみ使用できます。 ビルド記述子パーツにビルド記述子オプションを指定する。この技法は、Eclipse 環境で生成を行う場合にも、バッチ生成を行う場合にも使用できます。 どちらの場合も、プロパティー・ファイルが生成されるように、 genProperties ビルド記述子オプションを "GLOBAL" または "PROGRAM" に設定する必要があります。 値を実行時に指定する必要がある場合は、プロパティー・ファイルのランタイム・プロパティーを変更できます。

生成オプション・パーツで使用される変換テーブル名

表 146. 生成オプション: 変換テーブル名

言語	変換テーブルの VAGen <i>/contable</i> 値	EBCDIC 文字セットの EGL <i>serverCodeSet</i>	ASCII 文字セットの EGL <i>clientCodeSet</i>
アラビア語	ELACNARA	IBM-420	IBM-1256
中国語 (簡体字)	ELACNCHS	IBM-935	IBM-1381
中国語 (簡体字)	ELACNGBK	IBM-1388	IBM-1386
中国語 (繁体字)	ELACNCHT	IBM-937	IBM-950
デンマーク語	ELACNDKN	IBM-277	IBM-1252
東ヨーロッパ言語	ELACN870	IBM-870	IBM-1250
英語 (英国)	ELACN285	IBM-285	IBM-1252
英語 (米国)	ELACNENU	IBM-037	IBM-1252
フィンランド語	ELACNFIN	IBM-298	IBM-1252
フランス語	ELACNFRA	IBM-297	IBM-1252
ドイツ語	ELACNDEU	IBM-273	IBM-1252
ギリシャ語	ELACNGRE	IBM-875	IBM-1253
ヘブライ語	ELACNHEB	IBM-424	IBM-1255

表 146. 生成オプション: 変換テーブル名 (続き)

言語	変換テーブルの VAGen /contable 値	EBCDIC 文字セットの EGL serverCodeSet	ASCII 文字セットの EGL clientCodeSet
イタリア語	ELACNITA	IBM-280	IBM-1252
日本語、カタカナ	ELACNJPN	IBM-930	IBM-943
日本語、ローマ字	ELACNJPL	IBM-939	IBM-943
韓国語	ELACNKOR	IBM-933	IBM-949
ノルウェー語	ELACNDKN	IBM-277	IBM-1252
ポルトガル語	ELACNPTB	IBM-037	IBM-1252
ロシア語	ELACNCYR	IBM-1025	IBM-1251
スペイン語	ELACNESP	IBM-284	IBM-1252
スウェーデン語	ELACNSWE	IBM-278	IBM-1252
スイス・ドイツ語	ELACNDES	IBM-500	IBM-1252
トルコ語	ELACNTUR	IBM-1026	IBM-1254
ユーザー定義 (この テーブルの以前の行 ではリストされてい ません)	XXXXXXXX	XXXXXXXX	XXXXXXXX

リンケージ・テーブルおよびリソース関連パーツで使用されている 変換テーブル名

マイグレーション・ツールは、以下の方法で VAGen /contable オプションを EGL **conversionTable** 属性に変換します。

- 変換テーブル名の先頭の 4 文字が "CSOX" (AIX サーバー) または "CSOI" (OS/2 または Windows サーバー) である場合、マイグレーション・ツールは先頭の 4 文字を "CSOJ" に変換し、名前の残りの部分は変更しません。OS/2 サーバーのために変換された変換テーブル名が、無効な EGL 変換テーブル名、または有効ではあっても不適切な変換テーブル名になる場合があります。
- 変換テーブル名の先頭の 4 文字が "CSOE" (MVS または OS/400 サーバー) である場合、マイグレーション・ツールは変換テーブル名を変更しません。
- 変換テーブル名の先頭の 5 文字が "ELACN" または "ELAAX" である場合、マイグレーション・ツールはテーブル名を 443 ページの表 147 に示すように変更します。
- 名前がテーブルに見つからない、または他の命名規則に従っている場合、マイグレーション・ツールはテーブル名を「現状どおり」変換します。

表 147. リンケージ・テーブルおよびリソース関連パーツ: 変換テーブルの値

言語	VAGen /contable ローカル: ASCII リモート: EBCDIC	EGL 変換テーブル	VAGen /contable ローカル: Windows リモート: AIX、HP-UX、 または Solaris	EGL 変換テーブル
アラビア語	ELACNARA	CSOE420	ELAAXARA	CSOJ1046
中国語 (簡体字)	ELACNCHS	CSOE935	BINARY	CSOJ1381
中国語 (簡体字)	ELACNGBK	CSOE935	非サポート	CSOJ1386
中国語 (繁体字)	ELACNCHT	CSOE937	BINARY	CSOJ950
デンマーク語	ELACNDKN	CSOE277	ELAAX850	CSOJ850
東ヨーロッパ言語	ELACN870	CSOE870	ELAAX912	CSOJ852
英語 (英国)	ELACN285	CSOE285	ELAAX850	CSOJ850
英語 (米国)	ELACNENU	CSOE037	ELAAX437	CSOJ850
フィンランド語	ELACNFIN	CSOE298	ELAAX850	CSOJ850
フランス語	ELACNFRA	CSOE297	ELAAX850	CSOJ850
ドイツ語	ELACNDEU	CSOE273	ELAAX850	CSOJ850
ギリシャ語	ELACNGRE	CSOE875	ELAAXGRE	CSOJ813
ヘブライ語	ELACNHEB	CSOE424	ELAAXHEB	CSOJ856
イタリア語	ELACNITA	CSOE280	ELAAX850	CSOJ850
日本語、カタカナ	ELACNJPN	CSOE930	BINARY	CSOJ943
日本語、ローマ字	ELACNJPL	CSOE939	BINARY	CSOJ943
韓国語	ELACNKOR	CSOE933	BINARY	CSOJ1363
ノルウェー語	ELACNDKN	CSOE277	ELAAX850	CSOJ850
ポルトガル語	ELACNPTB	CSOE037	ELAAX850	CSOJ850
ロシア語	ELACNCYR	CSOE1025	ELAAXCYR	CSOJ866
スペイン語	ELACNESP	CSOE284	ELAAX850	CSOJ850
スウェーデン語	ELACNSWE	CSOE278	ELAAX850	CSOJ850
スイス・ドイツ語	ELACNDES	CSOE500	ELAAX850	CSOJ850
トルコ語	ELACNTUR	CSOE1026	ELAAXTUR	CSOJ920

リンケージ・テーブル・パーツ

リンケージ・テーブル・パーツは、Calllink、Filelink、Crtxlink、および Dxfrlink です。

callLink

表 148. :callLink のリンケージ・テーブル・オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
:callLink	callLink	特別な考慮事項なし。
linktype=xxx xxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> 動的 static cicslink remote csocall sessionejb 	呼び出しのタイプ。 以下の EGL 値は、VAGen 値に対応します。 <ul style="list-style-type: none"> localCall localCall localCall remoteCall remoteCall ejbCall 	VAGen の linktype が省略されている場合、マイグレーション・ツールは localCall を使用します。マイグレーション・ツールは、EGL の callLink エントリーに関するその他のプロパティを設定するために、他の場所でも linktype を使用します。
applname=programName programName は、呼び出されるプログラムの名前です。ワイルドカードを使用できます。	pgmName = "programName"	特別な考慮事項なし。
externalname=applname	alias = "applname"	名前が EGL の予約語であるために VAGen プログラムの名前を変更する必要がある場合は、プログラム定義またはリンケージ・オプション・パーツのどちらかに alias プロパティを使用して、生成されるプログラムの名前としてプログラムの元の VAGen 名を指定できます。どちらの技法を使用しても、VAGen プログラムによって呼び出される非 VAGen プログラムを変更する必要がなくなります。
package=packageName	package = "packageName"	Java の生成を行う場合に、呼び出し側と呼び出し先のプログラムが別々のパッケージにあるときは、パッケージ名を呼び出し先プログラムのリンケージ・エントリーに組み込むことができます。または、 call ステートメントを変更して、パッケージ名でプログラムを明示的に修飾するか、 call ステートメントを含むファイルにパッケージの import ステートメントを組み込みます。

表 148. :callLink のリンケージ・テーブル・オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
library=libraryName または dllname=libraryName VisualAge Generator の場合、library と dllname は同義語として扱われます。	library = "libraryName"	マイグレーション・ツールは、VAGen のライブラリーまたは dllname を EGL の library プロパティにマージします。
linktype=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> 動的 static cicslink 	linkType = "xxxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> DYNAMIC STATIC CICSLINK 	特別な考慮事項なし。
parmform=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> oslink commptr commdata cicsoslink 	parmForm = "xxxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> OSLINK COMMPTR COMMDATA CICSOSLINK 	特別な考慮事項なし。
contable=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> conversionTableName * EZECONVT BINARY NONE 	conversionTable = "xxxx" 以下の EGL 値は、VAGen 値に対応します。 <ul style="list-style-type: none"> conversionTableName * PROGRAMCONTROLLED 非サポート 非サポート 	<p>マイグレーション・ツールは、442 ページの『リンケージ・テーブルおよびリソース関連パーツで使用されている変換テーブル名』の情報を使用して、EGL callLink エLEMENTの conversionTable プロパティを変換します。</p> <p>マイグレーション・ツールは、VAGen の contable=BINARY を BINARY にマイグレーションしますが、この値は EGL ではサポートされません。また、マイグレーション・ツールはエラー・メッセージを出します。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。 .eglbld ファイルを編集し、サポートされる値を選択して使用することによって、エラーを訂正する必要があります。</p> <p>VAGen contable=NONE の場合、マイグレーション・ツールは conversionTable プロパティを省略します。</p>

表 148. :callLink のリンケージ・テーブル・オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>location=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • systemName • EZELOC 	<p>location = "xxxx"</p> <p>以下の EGL 値は、VAGen 値に対応しています。</p> <ul style="list-style-type: none"> • systemName • PROGRAMCONTROLLED 	<p>特別な考慮事項なし。</p>
<p>remotecomtype=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • appcims • ca400 • cicsclient • dce • dcsecure • direct • exci • ipc • java400 • lu2 • tcpip 	<p>remoteComType = "xxxx"</p> <p>以下の EGL 値は、VAGen 値に対応しています。</p> <ul style="list-style-type: none"> • 非サポート • 非サポート • CICSECI • 非サポート • 非サポート • DIRECT • 非サポート • DISTINCT • JAVA400 • 非サポート • TCPIP 	<p>マイグレーション・ツールは、cicsclient を対応する最も近い EGL 値である CICSECI に変換します。VAGen の :callLink エントリーで ctgPort と ctgLocation がまだ指定されていない場合、マイグレーション・ツールはエラー・メッセージを出してこれらの値の指定を促します。</p> <p>マイグレーション・ツールは、リストに「サポートなし」と示されている値を「現状のまま」マイグレーションし、メッセージを出します。この時点で、使用する通信プロトコルを決定し、正しい情報を使用して EGL の callLink エントリーを更新する必要があります。 callLink エントリーを訂正するまで、EGL 検査によって、「問題」ビューにエラー・メッセージが示されます。</p> <p>CICSSSL を使用する場合は、ctgPort、ctgLocation、ctgKeyStore、および ctgKeyStorePassword の各プロパティを EGL の callLink エントリーに追加する必要があります。</p> <p>CICSJ2C を使用する場合は、pgmName、conversionTable、remotePgmType、luwControl、remoteBind、location、および parmForm の各プロパティを EGL の callLink エントリーに追加する必要があります。</p> <p>マイグレーション・ツールは、APPCIMS を「現状のまま」マイグレーションします。その理由は、APPCIMS がサポートされていないため、およびその他のプロパティの値がまったく異なるためです。APPCIMS に最適な置換値は IMSTCP です。</p>

表 148. :callLink のリンケージ・テーブル・オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
remoteapptype=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • vg • nonvg • vgjava • itf 	remotePgmType = "xxxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> • EGL • EXTERNALLYDEFINED • 適用不可 • 非サポート 	VisualAge Generator remoteapptype=vgjava の場合、マイグレーション・ツールは :callLink エントリをマイグレーションしますが、 remotePgmType プロパティを省略します。 remoteapptype=itf の場合、マイグレーション・ツールは :callLink エントリ全体をコメント化します。
serverid=serverName	serverID="serverName"	特別な考慮事項なし。
luwcontrol=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • client • server 	luwControl = "xxxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> • CLIENT • SERVER 	特別な考慮事項なし。
remotebind=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • 生成 • ランタイム 	remoteBind = "xxxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> • GENERATION • RUNTIME 	特別な考慮事項なし。
providerURL=URLName	providerURL = "URLName"	特別な考慮事項なし。
ctglocation='tcpipInfo'	ctgLocation = "tcpipInfo"	特別な考慮事項なし。
ctgport=portID	ctgPort = "portID"	特別な考慮事項なし。
bitmode=nn nn は、以下のいずれかの値です。 <ul style="list-style-type: none"> • 16 • 32 	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。
binform=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • intel • host 	サポートされていません。	マイグレーション・ツールは、このオプションをコメントとして組み込みます。

表 148. :callLink のリンケージ・テーブル・オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
サポートされていません。 VisualAge Generator では、呼び出し先プログラムが画面にマップを送らない場合に、CALL ステートメントに NOMAPS オプションを指定するとパフォーマンスが向上します。	refreshScreen = "YES" "NO"	マイグレーション・ツールは、このプロパティを設定しません。以前、VAGen の CALL ステートメントに NOMAPS を指定していた場合は、EGL の call ステートメントに isNoRefresh = YES プロパティを引き続き使用できます。代わりに、呼び出し先プログラムに関する callLink エントリーに refreshScreen = "NO" を指定しても、同じサポートを利用できます。
使用されません。 VisualAge Generator によってサポートされる通信プロトコルは、この情報を必要としません。	ctgKeyStore ctgKeyStorePassword	マイグレーション・ツールは、以下のプロパティを設定しません。 remoteComType = "CICSSSL" を使用する場合は、 ctgKeyStore と ctgKeyStorePassword が必要です。
使用されません。 VisualAge Generator では、呼び出し先パッチ・プログラム用の Java ラッパーを生成する場合には、/system=JAVAWRAPPER 生成オプションを使用します。	javaWrapper = "YES" "NO"	マイグレーション・ツールは、このプロパティを設定しません。呼び出し先プログラムを生成するたびに Java ラッパーが生成されるようにする場合は、 javaWrapper = "YES" を指定する必要があります。

fileLink

表 149. :filelink のリンケージ・テーブル・オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
:filelink	fileLink	特別な考慮事項なし。
linktype=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> ローカル remote VisualAge Generator のデフォルトは local です。	ファイルのタイプ。 以下の EGL 値は、VAGen 値に対応します。 <ul style="list-style-type: none"> localFile remoteFile 	VAGen の linktype が指定されていない場合、マイグレーション・ツールは localFile に変換します。
filename=fileName fileName は、VAGen レコード定義にあるファイルの名前です。ワイルドカードを使用できます。	fileName = "fileName"	特別な考慮事項なし。

表 149. :filelink のリンケージ・テーブル・オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>contable=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • conversionTableName • * • EZECONVT • BINARY 	<p>conversionTable="xxxx"</p> <p>以下の EGL 値は、VAGen 値に対応しています。</p> <ul style="list-style-type: none"> • conversionTableName • * • PROGRAMCONTROLLED • 非サポート 	<p>マイグレーション・ツールは、442 ページの『リンケージ・テーブルおよびリソース関連パーツで使用されている変換テーブル名』の情報を使用して、EGL fileLink エレメントの conversionTable プロパティを変換します。</p> <p>マイグレーション・ツールは、VAGen の contable=BINARY を BINARY にマイグレーションしますが、この値は EGL ではサポートされません。また、マイグレーション・ツールはエラー・メッセージを出します。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。 .eglbld ファイルを編集し、サポートされる値を選択して使用することによって、エラーを訂正する必要があります。</p>
<p>location=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • CICS • EZELOC 	<p>locationSpec="xxxx"</p> <p>以下の EGL 値は、VAGen 値に対応しています。</p> <ul style="list-style-type: none"> • CICS • PROGRAMCONTROLLED 	<p>特別な考慮事項なし。</p>

Crtxlink

表 150. :crtxlink のリンケージ・テーブル・オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
:crtxlink	asynchLink	特別な考慮事項なし。
<p>linktype=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • ローカル • remote <p>注: VisualAge Generator のデフォルトは local です。</p>	<p>起動のタイプ。</p> <p>以下の EGL 値は、VAGen 値に対応します。</p> <ul style="list-style-type: none"> • localAsynch • remoteAsynch 	<p>VAGen の linktype が指定されていない場合、マイグレーション・ツールは localAsynch に変換します。</p>
<p>recdname=recordName</p> <p>recordName は、VAGen レコード定義の名前です。ワイルドカードを使用できます。</p>	<p>recordName = "recordName"</p>	<p>特別な考慮事項なし。</p>

表 150. :crtxlink のリンケージ・テーブル・オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
contable=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • conversionTableName • * • EZECONVT • BINARY 	conversionTable = "xxxx" 以下の EGL 値は、VAGen 値に対応します。 <ul style="list-style-type: none"> • conversionTableName • * • PROGRAMCONTROLLED • 非サポート 	マイグレーション・ツールは、442 ページの『リンケージ・テーブルおよびリソース関連パーツで使用されている変換テーブル名』の情報を使用して、EGL asynchLink エレメントの conversionTable プロパティを変換します。 マイグレーション・ツールは、VAGen の contable=BINARY を BINARY に変換しますが、この値は EGL ではサポートされません。また、マイグレーション・ツールはエラー・メッセージを出します。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。 .eglbld ファイルを編集し、サポートされる値を選択して使用することによって、エラーを訂正する必要があります。
location=xxxx xxxx は、以下のいずれかの値です。 <ul style="list-style-type: none"> • CICS • EZELOC 	locationSpec = "xxxx" 以下の EGL 値は、VAGen 値に対応しています。 <ul style="list-style-type: none"> • CICS • PROGRAMCONTROLLED 	特別な考慮事項なし。
package=packageName	package = "packageName"	特別な考慮事項なし。

Dxfrlink

表 151. :dxfrlink のリンケージ・テーブル・オプション

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
:dxfrlink	transferToProgram	特別な考慮事項なし。
fromappl=programName <i>programName</i> は、DXFR を使用して別のプログラムへの転送を行うプログラムの名前です。ワイルドカードは使用できません。	fromPgm = "programName"	特別な考慮事項なし。
toappl=programName2 <i>programName2</i> は、転送先のプログラムの名前です。	toPgm = "programName2"	特別な考慮事項なし。

表 151. :dxfrlink のリンケージ・テーブル・オプション (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>linktype=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> 動的 static noncsp 	<p>linkType = "xxxx"</p> <p>以下の EGL 値は、VAGen 値に対応します。</p> <ul style="list-style-type: none"> DYNAMIC STATIC EXTERNALLYDEFINED 	<p>以前、VAGen の DXFR ステートメントに NONCSP を指定していた場合は、EGL のtransfer to program ステートメントに isExternal = "YES" プロパティを引き続き使用できます。代わりに、転送先のプログラムに関する transferToProgram エlementに linkType = "EXTERNALLYDEFINED" を指定しても、同じサポートを利用できます。</p>
サポートされていません。	<p>alias = "applname"</p>	<p>名前が EGL の予約語であるために VAGen プログラムの名前を変更する必要がある場合は、プログラム定義またはリンケージ・オプション・パーツのどちらかに alias プロパティを使用して、生成されるプログラムの名前としてプログラムの元の VAGen 名を指定できます。どちらの技法を使用しても、VAGen プログラムからの転送先である非 VAGen プログラムを変更する必要がなくなります。</p>

リソース関連パーツ

表 152. リソース関連

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>VisualAge Generator では、リソース関連パーツは、特定のターゲット環境に対してファイルをインプリメントする方法を指定します。このファイルは、VAGen レコード定義に指定されているファイル名です。</p> <p>リソース関連パーツは、特定のターゲット環境に対して印刷出力をインプリメントする方法も指定できます。</p> <p>プログラムの生成時には、それぞれの索引付き出力、シリアル出力、相対出力、または印刷出力ごとに、fileName がリソース関連パーツと突き合わせられます。</p> <p>fileName と生成ターゲット環境を基準として最初にマッチングしたエントリーが、そのファイルに対して使用されるエントリーになります。</p>	<p>EGL のリソース関連パーツは、特定のターゲット環境に対してファイルをインプリメントする方法を指定します。このファイルは、EGL レコード定義に指定されている fileName プロパティです。</p> <p>リソース関連パーツは、特定のターゲット環境に対して印刷出力をインプリメントする方法も指定できます。</p> <p>プログラムの生成時には、それぞれの索引付き出力、シリアル出力、相対出力、または印刷出力ごとに、fileName プロパティがリソース関連パーツと突き合わせられます。fileName と生成ターゲット環境について最初にマッチングしたエントリーが、そのファイルに対して使用されるエントリーになります。</p>	<p>特別な考慮事項なし。</p>
<p>VisualAge Generator の場合、C++ を生成すると、リソース関連ファイルは実行時にも使用されます。</p>	<p>EGL の場合、リソース関連情報は EGL パーツに格納されます。</p>	<p>マイグレーション・ツールは、VAGen リソース関連ファイル内でのみ有効だった、その他のオプションの変換をサポートしています。</p>
<pre>file = fileName EZEPRINT</pre>	<pre>fileName="fileName" "printer"</pre>	<p>特別な考慮事項なし。</p>

表 152. リソース関連 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p><code>/system=targetSystem</code></p> <p><code>targetSystem</code> は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • AIX * • AIXCICS * • HP-UX * • IMSBMP • IMSVS • LINUX ** • MVS BATCH • MVSCICS • NTCICS * • OS2 * • OS2CICS • OS400 • SCO * • SOLACICS * • SOLARIS * • TSO • VMCMS • VMBATCH • VSEBATCH • VSECICS • WINNT ** <p>注:</p> <ul style="list-style-type: none"> • * - C++ 生成の場合に使用される環境を示します。 • ** - Java 生成の場合に使用される環境を示します。 • <code>/system</code> はオプションです。 • VisualAge Generator は、ターゲット・システムのワイルドカードとして * をサポートします。(例: MVS* または *CICS) 	<p>これは EGL ターゲット環境です。</p> <p>以下の EGL 値は、VAGen 値に対応しています。</p> <ul style="list-style-type: none"> • aix • 非サポート • hpux • imsbmp • imsvs • linux • zosbatch • zoscics • 非サポート • 非サポート • 非サポート • iseriesc • 非サポート • 非サポート • 非サポート • solaris • 非サポート • 非サポート • vsebatch • vsecics • win <p>注: ワイルドカードはサポートされません。</p>	<p>マイグレーション・ツールは、<code>/system</code> オプションを以下のように処理します。</p> <ul style="list-style-type: none"> • リストに「サポートなし」と示されているターゲット・システムの場合、マイグレーション・ツールは、VAGen リソース関連エントリーの情報をコメントとして EGL リソース関連パーツに組み込みます。これにより、情報が可能なかぎり保持されます。 • <code>/system</code> オプションが VAGen リソース関連エントリーから省略された場合、マイグレーション・ツールは any を EGL リソース関連ターゲット環境として使用します。 • <code>/system</code> オプションにワイルドカードが使用されている場合、マイグレーション・ツールはワイルドカードを含めてオプションをそのままマイグレーションします (例: <code>mvs*</code> または <code>*cics</code>)。また、マイグレーション・ツールはエラー・メッセージを出します。

表 152. リソース関連 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>/filetype=<i>fileType</i></p> <p><i>fileType</i> は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • BTRIEVE • GSAM • IBMCOBOL • MFCOBOL • MMSGQ • MQ • OS2COBOL • SEQ • SEQRS • SMSGQ • SPOOL • TEMPAUX • TEMPMAIN • TRANSIENT • VSAM • VSAMRS 	<p>EGL のファイル・タイプ。</p> <p>以下の EGL 値は、VAGen 値に対応します。</p> <ul style="list-style-type: none"> • 非サポート • gsam • ibmcobol • 非サポート • mmsgq • mq • 非サポート • seq または seqws • seqrs • smsgq • spool • tempaux • tempmain • transient • vsam • vsamrs 	<p>マイグレーション・ツールは、/filetype オプションを以下のように処理します。</p> <ul style="list-style-type: none"> • /filetype オプションが VAGen リソース関連エントリから省略された場合、マイグレーション・ツールは default を EGL ファイル・タイプとして使用します。 • /system オプションがホスト・ターゲット環境を指定している場合、マイグレーション・ツールは VAGen の SEQ ファイル・タイプを EGL の seq ファイル・タイプに変換します。 • /system オプションがワークステーション環境である場合、マイグレーション・ツールは VAGen の SEQ ファイル・タイプを EGL の seqws ファイル・タイプに変換します。 • サポートされないファイル・タイプ値の場合、リソース関連の対象の /system がサポートされていると、マイグレーション・ツールは VAGen ファイル・タイプを使用して EGL リソース関連エントリを作成し、エラー・メッセージを出します。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。EGL リソース関連パーツを使用するには、その前にこのエラーを修正する必要があります。
/sysname= <i>systemName</i>	systemName = " <i>systemName</i> "	マイグレーション・ツールは、/sysname オプションの中で使用されているシンボリック・パラメーターを、対応する EGL の置換シンボリック・パラメーターに変換します。
/replace /noreplace	replace = "YES" replace = "NO"	特別な考慮事項なし。
/dup /nodup	duplicates = "YES" duplicates = "NO" 注: これは ISERIESC 用です。	特別な考慮事項なし。

表 152. リソース関連 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
/commit /nocommit これらのオプションは、OS/400 ターゲット環境の場合のみ使用されます。	commit = "YES" commit = "NO" 注: これは ISERIESC 用です。	特別な考慮事項なし。
/blksize=xxxx,yyyy,zzzz VisualAge Generator では、このオプションは VSE ターゲット環境の場合のみ使用されます。	blockSize = "xxxx,yyyy,zzzz"	特別な考慮事項なし。
/sysnum=xxxx VisualAge Generator では、このオプションは VSE ターゲット環境の場合のみ使用されます。	systemNumber = "xxxx"	特別な考慮事項なし。
/label /nolabel VisualAge Generator では、このオプションは VSE ターゲット環境の場合のみ使用されます。	standardLabel = "YES" standardLabel = "NO"	特別な考慮事項なし。
/pcbno= <i>n</i> これは、IMSVS または IMSBMP のターゲット環境、またはファイル・タイプが GSAM ならば MVS バッチの場合のみ有効です。	pcbName = "pcb <i>n</i> "	マイグレーション・ツールは、リテラルの "pcb" と PCB 番号を連結することで、PCB 番号を名前に変換します。
/noff VisualAge Generator に /FF オプションはありません。このオプションは、VAGen リソース関連ファイル内でのみサポートされます。	FormFeedOnClose = "NO" "YES"	マイグレーション・ツールは /noff を FormFeedOnClose = "NO" に変換します。
/text VisualAge Generator に /NOTEXT オプションはありません。このオプションは、VAGen リソース関連ファイル内でのみサポートされます。	text = "YES" "NO"	マイグレーション・ツールは /text を text = "YES" に変換します。

表 152. リソース関連 (続き)

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
<p>/contable=xxxx</p> <p>xxxx は、以下のいずれかの値です。</p> <ul style="list-style-type: none"> • conversionTableName • EZECONVT <p>このオプションは、VAGen リソース関連ファイル内でのみサポートされます。</p>	<p>conversionTable = "xxxx"</p> <p>以下の EGL 値は、VAGen 値に対応します。</p> <ul style="list-style-type: none"> • conversionTableName • PROGRAMCONTROLLED 	<p>マイグレーション・ツールは、442 ページの『リンケージ・テーブルおよびリソース関連パーツで使用されている変換テーブル名』の情報を使用して、EGL のリソース関連エレメントの conversionTable プロパティを変換します。</p>
<p>/keys=xxxx</p> <p>VisualAge Generator では、このオプションは /filetype=BTRIEVE を指定した場合のみ使用できます。このオプションは、VAGen リソース関連ファイル内でのみサポートされます。</p>	<p>keys = "xxxx"</p>	<p>BTRIEVE はサポートされるターゲット環境で使用されるので、マイグレーション・ツールは /keys オプションを EGL の keys プロパティにマイグレーションします。</p>
<p>/basename=xxxx</p> <p>VisualAge Generator では、このオプションは OS/2 ターゲット環境の場合のみ使用されます。このオプションは、VAGen リソース関連ファイル内でのみサポートされます。</p>	<p>サポートされていません。</p>	<p>マイグレーション・ツールは、OS/2 ターゲット環境に関するエントリーをすべてコメント化します。</p>

リンク・エディット・パーツ

表 153. リンク・エディット・パーツ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
VisualAge Generator では、リンク・エディット・パーツの標準の名前は <i>programName.suffix</i> です。ここで、名前の先頭部分は VAGen プログラム名と同じで、接尾部は LKG です。VAGen の /linkedit 生成オプションが、接尾部の値を指定します。	デフォルトでは、EGL のリンク・エディット・パーツ名はプログラム名と同じであることが必要です。この場合、 linkEdit ビルド記述子オプションを指定する必要はありません。 複数のランタイム環境がある場合、環境ごとに異なるリンク・エディット・パーツが必要な場合があります。この場合、リンク・エディット・パーツ名の 1 つを除いたすべてがプログラム名と異なっている必要があります。異なるリンク・エディット・パーツ名を参照するには、完全なリンク・エディット・パーツ名を linkEdit ビルド記述子オプションで指定します。	接尾部が .LKG ならば、マイグレーション・ツールは新規の EGL リンク・エディット・パーツを作成する際に接尾部を除去します。接尾部が .LKG 以外のものならば、ピリオド (.) は EGL パーツ名の中では無効な文字なので、マイグレーション・ツールは <i>.suffix</i> を <i>_suffix</i> に変更します。
VAGen のリンク・エディット・パーツは、ホスト環境での準備プロセス中にプログラムをリンク・エディットするために必要なリンク・エディット・ステートメントを含んでいます。	EGL のリンク・エディット・パーツは、ホスト環境でのビルド・プロセス中にプログラムをリンク・エディットするために必要なリンク・エディット・ステートメントを含んでいます。	マイグレーション・ツールは、以下の処理を行います。 <ul style="list-style-type: none"> リンク・エディット・パーツの中で使用されているシンボリック・パラメーターを、対応する EGL の置換シンボリック・パラメーターに変換します。 VAGen パーツ内と同じ字下げを使用します。

バインド制御パーツ

表 154. バインド制御パーツ

VisualAge Generator 4.5	マイグレーション・ツールによって生成される EGL	マイグレーション・ツールに関する考慮事項
VisualAge Generator では、バインド制御パーツの標準の名前は <i>programName.suffix</i> です。ここで、名前の先頭部分は VAGen プログラム名と同じで、接尾部は BND です。VAGen の /bind 生成オプションが、接尾部の値を指定します。	デフォルトでは、EGL のバインド制御パーツ名はプログラム名と同じであることが必要です。この場合、 bind ビルド記述子オプションを指定する必要はありません。 複数のランタイム環境がある場合、環境ごとに異なるバインド制御パーツが必要な場合があります。この場合、バインド制御パーツ名の 1 つを除いたすべてがプログラム名と異なっている必要があります。異なるバインド制御パーツ名を参照するには、完全なバインド制御パーツ名を bind ビルド記述子オプションで指定します。	接尾部が .BND ならば、マイグレーション・ツールは新規の EGL バインド制御パーツを作成する際に接尾部を除去します。接尾部が .BND 以外のものならば、ピリオド (.) は EGL パーツ名の中では無効な文字なので、マイグレーション・ツールは <i>.suffix</i> を <i>_suffix</i> に変更します。
VAGen のバインド制御パーツは、MVS ホスト環境での準備プロセス中に行われる、プログラムへの DB2 データベース・リソース・モジュール (DBRM) のバインドに必要な DB2 バインド・コマンドを格納しています。	EGL のバインド制御パーツは、z/OS ホスト環境でのビルド・プロセス中に DBRM をプログラムにバインドするために必要なバインド・コマンドを含んでいます。	マイグレーション・ツールは、以下の処理を行います。 <ul style="list-style-type: none"> バインド制御パーツの先頭にコマンドを追加します。これらのコマンドは、ビルド・サーバーが必要とするものです。 バインド制御パーツの中で使用されているシンボリック・パラメーターを、対応する EGL の置換シンボリック・パラメーターに変換します。 VAGen パーツ内と同じ字下げを使用します。

シンボリック・パラメーター

次の表に、VAGen のシンボリック・パラメーターと EGL のシンボリック・パラメーターの関係を示します。

表 155. パーツ関連のシンボリック・パラメーター

パーツ関連のシンボリック・パラメーター	対応する EGL のシンボリック・パラメーター
EZECOBOLTYPE	非サポート
EZEDATA	ビルド計画での DATA または EZEDATA
EZEDBCS	非サポート
EZEDESTLIB	EZEDESTLIBRARY
EZEDESTNAME	非サポート
EZEDLI	EZEDLI - DL/I のみ
EZEENTRY	非サポート

表 155. パーツ関連のシンボリック・パラメーター (続き)

パーツ関連のシンボリック・パラメーター	対応する EGL のシンボリック・パラメーター
EZEENV	ビルド計画での SYSTEM または EZEENV
EZEGDATE	EZEGDATE
EZEGENOUT	非サポート
EZEGMBR	EZEGMBR
EZEGTIME	EZEGTIME
EZEJOB	非サポート
EZEMBR	JCL (ジョブ制御言語) スクリプトでは、リンク・エディット、またはバインド・パーツ、EZEALIAS。それ以外の場合は、EZEMBR。
EZEMBRPATH	非サポート
EZEMSG	非サポート
EZENLS	EZENLS
EZEPID	EZEPID
EZEPREPDESTACCOUNT	非サポート
EZEPREPDESTHOST	非サポート
EZEPREPDESTDIR	非サポート
EZEPREPDESTPASSWORD	非サポート
EZEPREPDESTUID	EZEDESTUSERID
EZEPREPFTPCMDBSCS	非サポート
EZEPREPFTPCMDBCS	非サポート
EZEPRESENDICMDDBCS	非サポート
EZEPREPSESSION	非サポート
EZEPREPSP	非サポート
EZEPREPSQLDB	非サポート
EZEPREPWORKDB	非サポート
EZEPSB	サポートなし - DL/I および IMS のみ
EZEPTYPE	非サポート
EZESQL	EZESQL
EZETBLNAME	非サポート
EZETPROC	非サポート
EZETRAN	EZETRAN
EZETRANSFERTYPE	非サポート
EZETRO	非サポート
EZETWASIZE	非サポート
EZEUSERID	非サポート
EZEVMLoadLIB	適用不可 - VM のみ
EZEVSELIB	ESEVSELIB — VSE のみ
EZEXAPP	サポートされていません。

表 156. ファイル関連のシンボリック・パラメーター

ファイル関連のシンボリック・パラメーター	対応する EGL のシンボリック・パラメーター
EZEBLK	EZEBLK
EZEDBD	非サポート
EZEDD	EZEDD
EZEDLBL	EZEDLBL — VSE のみ
EZEDSN	EZEDSN
EZELRECL	EZELRECL
EZERECFM	EZERECFM

表 157. ユーザー定義のシンボリック・パラメーター

ユーザー定義のシンボリック・パラメーター	対応する EGL のシンボリック・パラメーター
COB2LIB	COBCICS
COBLIST	非サポート
DBDLIB	DBDLIB - DL/I のみ
DSNLOAD	DSNLOAD
DSYS	DSYS
ELA	ELA
EZALTXTR	標準ビルド記述子オプションへの特殊なマイグレーション。『生成オプション』セクションの transferErrorTransaction = "xxx" を参照
EZONEAS2	標準ビルド記述子オプションへの特殊なマイグレーション。『生成オプション』セクションの oneFormItemCopybook = "YES" を参照
EZUAUTH	EZUAUTH
EZUINST	EZUINST
PSBLIB	PSBLIB - DL/I のみ
PROCLIB	PROCLIB — VSE のみ
PWRCLASS	PWRCLASS — VSE のみ
RESLIB	RESLIB - DL/I および IMS のみ
SQLDBNAM	SQLDBNAM — VSE のみ
SQLPKGNM	SQLPKGNM — VSE のみ
SQLPROPT	SQLPROPT — VSE のみ
SQLSTMDE	SQLSTMDE — VSE のみ
SQLSTOPT	SQLSTOPT — VSE のみ
SQLUSRPW	SQLUSRPW — VSE のみ
VMFMODE	適用不可 - VM のみ
VMDISKADDR	適用不可 - VM のみ
VUSERLIB	VUSERLIB — VSE のみ

他の生成情報

このセクションは、以下の表で構成されています。

- MVS の準備テンプレートおよびプロシージャー、461 ページの表 158
- MVS のランタイム・テンプレート、463 ページの表 159
- MVS のファイルおよびデータベース割り振りテンプレート、464 ページの表 160
- MVS のファイルおよびデータベース割り振りプレースホルダー・テンプレート、465 ページの表 161
- OS/400 ランタイム・テンプレート、465 ページの表 162

注: VSE 情報については、「*Rational Business Developer V7.5 Generation for z/VSE feature Reference Manual*」(SC19-2539-00) を参照してください。

準備テンプレートおよびプロシージャー

表 158は、MVS ランタイム環境の COBOL 生成の出力の準備で使用する VAGen 準備テンプレートおよびプロシージャーを示しています。表には、対応する EGL z/OS ランタイム環境がある MVS ランタイム環境のみが含まれています。最初にテンプレート名、その後にプロシージャー名が示されます。ほとんどのプロシージャー名では、先頭の 3 文字は ELA で、残りの文字は、次のようにプロシージャーに含まれる手順を示します。P (DB2 プリコンパイル)、T (CICS 変換)、C (COBOL コンパイル)、L (リンク・エディット)、および B (DB2 バインド)。

注:

- MVS ランタイム環境では、VisualAge Generator はバインド制御テンプレートも使用します。バインド制御テンプレートの変換方法について詳しくは、254 ページの『テンプレートとして使用するバインド制御パーツの設定』を参照してください。
- VSE 用の生成フィーチャーは、VisualAge Generator と同様に、引き続き準備テンプレートおよび準備プロシージャーを使用します。ただし、テンプレート名およびプロシージャー名は変更されています。詳しくは、「*Rational Business Developer V7.5 Generation for z/VSE feature Reference Manual*」(SC19-2539-00) を参照してください。
- EGL ビルド・スクリプト FDAPREP は、すべての VAGen の OS/400 準備テンプレートを置き換えるため、OS/400 は以下の表に含まれていません。FDAPREP ビルド・スクリプトのコメントは、どの VAGen 準備テンプレートがビルド・スクリプトのそのセクションの基礎になったかを示します。

表 158. MVS の準備テンプレートおよびプロシージャー

環境	パーツ型 およびデータベース	VisualAge Generator の テンプレートおよび プロシージャー	EGL ビルド・ スクリプト
MVS CICS	プログラム - DB2 なし	EFK2MPCB ELATCL	FDATCL
	プログラム - DB2 あり	EFK2MPCA ELAPTCLB	FDAPTCL の後に FDABIND
	マップ・グループ - 印刷サービス	EFK2MMCA ELACL	FDACL

表 158. MVS の準備テンプレートおよびプロシージャ (続き)

環境	パーツ型 およびデータベース	VisualAge Generator の テンプレートおよび プロシージャ	EGL ビルド・ スクリプト
	マップ・グループ - フォーマット・モジ ュール	EFK2MMTF ELAL	FDALINK
MVS パッチ	プログラム - DB2 なし	EFK2MPBA ELACL	FDABCL
	プログラム - DL/I および DB2 の両方 あり	EFK2MPBB ELAPCLB	FDAPCL の後に FDABIND
	プログラム - DB2 のみあり	EFK2MPBC ELAPCLB	FDAPCL の後に FDABIND
	マップ・グループ - 印刷サービス	EFK2MMCA ELACL	FDACL
IMS/VS	プログラム - DL/I のみあり	EFK2MPIC ELACL	FDABCL
	プログラム - DL/I および DB2 作業デ ータベースあり	EFK2MPID ELACLB	FDABCL の後に FDABIND
	プログラム - DL/I および DB2 の両方 あり	EFK2MPIE ELAPCLB	FDAPCL の後に FDABIND
	マップ・グループ - MFS の印刷サービ ス	EFK2MMCB ELACL	FDACL
	マップ・グループ - MFS、/mfstest 生成 オプションあり	EFK2MMST MFSTEST	FDAMFS
	マップ・グループ - MFS、/nomfstest 生 成オプションあり	EFK2MMSU MFSUTL	FDAMFS
	マップ・グループ - フォーマット・モジ ュール	EFK2MMTF ELAL	FDALINK
IMS BMP	プログラム - DL/I のみあり	EFK2MPIA ELACL	FDABCL
	プログラム - DL/I および DB2 の両方 あり	EFK2MPIB ELAPCLB	FDAPCL の後に FDABIND
	マップ・グループ - SEQ および GSAM の印刷サービス	EFK2MMCA ELACL	FDACL

表 158. MVS の準備テンプレートおよびプロシージャー (続き)

環境	パーツ型 およびデータベース	VisualAge Generator の テンプレートおよび プロシージャー	EGL ビルド・ スクリプト
	マップ・グループ - MFS の印刷サービ ス	EFK2MMCB ELACL	FDACL
	マップ・グループ - MFS、/mfstest 生成 オプションあり	EFK2MMST MFSTEST	FDAMFS
	マップ・グループ - MFS、/nomfstest 生 成オプションあり	EFK2MMSU MFSUTL	FDAMFS
すべての MVS 環境	再リンク・プログラ ム	EFK2MPRE ELARLINK	FDALINK
	テーブル	EFK2MMCA ELACL	FDACL

ランタイム・テンプレート

表 159は、MVS バッチおよび IMS BMP 環境の基本ランタイム JCL (ジョブ制御言語) の生成で使用される VAGen ランタイム・テンプレートを示しています。表 160は、生成されたランタイム JCL (ジョブ制御言語) 内での DD ステートメントの作成に使用されるファイルおよびデータベース割り振りテンプレートを示しています。表 161は、XFER ステートメントまたは DXFR ステートメントを使用して呼び出されたか転送された別のプログラム、または EZEDEST か EZEDESTP を使用するプログラムのために、追加の DD ステートメントが必要になる場合があることを示すために使用される、ファイルおよびデータベースの割り振りプレースホルダー・テンプレートを示しています。表 162は、OS/400 環境の制御言語 (CL) の生成で使用される VAGen のランタイム・テンプレートを示しています。4 つの表はすべて、対応する VAGen および EGL の情報を示しています。

注: VSE バッチは表に含まれていません。これは、VSE 用の生成フィーチャーが VisualAge Generator と同様のテンプレートを使用するためです。ただし、テンプレート名は変更されています。詳しくは、「*Rational Business Developer V7.5 Generation for z/VSE feature Reference Manual*」(SC19-2539-00) を参照してください。

表 159. MVS ランタイム・テンプレート

環境	プログラム・タイプ およびデータベース	VisualAge Generator のランタイム JCL (ジョブ制御言語) テンプレート	EGL ランタイム JCL (ジョブ制御言 語) テンプレート
MVS バッチ	呼び出し先プログラ ム	EFK2MEBA	fda2meba.tpl
	メインプログラム - データベースなし	EFK2MEBE	fda2mebe.tpl

表 159. MVS ランタイム・テンプレート (続き)

環境	プログラム・タイプ およびデータベース	VisualAge Generator のランタイム JCL (ジョブ制御言語) テンプレート	EGL ランタイム JCL (ジョブ制御言語) テンプレート
	メインプログラム - DL/I のみ	EFK2MEBC	fda2mebc.tpl
	メインプログラム - DB2 のみ	EFK2MEBD	fda2mebd.tpl
	メインプログラム - DL/I および DB2	EKF2MEBB	fda2mebb.tpl
IMS BMP	呼び出し先プログラム	EFK2MEBA	fda2meba.tpl
	メインプログラム - DL/I のみ	EFK2MEIB	fda2meib.tpl
	メインプログラム - DL/I および DB2	EFK2MEIA	fda2meia.tpl

表 160. MVS ファイルおよびデータベース割り振りテンプレート

環境	ファイルまたはデー タベース・タイプ	VisualAge Generator のランタイム JCL (ジョブ制御言語) テンプレート	EGL ランタイム JCL (ジョブ制御言語) テンプレート
MVS バッチおよび IMS BMP	MVS バッチ内の DL/I データベース	EFK2MDLI	fda2mdli.tpl
	シリアル・ファイ ル、索引付きファイ ル、または相対ファ イルの VSAM 入力 または VSAMRS 入 力	EFK2MVS1	fda2mvsi.tpl
	シリアル・ファイ ル、索引付きファイ ル、または相対ファ イルの VSAM 出力 または VSAMRS 出 力	EFK2MVSO	fda2mvso.tpl
	シリアル・ファイ ルの SEQ 入力または SEQRS 入力	EFK2MSDI	fda2msdi.tpl
	シリアル・ファイ ルの SEQ 出力または SEQRS 出力	EFK2MSDO	fda2msdo.tpl
	シリアル・ファイ ルの GSAM 入力	EFK2MGSI	fda2mgsi.tpl

表 160. MVS ファイルおよびデータベース割り振りテンプレート (続き)

環境	ファイルまたはデータベース・タイプ	VisualAge Generator のランタイム JCL (ジョブ制御言語) テンプレート	EGL ランタイム JCL (ジョブ制御言語) テンプレート
	シリアル・ファイル の GSAM 出力	EFK2MGSO	fda2mgso.tpl
	IMS BMP 内の GSAM ファイル	EFK2MIMS	fda2mims.tpl

表 161. MVS ファイルおよびデータベースの割り振りプレースホルダー・テンプレート

環境	ファイルおよびデータベースの割り振り プレースホルダー・ タイプ	VisualAge Generator のランタイム JCL (ジョブ制御言語) テンプレート	EGL ランタイム JCL (ジョブ制御言語) テンプレート
MVS パッチおよび IMS BMP	EZEAPP への XFER または DXFR	EFK2MEZA	fda2meza.tpl
	特定のアプリケーションへの CALL、XFER、DXFR または RT 生成オプションは、特定のアプリケーションに転送します	EFK2MCAL	fda2mcad.tpl
	アプリケーションは EZEDEST または EZEDESTP を使用します	EFK2MEZD	fda2mezad.tpl

表 162. OS/400 ランタイム・テンプレート

環境	テンプレートの目的	VisualAge Generator のランタイム・ テンプレート	EGL ランタイム JCL (ジョブ制御言語) テンプレート
OS/400	これがクライアントによって呼び出される最初のサーバー・プログラムである場合に、クライアント/サーバーのジョブにライブラリーを追加し、コミットメント制御を開始する CL を提供します	EFK24EBC	fda24ebc.tpl
	発生したエラーを処理するランタイム CL に対し、エピソードを提供します	EFK24EEC	fda24eec.tpl

他のランタイム情報

このセクションは、以下の表で構成されています。

- ランタイム環境変数、466 ページの表 163
- vgj.properties、468 ページの表 164

ランタイム環境変数

VAGen ランタイム環境変数は、CICSOS/2 の場合の生成済み COBOL、およびワークステーション環境の場合の生成済み C++ で使用されます。ほとんどの VAGen ランタイム環境変数には、対応する EGL ランタイム・プロパティがありません。対応がある場合でも、EGL ランタイム・プロパティの値は異なる値を持つか、または少し異なる意味になります。また、多数の新規 EGL ランタイム・プロパティがあります。したがって、このセクションの資料を EGL ランタイム・プロパティの作成の補助として使用してください。ただし、オンライン・ヘルプのすべての EGL ランタイム・プロパティを入念に確認し、設定する必要がある追加のプロパティがあるかどうかを決定します。

表 163. ランタイム環境変数

VAGen のランタイム環境変数	EGL ランタイム・プロパティ
BTRINTF	使用されません -- CICS OS/2 のみ
CICSCOBCOPY	使用されません -- CICS OS/2 のみ
CICSRGRP	使用されません -- CICS OS/2 のみ
CICSCOBOL	使用されません -- CICS OS/2 のみ
CICSRD	使用されません -- CICS OS/2 のみ
CICSWRK	使用されません -- CICS OS/2 のみ
COBPATH	使用されません -- CICS OS/2 のみ
CSODIR	使用されません
CSO_DUMP_CONV	使用されません
CSO_DUMP_DATA	使用されません
CSOTIMEOUT (時間は秒単位)	cso.cicsj2c.timeout (時間はミリ秒単位)
CSOTROPT	トレースする必要があるものに応じて tcpiplistener.trace.flag または vgj.trace.type
CSOTROUT	トレースする必要があるものに応じて tcpiplistener.trace.file または vgj.trace.device.spec
DB2INSTANCE	使用されません
DLITROPT	非サポート -- リモートの DL/I のみ
DLITROUT	非サポート -- リモートの DL/I のみ
DPATH	使用されません
ELAPATH	使用されません
ELARTRDB_####、ここで、#### は CICS トランザクション・コード	使用されません -- CICS OS/2 のみ
EZERGRGL_###、ここで ### は NLS 言語コード	vgj.datemask.gregorian.long.locale

表 163. ランタイム環境変数 (続き)

VAGen のランタイム環境変数	EGL ランタイム・プロパティ
EZERGRGS_xxx、ここで xxx は NLS 言語コード	vgj.datemask.gregorian.short.locale
EZERJULL_xxx、ここで xxx は NLS 言語コード	vgj.datemask.julian.long.locale
EZERJULS_xxx、ここで xxx は NLS 言語コード	vgj.datemask.julian.short.locale
EZERNLS	vgj.nls.code
EZERSQLDATE	非サポート -- EGL が JDBC を使用します
EZERSQLDB	vgj.jdbc.database.SN、ここで SN はサーバー名です。値の書式は EZERSQLDB とは異なります
EZERSQLM1	非サポート -- EGL が JDBC を使用します
EZERSQLM2	使用されません -- EGL が JDBC を使用します
EZERSQLMF	使用されません -- EGL が JDBC を使用します
EZERSQLUS	非サポート -- EGL が JDBC を使用します
FCEOPT	使用されません -- C++ 生成のみ
FCETROPT	使用されません -- C++ 生成のみ
FCWCOMP	使用されません -- C++ 生成のみ
FCWDB2DIR	使用されません -- EGL が JDBC を使用します
FCWDBNAME_programName	vgj.jdbc.default.database.programName。値の書式は FCWDBNAME とは異なります
FCWDBNOOP	使用されません -- 配布された CICS のみ
FCWDBPASSWORD	vgj.jdbc.default.password
FCWDBUSER	vgj.jdbc.default.userid
FCWDBVERSION (Oracle バージョン)	使用されません -- EGL が JDBC を使用します
FCWDPATH (テーブルおよびリソース関連のディレクトリー)	使用されません。テーブルはクラスパスに入っている必要があります。リソース関連は vgj.ra. プロパティになります
FCWFIODB	使用されません -- 配布された CICS のみ
FCWLIBPATH	使用されません -- C++ 生成のみ
FCWMAKE	使用されません -- C++ 生成のみ
FCWRSC (raf ファイル名)	使用されません。リソース関連は vgj.ra. プロパティになります
FCWOPT (日付マスクのあるマップ・フィールド)	非サポート
FCWTRDB_tttt、ここで tttt は CICS トランザクション・コード	使用されません -- 配布された CICS のみ
FCWTROPT	vgj.trace.type。値および意味は FCWTROPT とは異なります

表 163. ランタイム環境変数 (続き)

VAGen のランタイム環境変数	EGL ランタイム・プロパティ
FCWTROUT	vgj.trace.device.spec. vg.trace.device.option=2 も指定する必要があります
INFORMIXDIR	非サポート -- ODBC のみ
MDLROOT	非サポート -- VAGen テンプレートのみ
ORACLE_HOME	使用されません -- EGL が JDBC を使用し ます
RMTDLI_PARTNER_LU	非サポート -- MVS 上のリモートの DL/I の み
RMTDLI_PARTNER_TP	非サポート -- MVS 上のリモートの DL/I の み
RMTDLI_SERVER_ENV	非サポート -- MVS 上のリモートの DL/I の み
VSEDLI_CFG	非サポート -- VSE 上のリモートの DL/I の み
VSEDLI_TRACE	非サポート -- VSE 上のリモートの DL/I の み

vgj.properties

VAGen の `vgj.properties` ファイルは、ワークステーション環境の Java 生成に使用されます。ほとんどの VAGen の `vgj.properties` 変数には、対応する EGL ランタイム・プロパティがあります。ただし、場合によっては、EGL ランタイム・プロパティ (設定とも呼ばれる) の値は異なる値を持つか、または少し異なる意味になります。また、多数の新規 EGL ランタイム・プロパティがあります。したがって、このセクションの資料は EGL ランタイム・プロパティの作成の補助として使用します。ただし、オンライン・ヘルプのすべての EGL ランタイム・プロパティを入念に確認し、設定する必要がある追加のプロパティがあるかどうかを決定します。

表 164. *vgj.properties*

VAGen <code>vgj.properties</code>	EGL ランタイム・プロパティ
<code>cso.application.xxx</code> 、ここで <code>xxx</code> はサーバー・グループです	使用されません -- EGL はリンクージ・プロパティ・ファイルを使用します
<code>cso.linkagetable.xxx</code> 、ここで <code>xxx</code> はリンクージ・テーブル名です	<code>cso.linkageOptions.LO</code> 、ここで <code>LO</code> はリンクージ・オプション・パーツ名であり、対応するリンクージ・プロパティ・ファイルの名前は <code>LO.properties</code> になります
<code>cso.serverLinkage.xxx.yyy</code> 、ここで <code>xxx</code> はサーバー・グループであり、 <code>yyy</code> は属性名です。	使用されません -- EGL はリンクージ・プロパティ・ファイルを使用します
<code>vgj.datemask.gregorian.long.xxx</code> 、ここで <code>xxx</code> は NLS 言語コード	<code>vgj.datemask.gregorian.long.locale</code>
<code>vgj.datemask.julian.long.xxx</code> 、ここで <code>xxx</code> は NLS 言語コード	<code>vgj.datemask.julian.long.locale</code>
<code>vgj.java.command</code>	<code>vgj.java.command</code>

表 164. *vgj.properties* (続き)

VAGen <i>vgj.properties</i>	EGL ランタイム・プロパティ
<i>vgj.jdbc.database.SN</i> 、ここで <i>SN</i> はサーバー名です	<i>vgj.jdbc.database.SN</i> 、ここで <i>SN</i> はサーバー名です。値の書式は EZERSQLDB とは異なります
<i>vgj.jdbc.default.database</i>	<i>vgj.jdbc.default.database</i> または <i>vgj.jdbc.default.database.programName</i>
<i>vgj.jdbc.default.database.user.id</i>	<i>vgj.jdbc.default.userid</i>
<i>vgj.jdbc.default.database.user.password</i>	<i>vgj.jdbc.default.password</i>
<i>vgj.jdbc.drivers</i>	<i>vgj.jdbc.drivers</i>
<i>vgj.nls.code</i>	<i>vgj.nls.code</i>
<i>vgj.nls.number.decimal</i>	<i>vgj.nls.number.decimal</i>
<i>vgj.powerserver.location</i>	使用されません
<i>vgj.ra.FN.contable</i> 、ここで <i>FN</i> は論理ファイル名です	<i>vgj.ra.QN.conversionTable</i> 、ここで <i>QN</i> は <i>MQ</i> シリーズのメッセージ・キュー名です。 VAGen の値は一部有効でないものがあります
<i>vgj.ra.FN.filetype</i>	<i>vgj.ra.FN.fileType</i>
<i>vgj.ra.FN.replace</i>	<i>vgj.ra.FN.replace</i>
<i>vgj.ra.FN.sysname</i>	<i>vgj.ra.FN.sysname</i>
<i>vgj.ra.FN.text</i>	<i>vgj.ra.FN.text</i>
<i>vgj.trace.device.option</i>	<i>vgj.trace.device.option</i>
<i>vgj.trace.device.spec</i>	<i>vgj.trace.device.spec</i>
<i>vgj.trace.type</i>	<i>vgj.trace.type</i> 。値および意味は FCWTROPT とは異なります

付録 C. マイグレーション・ツールからのメッセージ

ここでは、マイグレーション・ツールから出されるメッセージについて説明します。メッセージは、接頭部別に次に示すセクションに記載されています。

- HPT.EGL.00xxx - ステージ 1 共通メッセージ
- HPT.EGL.01xxx - VisualAge for Java のステージ 1
- HPT.EGL.02xxx - VisualAge Smalltalk のステージ 1
- IWN.MIG - EGL のステージ 2 および 3

それぞれのメッセージ番号の末尾にある文字は、メッセージの重大度を示す接尾部です。

- *i* - 情報メッセージ。状況を示したり、VisualAge Generator と EGL の言語が異なるために、マイグレーション・ツールがマイグレーション中に情報を除去したことを通知したりします。ユーザー・アクションは必要ありません。
- *w* - 警告メッセージ。起こりうる問題を示します。例えば、マイグレーション・ツールが EGL 構文について最適な推測を行った場合などです。検証または生成時にエラーが検出された場合のみ、ユーザー・アクションが必要です。
- *e* - エラー・メッセージ。マイグレーション・ツールは、EGL 構文について適切な推測を行うことができませんでした。ユーザー・アクションを行って、欠落している情報、または不完全な情報を提供する必要があります。
- *t* - トレース・メッセージ。情報メッセージよりも詳細な状況を示します。トレース・メッセージには、コミット点が記録された時点についての詳しい情報があります。トレース・メッセージは説明を必要としないので、本マイグレーション・ガイドには記載されていません。

VisualAge Generator から EGL マイグレーション・ツールへのメッセージ - ステージ 1

ステージ 1 マイグレーション・ツールは、サンプルとして出荷されます。メッセージは、サンプル・ツール自体の中では翻訳されていません。ただし、本マイグレーション・ガイドの中では、サンプルに付属するメッセージが翻訳されています。

ステージ 1 共通メッセージ

次のメッセージは、VisualAge for Java と VisualAge Smalltalk の両方の VAGen マイグレーション・ツールに共通です。

HPT.CM.215.e ファイル *filename* を開くことができません。戻りコードは *returnCode* (*returnCodeText*) です。

説明: 指定されたファイルを開くことができません。*returnCode* と *returnCodeText* が、理由を示しています。*returnCode* 2 は、見つからないファイルを示します。

ユーザーの処置: マイグレーション・ツールに有効なファイルを指定します。

HPT.EGL.0001.w テーブル名 *tableName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、テーブルの名前を自動的に変更しません。

ユーザーの処置: テーブルの名前と、それに対するすべての参照 (プログラムのテーブルおよび追加レコード・リスト、ロジック・ステートメント、データ項目編集ルーチン、マップ編集ルーチン、および UI 編集テーブルの中での参照など) を変更する必要があります。

VAGen 以外によるテーブル名の参照 (CICS プログラム定義など) をすべて変更してください。この代わりに、マイグレーションが完了するまで待ち、EGL で DataTable を名前変更し、EGL **alias** プロパティを使用して元のテーブル名を指定することもできます。

HPT.EGL.0002.w マップ・グループ名 *mapGroupName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、マップ・グループの名前を自動的に変更しません。

ユーザーの処置: マップ・グループの名前、マップ・グループ内のすべてのマップの名前、およびマップ・グループへのすべての参照 (プログラムのメイン・マップ・グループ、またはヘルプ・マップ・グループとしての参照を含む) を変更する必要があります。VAGen 以外によるマップ・グループ名の参照 (CICS プログラム定義など) をすべて変更したことを確認してください。この代わりに、マイグレーションが完了するまで待ち、EGL で FormGroup を名前変更し、EGL **alias** プロパティを使用して元のマップ・グループ名を指定することもできます。

HPT.EGL.0003.w プログラム名 *programName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、プログラムの名前を自動的に変更しません。

ユーザーの処置: プログラムの名前と、それに対するすべての参照 (CALL、DXFR、および XFER の各ステートメントでの参照、およびリンケージ・テーブル・パーツの中での参照など) を変更する必要があります。また、このプログラムに対応するバインド制御パーツ、またはリンク・エディット・パーツの名前も変更します。VAGen 以外によるプログラム名の参照 (CICS プログラムおよびトランザクション定義など) をすべて変更してください。この代わりに、マイグレーションが完了するまで待ち、EGL でプログラムを名前変更し、EGL **alias** プロパティを使用して元のテーブル名を指定することもできます。

HPT.EGL.0004.w 制御パーツ名 *partName* は予約語です。名前を変更する必要があります。

説明: 指定された制御パーツ名はドット表記を使用していますが、ドットの前にある名前が予約語です。マイグ

レーション・ツールは、ドットの前にある名前がプログラム名であり、この制御パーツがプログラムに密接に関連していることを想定します。マイグレーション・ツールはプログラムの名前を変更しないので、ドット表記の制御パーツの名前も変更しません。

ユーザーの処置: プログラムの名前と、それに対するすべての参照 (CALL、DXFR、および XFER の各ステートメントでの参照、およびリンケージ・テーブル・パーツの中での参照など) を変更する必要があります。また、このプログラムに対応するバインド制御パーツ、またはリンク・エディット・パーツの名前も変更します。VAGen 以外によるプログラム名の参照 (CICS プログラムおよびトランザクション定義など) をすべて変更してください。この代わりに、マイグレーションが完了するまで待ち、EGL でプログラムを名前変更し、EGL **alias** プロパティを使用して元のテーブル名を指定することもできます。EGL 予約語のリストについては、295 ページの『付録 A. 予約語』を参照してください。

HPT.EGL.0005.w UI レコード *recordName* は、予約語であるか、# または @ シンボルで始まっています。名前を変更する必要があります。

説明: ステージ 1 マイグレーション・ツールは、UI レコードの名前を変更しません。しかし、ステージ 2 ツールは、ステージ 2 の「接頭部の名前変更」を使用してレコードの名前を変更します。ステージ 2 ツールには、VGUI レコード用の **alias** プロパティも含まれているため、EGL で生成された出力内の名前は、VisualAge Generator の出力内の名前と同一になります。ステージ 3 ツールも、VGUI レコードが含まれているファイルを名前変更します。単一ファイル・モードでマイグレーションを行う場合、マイグレーション・ツールは同じ変更を行います。

ユーザーの処置: なし。ステージ 2 マイグレーション・ツールによるレコード名の変更を可能にします。これにより、レコードとファイル名に対するすべての参照も変更されます。

HPT.EGL.0006.i *preferenceFile* のマイグレーションにより、*outputList* が作成されます。

説明: *preferenceFile* のマイグレーションにより、*outputList* が作成されます。可能な出力は、マイグレーション・プラン、レポート、およびデータベースの更新です。

ユーザーの処置: なし。

HPT.EGL.0007.w 現行フィルターを基にしたマイグレーション・ファイルは作成されませんでした。

説明: 現行フィルターを基にしたマイグレーション・ファイルは作成されませんでした。

ユーザーの処置: フィルター設定を変更します。他のエラー・メッセージに、このエラーについての詳しい情報が記載されている可能性がありますので、確認してください。

HPT.EGL.0008.e 設定オプション *preferenceOption* に対して *PreferenceValue* は無効な値です。

説明: この値は、この設定オプションに対して無効です。

ユーザーの処置: 設定ファイル内の設定オプション値を変更します。

HPT.EGL.0009.e マイグレーション・セット *migrationSetName* に対しては、スパン・マップ接尾部の設定値を指定する必要があります。

説明: 指定されたマイグレーション・セットには、複数のプロジェクトまたは複数のパッケージにわたるマップ・グループが 1 つ以上含まれています。マップ・グループに必要な EGL プロジェクトまたはパッケージをマイグレーション・ツールが作成できるように、スパン・マップ接尾部の設定値を指定する必要があります。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集します。「マッピング」ページの「スパン・マップ (Spanning Maps)」セクションで、「プロジェクト接尾部 (Project suffix)」フィールドと「パッケージ接尾部 (Package suffix)」フィールドの値を指定します。詳しくは、Java の場合は 156 ページの『「マッピング」ページ』、Smalltalk の場合は 183 ページの『「マッピング」ページ』を参照してください。

HPT.EGL.0010.w マイグレーション・アクションが要求されませんでした。

説明: ステージ 1 マイグレーション・ツールの出力オプションを選択していません。

ユーザーの処置: 1 つ以上のオプションを選択します。これらのオプションにより、マイグレーション・プラン・ファイルの作成、レポートの作成、またはデータベースの更新が可能です。

HPT.EGL.0011.i マイグレーション・セット *migrationSetName* のデータベース・クリーンアップを開始します。

説明: マイグレーション・データベースは、指定されたマイグレーション・セットに関する情報をすでに含んでいます。マイグレーション・ツールは、一連の新しい設定を使用してステージ 1 を実行する準備のために、マイグレーション・セット情報を削除しました。

ユーザーの処置: なし。

HPT.EGL.0012.i マイグレーション・セット *migrationSetName* のデータベース・クリーンアップが完了しました。

説明: マイグレーション・データベースは、指定されたマイグレーション・セットに関する情報をすでに含んでいます。マイグレーション・ツールは、一連の新しい設定を使用してステージ 1 を実行する準備のために、マイグレーション・セット情報を削除しました。

ユーザーの処置: なし。

HPT.EGL.0013.e それぞれの名前変更規則に、固有の順序値が必要です。

説明: 複数の名前変更規則の順序番号が同じです。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、それぞれの規則のオーダー番号が固有になるように名前変更規則を変更します。

HPT.EGL.0014.i マイグレーション・セット *migrationSetName-migrationSetVersion* は、*n* 個のエラー・メッセージ、*n* 個の警告メッセージ、および *n* 個の情報メッセージを生成しました。

説明: *n* は、指定されたマイグレーション・セットに対してステージ 1 マイグレーション・ツールが発行したメッセージの数です。情報メッセージの数は、メッセージ HPT.EGL.0014.i を含みます。

ユーザーの処置: なし。

HPT.EGL.0015.e 得られた EGL プロジェクト名 *eglProjectName* は、無効文字 *characterList* を含んでいます。名前変更規則を変更してください。

説明: ユーザーが指定した名前変更規則を使用して、ステージ 1 マイグレーション・ツールは提示された EGL プロジェクト名を作成しましたが、この名前は EGL プロジェクトの命名規則に適合しません。無効な文字は、*characterList* に示されています。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、有効な EGL プロジェクト名が得られるようにプロジェクトの名前変更規則を変更します。名前変更規則を変更する際には、マッピング・コンテキストとして **both** を指定する名前変更規則の影響を必ず検討してください。

HPT.EGL.0016.e 得られた EGL パッケージ名

eglPackageName は、無効文字 *characterList* を含んでいます。名前変更規則を変更してください。

説明: ユーザーが指定した名前変更規則を使用して、マイグレーション・ツールは提示された EGL パッケージ名を作成しましたが、この名前は EGL パッケージの命名規則に適合しません。無効な文字は、*characterList* に示されています。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、有効な EGL パッケージ名が得られるようにパッケージの名前変更規則を変更します。名前変更規則を変更する際には、マッピング・コンテキストとして **both** を指定する名前変更規則の影響を必ず検討してください。

HPT.EGL.0017.e 得られた EGL プロジェクト名

eglProjectName は、末尾がピリオド (.) であってはなりません。名前変更規則を変更してください。

説明: ユーザーが指定した名前変更規則を使用して、マイグレーション・ツールは提示された EGL プロジェクト名を作成しましたが、この名前の末尾がピリオドです。この名前は、EGL プロジェクトの命名規則に適合しません。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、有効な EGL プロジェクト名が得られるようにプロジェクトの名前変更規則を変更します。名前変更規則を変更する際には、マッピング・コンテキストとして **both** を指定する名前変更規則の影響を必ず検討してください。また、ストリング・コンテキストとして **any**、**back**、または **token** を指定する名前変更規則が与える影響も考慮してください。

HPT.EGL.0018.e 得られた EGL パッケージ名

eglPackageName は、先頭が数字、または末尾がピリオド (.) であってはなりません。名前変更規則を変更してください。

説明: ユーザーが指定した名前変更規則を使用して、マイグレーション・ツールは提示された EGL パッケージ名を作成しましたが、この名前の末尾がピリオドであるか、先頭が数字です。この名前は、EGL パッケージの

命名規則を満たしていません。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、有効な EGL パッケージ名が得られるようにパッケージの名前変更規則を変更します。名前変更規則を変更する際には、マッピング・コンテキストとして **both** を指定する名前変更規則の影響を必ず検討してください。

HPT.EGL.0019.i マイグレーション・フィーチャー

featureName *versionName* がロードされました。

説明: この情報メッセージは、現在 Java ワークスペースまたは Smalltalk イメージにロードされている、マイグレーション・フィーチャーの名前とバージョンを示します。Java 上の VAGen の場合、このメッセージを繰り返し出すことで、VAGen ユーティリティ機能に関する情報を提供します。Java 上の VAGen と Smalltalk 上の VAGen のどちらの場合でも、このメッセージを繰り返し出すことで、ロードされる予約語リストのバージョンが示されます。

ユーザーの処置: なし。

HPT.EGL.0020.i マイグレーション・フィーチャー

featureName *versionName* はロードされませんでした。*listOfNames* がロードされていません。

説明: この情報メッセージは、Java ワークスペースに追加する必要がある、または Smalltalk イメージにロードする必要があるマイグレーション・フィーチャーの名前とバージョンを示します。ただし、1 つ以上の Java パッケージまたは Smalltalk アプリケーションが、マイグレーション・フィーチャーに対して予期されるバージョンではありません。*listOfNames* は、現在ロードされている、予期されるバージョンではない Java パッケージまたは Smalltalk アプリケーションのリストです。

ユーザーの処置: ステージ 1 マイグレーション・ツールを変更していなければ、Java の場合はマイグレーション・フィーチャーの再追加、Smalltalk の場合はマイグレーション・フィーチャーの再ロードを試みてください。ステージ 1 マイグレーション・ツールを変更した場合、このメッセージは、変更した Java パッケージまたは Smalltalk アプリケーションを知らせるためのものです。

HPT.EGL.0021.e 外部化された EGL 予約語リストをロードできません。 *fileName* を検査してください。

説明: 指定された *fileName* を見つけることができません。*fileName* はファイルの絶対パス名です。149 ページ

の『第 4 章 ステージ 1 - Java からの抽出』、または 177 ページの『第 5 章 ステージ 1 - Smalltalk からの抽出』の説明のとおりステージ 1 マイグレーション・ツールをインストールした場合、ファイルの名前とロケーションは正しく設定されます。

ユーザーの処置: EGL 予約語のファイルが正しいパスにあり、正しいファイル名が指定されていることを確認します。正しくない場合は、ステージ 1 マイグレーション・ツールのインストール手順を確認します。

HPT.EGL.0022.e パーツ *partName* の外部ソース形式は無効です。

説明: ステージ 1 マイグレーション・ツールは、リポジトリから外部ソース形式の抽出を 3 回試行しましたが、パーツの開始タグと終了タグが有効な対 (例えば、:record と :erecord) ではありません。ステージ 1 マイグレーション・ツールは処理を継続します。ただし、指定された *partName* に対する正しい外部ソース形式がマイグレーション・データベースに格納されません。

ユーザーの処置: リポジトリ内の外部ソース形式を調べて、パーツ内に明らかなエラーがないかどうか確認します。リモート・リポジトリを使用している場合は、リポジトリをローカル・ドライブにコピーして、ステージ 1 マイグレーション・ツールの再実行を試してください。問題を解決できない場合には、IBM サポートに連絡してください。

HPT.EGL.0023.e 重複するパーツがあるために、パーツ *partName* の外部ソース形式は無効です。

説明: 同じパーツ型のパーツに、同じパーツ名のものが 1 つ以上あります。マイグレーション・ツールは、マイグレーション・データベースに格納する外部ソース形式を判別できません。

ユーザーの処置: マイグレーション・セット内に重複するパーツ名がないように、マイグレーション・セットを変更します。

HPT.EGL.0024.e 派生マップ・グループ名 *mapGroupName* は別のパーツ名と同じです。

VisualAge for Java のステージ 1

次のメッセージは、VisualAge Generator to EGL マイグレーション・ツールの VisualAge for Java バージョンでのみ出されます。

説明: 浮動域仕様が存在する場合、VisualAge Generator にはマップ・グループ・パーツのみが必要です。EGL には通常、書式グループ・パーツが必要です。ステージ 1 マイグレーション・ツールは、マップ名のマップ・グループ部分を使用してマップ・グループ・パーツの作成を試みました。ただし、マップ・グループ名はマイグレーション・セットの他のパーツの 1 つと同じ名前です。ステージ 1 マイグレーション・ツールは、マイグレーション・データベースを変更せずに処理を終了します。

ユーザーの処置: マップ名を変更して、その名前のマップ・グループ部分がマイグレーション・セットの他のパーツと競合しないようにします。また、そのマップ・グループを使用するすべてのプログラムを変更して、新しいマップ・グループ名を指定します。ステージ 1 を再度実行してください。

HPT.EGL.0025.e マイグレーション・データベースに、必要なテーブルがすべて揃っていません。 *setupdatabase.bat* および *setuptables.bat* を再実行してください。

説明: マイグレーション・データベースは EGL バージョン 7.1 で変更され、いくつかのテーブルを追加することが必要になりました。また、DB2 の前提条件も変更されました。

ユーザーの処置: DB2 が正しいレベルでインストールされていることを確認してください。その後、*setupdatabase.bat* ファイルおよび *setuptables.bat* ファイルを実行して、マイグレーション・データベースとそのテーブルを定義します。詳しくは、547 ページの『マイグレーション・データベースの作成』を参照してください。

HPT.EGL.0026.i マイグレーション・データベース・サーバーのバージョンは *versionNumber* です。

説明: この情報メッセージは、DB2 のバージョン情報を示します。

ユーザーの処置: なし。

HPT.EGL.0101.e 現行パッケージ名 *vagenPackageName* から、EGL パッケージ名 *eglPackageName* が作成されましたが、名前の先頭が記号 # または @ であるか、予約語 *reservedWordList* を使用していません。名前を変更する必要があります。

説明: EGL パッケージ名のドット表記で、いずれかのワードとして EGL 予約語を使用することはできません。

ユーザーの処置: ステージ 1 名前変更規則を使用して、EGL の命名上の制約に違反しない EGL パッケージ名を作成します。EGL 予約語のリストについては、295 ページの『付録 A. 予約語』を参照してください。作成される EGL パッケージ名の先頭が # または @ 記号にならないようにします。

HPT.EGL.0102.e マイグレーション・セット *migrationSetName* - *migrationSetVersion* は、プロジェクト *projectName* のバージョン *projectVersion1* と *projectVersion2* を参照しています。マイグレーション・セットは作成されませんでした。

説明: マイグレーション・ツールは、指定されたマイグレーション・セット・バージョンの上位 PLP プロジェクトを展開しました。展開された上位 PLP プロジェクトは、同じプロジェクト名の複数バージョンを含んでいます。マイグレーションは継続できません。

ユーザーの処置: PLP プロジェクトを使用している場合は、PLP チェーンに各プロジェクトのバージョンがただ 1 つ含まれるように、上位 PLP プロジェクトと、そのプロジェクトが参照する下位 PLP プロジェクトを変更します。マイグレーション・プラン・ファイルを手作業で作成した場合は、マイグレーション・セットに対して各プロジェクトのバージョンがただ 1 つ指定されるように、マイグレーション・プラン・ファイルを変更します。

HPT.EGL.0103.e データベース・ドライバのロード中にエラーが発生しました。ドライバ: *driverName*。

説明: ステージ 1 設定ファイルに指定されたデータベース・ドライバを見つけることができませんでした。

ユーザーの処置: 以下のいずれかの状態が発生した可能性があります。

- ステージ 1 設定ファイルで指定されているデータベース情報 (データベース・ドライバ、データベース名、スキーマ、ユーザー ID、またはパスワード) に

誤りがある可能性があります。これらの値の設定方法について詳しくは、159 ページの『「実行」ページ』を参照してください。

- **VAGenToEGLMigration** クラスの **Properties** で指定されているクラスパスに、db2java.zip ファイルが含まれていません。 **Properties** の設定方法について詳しくは、168 ページの『ステージ 1 ツールの実行』を参照してください。
- **PATH** または **CLASSPATH** 環境変数が長すぎます。SQL 関連のディレクトリーを、**PATH** または **CLASSPATH** 環境変数の先頭に移動してください。または、db2jdbc.dll を *db2InstallationDirectory¥SQLLIB¥BIN* から *vajavaInstallationDirectory¥ide¥program* ディレクトリーにコピーしてください。デフォルトの *db2InstallationDirectory* は *c¥Program Files¥IBM¥SQLLIB* です。Java4.0 の場合、デフォルトの *vajavaInstallationDirectory* は *c¥Program Files¥IBM¥VisualAge* です。
- DB2 のバージョンは Java のステージ 1 ではサポートされません。例えば、DB2 バージョン 9.x を使用しようとしている可能性があります。Java のステージ 1 でサポートされる DB2 のバージョンについては、547 ページの『付録 G. マイグレーション・データベース』を参照してください。

HPT.EGL.0104.e データベースへの接続中にエラーが発生しました。データベース: *databaseName*。

説明: ステージ 1 マイグレーション・ツールは、マイグレーション・データベースに接続できませんでした。

ユーザーの処置: 指定されたデータベースが作成済みであることを確認してください。また、ステージ 1 設定ファイル内のユーザー ID とパスワードの設定値を検討して、これらが正しいことを確認してください。

HPT.EGL.0105.e データベース接続のクローズ中にエラーが発生しました。

説明: ステージ 1 マイグレーション・ツールは、マイグレーション・データベースへの接続をクローズできませんでした。

ユーザーの処置: ステージ 1 マイグレーション・ツールは、データベース接続のクローズを試行する前にすべてのコミットを行います。VisualAge Generator をシャットダウンすれば、接続のクローズを強制できます。

HPT.EGL.0106.e メソッド *methodName*
(*optionalErrorCode*) で、リポジトリへの
アクセス中にエラーが発生しました。

説明: ステージ 1 マイグレーション・ツール用の指定されたメソッドが、リポジトリにアクセスできませんでした。

ユーザーの処置: リモート・リポジトリを使用している場合は、リポジトリがアクセス可能であり、ネットワークに問題がないことを確認してください。また、以下の *optionalErrorCode* ベースの方法も試してください。

- *optionalErrorCode* は **ToolEnv** です。**IBM VisualAge Generator EGL** マイグレーション・プロジェクトを展開し、**com.ibm.vgj.mig** パッケージを展開する。
「**VAGenToEGLMigration** クラスを右クリックして、「プロパティ」をクリックする。「**クラスパス**」タブをクリックします。「**追加ディレクトリー・パス (Extra directories path)**」の隣にある「**編集**」をクリックします。次の相対パス・エントリーを選択します。

```
..\IBM IDE Utility local implementation\
```

次に、「**ディレクトリーの追加**」をクリックし、以下のフォーマットで明示パスを使用するようにエントリーを変更します。

```
drive:\VAJavaInstallDirectory\ide\  
project_resources\  
IBM IDE Utility local implementation
```

ここで、*drive* はドライブで、*VAJavaInstallDirectory* は VisualAgefor Java がインストールされているディレクトリーです。「**OK**」を 2 回クリックして「**クラスパス**」ページに戻ります。「**今すぐ計算する (Compute Now)**」をクリックして「**完全なクラスパス (Complete class path)**」情報を更新します。その後、「**OK**」をクリックします。

- *optionalErrorCode* はありません。ログ・ファイルまたはコンソール・ウィンドウに、追加のメッセージがあります。これらのメッセージを使用して、問題を解決します。

問題が解決したら、マイグレーションを再試行します。

HPT.EGL.0107.e XML ファイル *fileName* の書き込み
中にエラーが発生しました。

説明: ステージ 1 マイグレーション・ツールは、指定されたファイル名を書き込むことができませんでした。

ユーザーの処置: そのファイルのために十分なスペースが使用可能であることを確認してください。その後、マイグレーションを再試行します。

HPT.EGL.0108.w 名前 *partName* に無効な空白がある
ため、*partType* パーツはマイグレーションから除外されました。このパーツは、*packageName*, *versionName* にあります。

説明: VisualAge Generator では、パーツ名の末尾にブランクを含むパーツを作成できます。この場合、一方のパーツ名の末尾にブランクがあることを除いて同じ名前をもつパーツが 2 つ作成されることがよくあります。名前が少し異なっているので、これらは重複パーツではありません。ただし、外部ソース形式ファイル内では、パーツ名以外のソース・コードが異なっても、これらのパーツ名は同一です。ステージ 1 マイグレーション・ツールは、ブランクを含むパーツ名をマイグレーション・セットから省略します。これは、他の VAGen パーツがブランクで終わるパーツ名を参照できないからです。類似した名前のパーツが存在しない場合は、この手法によって、他のパーツが参照できないパーツがマイグレーションされないようになります。類似した名前のパーツが存在する場合は、この手法によって、ステージ 2 マイグレーション・ツールが正しいパーツ定義を EGL に変換します。

ユーザーの処置: なし。ただし、VisualAge Generator 内でパーツを検討して、類似した名前のパーツが存在するかどうか判別することをお勧めします。VAGen パーツ・ブラウザーを使用し、*partName** を指定してマイグレーション・セット内のすべてのパーツを検索します。

HPT.EGL.0109.e 予期しない例外が発生しました:
javaExceptionStackTrace

説明: ステージ 1 のマイグレーション・ツールの実行中に、予期しないエラーが発生しました。

ユーザーの処置: *javaExceptionStackTrace* を検討してください。エラーに応じて、無視できる場合も訂正できる場合もあります。例えば、次のようになります。

- 変換できない文字が置換文字によって置き換えられたことを示すメッセージは、無視できます。このメッセージは、外部ソース形式の中で無効文字が検出された場合に発生します。マイグレーション・データベース内では、文字はブランクに置き換えられます。ステージ 1 マイグレーション・ツールは処理を継続します。このマイグレーション・データベースは、ステージ 2 と 3 に使用できます。
- EGL ファイル名を格納するには SQL 列が短すぎることを示すメッセージの場合は、問題を訂正できません。この場合、マイグレーション・データベースの情報は無効なので、ステージ 1 マイグレーション・ツールは処理を停止します。SQL 表定義を変更して列の長さを増やし、ステージ 1 を再度実行することに

よって、問題を訂正できます。ただし、SQL 列の長さを増やす前に、マイグレーション後にこれらの長い名前をスクロールすることになって構わないかどうか、また長い名前が EGL の制限を超えないかどうかを検討してください。名前変更規則または SQL 列を変更した後、ステージ 1 マイグレーションを再度実行します。

- `javaExceptionStackTrace` に以下の SQL メッセージのいずれかが含まれている場合は、マイグレーションの実行に使用されている DB2 ユーザー ID がマイグレーション・データベースに対して十分な権限を持っていることを確認してください。

- SQL2311N The user does not have the authority to run the Run Statistics utility on table
- SQL0551N "userid" does not have the privilege to perform operation "sqlOperation" on object "migrationTable"

この問題は、マイグレーション・データベースを作成したユーザー ID が、ステージ 1 マイグレーション・ツールを実行しているユーザー ID と異なる場合に発生することがあります。必要な権限および特権の詳細については、547 ページの『DB2 権限要件』を参照してください。

- メッセージが ADMIN_CMD または RUNSTATS に SQL 問題があることを示している場合、これは、必要なレベルの DB2 をインストールする前に、マイグレーション・データベースを作成したことを意味しています。DB2 の正しいレベルについては、547 ページの『付録 G. マイグレーション・データベース』を参照してください。正しいレベルのものをインストールしてから、`setupdatabase.bat` および `setuptables.bat` を再度実行します。
- 詳細については、このメッセージの後に記載されている情報を確認してください。また、コンソールに HPT.EGL.0110.e などの追加メッセージが表示されていないか調べて、そのメッセージの指示に従ってください。

問題を解決できない場合は、IBM サポートに連絡して支援を受けてください。

HPT.EGL.0110.e プロジェクト `projectName` バージョン `versionName` は、リポジトリでは定義されません。

説明: ステージ 1 マイグレーション・ツールは、PLP プロジェクトのチェーンを含むマイグレーション・セットの上位 PLP を展開しました。指定されたプロジェクト・バージョンは、PLP チェーンでは参照されるがリポジトリでは使用不可のバージョンであるか、空のバージョンです。

ユーザーの処置: プロジェクトおよびバージョンをマイグレーション・セットに組み込む必要があるかどうか判別してください。組み込む必要があれば、プロジェクトの要求されたバージョンがリポジトリからパージされていないかどうか確認します。パージされている場合、プロジェクト・バージョンを復元してから、もう一度マイグレーションします。問題を解決できない場合は、IBM サポートに連絡して支援を受けてください。

HPT.EGL.0111.e 元の VAGen プロジェクト名

`projectName` から、空の派生 EGL プロジェクト名が作成されました。名前変更規則を変更してください。

説明: マイグレーション・ツールが、ユーザーによって指定された名前変更規則を使用して、提示された EGL プロジェクト名を作成しましたが、この名前には文字が含まれていません。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、有効な EGL プロジェクト名が得られるようにプロジェクトの名前変更規則を変更します。名前変更規則を変更する際には、マッピング・コンテキストとして **both** を指定する名前変更規則の影響を必ず検討してください。

HPT.EGL.0112.e 元の VAGen パッケージ名

`packageName` から、空の派生 EGL パッケージ名が作成されました。名前変更規則を変更してください。

説明: マイグレーション・ツールが、ユーザーによって指定された名前変更規則を使用して、提示された EGL パッケージ名を作成しましたが、この名前には文字が含まれていません。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、有効な EGL パッケージ名が得られるようにパッケージの名前変更規則を変更します。名前変更規則を変更する際には、マッピング・コンテキストとして **both** を指定する名前変更規則の影響を必ず検討してください。

HPT.EGL.0113.e マイグレーション・セット

`migrationSetName - migrationSetVersion` に `number` の重複パーツが含まれています。重複は許可されていません。

説明: ステージ 1 マイグレーション・ツールがパーツのエディションに正しいソース・コードを関連付けるためには、パーツ名が固有名になっている必要があります。指定されたマイグレーション・セットには、重複するパーツ名が含まれています。Number は、パーツ名が重複しているパーツの数を指定しています。メッセージ

HPT.EGL.0115.e は、パーツ名とそのパーツが存在するパッケージ名を示しています。パーツ名のペアごとに 1 つのメッセージが出されます。

ユーザーの処置: マイグレーション・セットからのプロジェクト・バージョンは、まだワークスペース内にあります。VAGen パーツ・ブラウザーから、「ツール (Tools)」->「重複パーツの表示 (Show Duplicate Parts)」を選択して、名前が重複しているパーツを判別します。問題を訂正し、プロジェクトにバージョンを付け、マイグレーション・セットの定義を更新してから、ステージ 1 のツールを再度実行します。

HPT.EGL.0114.e プロジェクト *projectName* バージョン *projectVersionName* のパッケージ *packageName* バージョン *packageVersionName* は、リポジトリ内で定義されていません。

説明: ステージ 1 マイグレーション・ツールは、PLP プロジェクトのチェーンを含むマイグレーション・セットの上位 PLP を展開しました。指定されているパッケージ・バージョンは、PLP チェーン内の指定されているプロジェクト・バージョンによって参照されますが、リポジトリ内で使用可能になっていません。

packageVersionName は、「Missing - (*versionDateTimeStamp*)」のフォーマットになっている場合があります。

ユーザーの処置: パッケージ・バージョンをプロジェクトに組み込む必要があるかどうかを判別してください。組み込む必要があれば、パッケージの要求されたバージョンがリポジトリからパージされていないかどうか確認します。パージされている場合、パッケージ・バージョンを復元してから、もう一度マイグレーションします。問題を解決できない場合は、IBM サポートに連絡して支援を受けてください。

HPT.EGL.0115.e 重複するパーツ *partName* が、*packageName1* ではタイプ *partType1* で、*packageName2* ではタイプ *partType2* で検出されました。

説明: ステージ 1 マイグレーション・ツールがパーツのエディションに正しいソース・コードを関連付けるためには、パーツ名が固有名になっている必要があります。指定されているパーツ名は、1 つ以上のパッケージに異なるパーツ型で存在している可能性があります。メッセージ HPT.EGL.0115.e は、パーツ名のペアごとに繰り返し出されます。

ユーザーの処置: マイグレーション・セットからのプロジェクト・バージョンは、まだワークスペース内にあります。VAGen パーツ・ブラウザーから、「ツール

(Tools)」->「重複パーツの表示 (Show Duplicate Parts)」を選択して、名前が重複しているパーツを判別します。問題を訂正し、プロジェクトにバージョンを付け、マイグレーション・セットの定義を更新してから、ステージ 1 のツールを再度実行します。

HPT.EGL.0116.w プロジェクト *projectName* は、プロジェクト名フィルターと一致しています。このプロジェクトには、いずれかのバージョン名フィルターと一致するバージョンがありません。

説明: 指定されている PLP プロジェクトがリポジトリから検出されましたが、指定されているバージョンではありません。

ユーザーの処置: マイグレーション設定ファイル内のリポジトリ・フィルターのバージョン名を訂正してください。バージョン名には、大/小文字の区別があります。

HPT.EGL.0118.w VAGen NLS 設定を判別できません。出力ファイルには、デフォルトのエンコード方式が使用されました。

説明: マイグレーション・ツールは、ステージ 1 からの出力ファイルのエンコード方式の設定に使用するロケールを判別できませんでした。VAGen 製品プロジェクトの 1 つが、クラスパスから欠落している可能性があります。ステージ 1 ツールの処理は続行されますが、出力ファイルの使用には障害が出る可能性があります。

ユーザーの処置: 以下のいずれかの方法を使用して、問題を訂正してください。

- 以下のステップを実行してクラスパス情報を設定する。
 1. 「ワークベンチ」ウィンドウの「プロジェクト」タブをクリックする。
 2. 「IBM VisualAge Generator EGL マイグレーション (IBM VisualAge Generator EGL Migration)」プロジェクトにナビゲートする。
 3. マイグレーション・プロジェクトを展開し、**com.ibm.vgj.mig** パッケージを展開する。
 4. パッケージ内で、**VAGenToEGLMigration** クラスを右クリックし、以下のステップを実行する。
 - a. ポップアップ・メニューで「プロパティー (Properties)」をクリックする。
 - b. 「クラスパス」タブをクリックする。
 - c. 「プロジェクト・パス」セクションの隣にある「編集」をクリックする。

- d. 「クラスパス」ウィンドウで、「IBMVisualAge Generator ランタイム (IBM VisualAge Generator Runtime)」を必ず選択するようにします。
 - e. 「OK」をクリックして、「クラスパス」ウィンドウを閉じる。
 - f. 「OK」をクリックして、「プロパティ」ウィンドウを閉じる。
5. **PreferencesUI** クラスについて、ステップ 4 を繰り返す。
6. ステージ 1 マイグレーションを再度実行する。
- 166 ページの『文字セット情報の指定』で説明しているステップを実行してから、ステージ 1 のマイグレーションを再度実行する。

ステージ 1 マイグレーションの再実行を回避するには、以下の変更を行います。

- miglog.xml ファイルと migdebug.xml ファイルを表示するために、ファイル内の次の行にある **encoding** プロパティの値を変更して、必要なエンコード方式を指示します。

```
<?xml version="1.0" encoding="iso-8859-1"?>
```

レポート・ファイルを表示するために、.html ファイル内の次の行にある **charset** プロパティの値を変更して、必要なエンコード方式を指示します。

```
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

あるいは、潜在的に多数の .html レポート・ファイルがある場合には、そのレポートのみを作成するように MigPreferences.xml ファイルで以下の行を指定して、ステージ 1 マイグレーションを再実行します。

```
<verification>
  <generateReport>true</generateReport>
  <reportName>c:\tempMigJ\JTurbo\report\
    MigrationReport.htm</reportName>
</verification>
<dbUpdate>false</dbUpdate>
```

Preferences GUI を使用して MigPreferences.xml ファイルを変更する場合は、その GUI の「実行」ページで「レポートの生成」を選択し、「データベースの更新 (Update database)」の選択は解除します。

VisualAge Smalltalk のステージ 1

次のメッセージは、VisualAge Generator to EGL マイグレーション・ツールの VisualAge Smalltalk バージョンでのみ出されます。

HPT.EGL.0201.e 現行アプリケーション名

vagenApplicationName から、EGL パッケージ名 *eglPackageName* が作成されましたが、先頭が # または @ であるか、予約語 *reservedWordList* を使用しています。名前を変更する必要があります。

説明: EGL パッケージ名のドット表記で、いずれかのワードとして EGL 予約語を使用することはできません。

ユーザーの処置: ステージ 1 名前変更規則を使用して、EGL の命名上の制約に違反しない EGL パッケージ名を作成します。EGL 予約語のリストについては、295 ページの『付録 A. 予約語』を参照してください。作成される EGL パッケージ名の先頭が # または @ 記号にならないようにします。

HPT.EGL.0202.e マイグレーション・セット

migrationSetName が構成マップ *configurationMapName* を参照していますが、このマップはリポジトリ内で定義されていません。

説明: ステージ 1 マイグレーション・ツールは、指定されたマイグレーション・セットの上位構成マップを展

開しました。しかし、ツールが上位構成マップを展開し、必須のマップとアプリケーションのチェーンを展開したときに、1 つ以上の必須マップがライブラリー内で使用できませんでした。

ユーザーの処置: 必要なマップをマイグレーション・セットに組み込む必要があるかどうか判別してください。組み込む必要があれば、構成マップの要求されたバージョンがライブラリーからバージョンされていないかどうか確認します。バージョンされている場合は、要求されている構成マップを修復し、マイグレーションを再度実行します。

HPT.EGL.0203.e ProgramContext が、データベース・エラー errorMessage を検出しました。

説明: データベース・エラーが発生しました。考えられる問題は、無効なスキーマ名、ユーザー権限の制限、誤ったレベルの DB2、または SQL 表の欠落です。

ユーザーの処置: errorMessage を検討します。エラーに応じて、必要なアクションが異なる可能性があります。例えば、次のようになります。

- errorMessage に SQL エラー・コード SQL0440N に関する情報が含まれている場合は、誤ったレベルの DB2 がインストールされていることを示していま

す。DB2 の正しいレベルについては、547 ページの『付録 G. マイグレーション・データベース』を参照してください。正しいレベルのものをインストールしてから、`setupdatabase.bat` および `setuptables.bat` を再度実行します。

- `errorMessage` に以下の SQL メッセージのいずれかが含まれている場合は、マイグレーションの実行に使用されている DB2 ユーザー ID がマイグレーション・データベースに対して十分な権限を持っていることを確認してください。

- SQL2311N The user does not have the authority to run the Run Statistics utility on the table
- SQL0551N "userid" does not have the privilege to perform operation "sqlOperation" on object "migrationTable"

この問題は、マイグレーション・データベースを作成したユーザー ID が、ステージ 1 マイグレーション・ツールを実行しているユーザー ID と異なる場合に発生することがあります。必要な権限および特権の詳細については、547 ページの『DB2 権限要件』を参照してください。

- マイグレーション設定ファイル内の情報を確認してください。

問題を解決できない場合は、IBM サポートに連絡して支援を受けてください。

HPT.EGL.0204.e データベースへの接続中にエラーが発生しました。`ErrorMessage`。

説明: 接続時にデータベース・エラーが発生しました。考えられる問題は、無効なユーザー ID やパスワード名、または無効なデータベース名です。

ユーザーの処置: マイグレーション設定ファイルを訂正してください。問題が解消しない場合は、IBM サポートに連絡して支援を受けてください。

HPT.EGL.0205.i マイグレーションにより、構成マップ `configMapName` の v 個のバージョンから、 n 個のマイグレーション・セットが作成されました。設定ファイルは、バージョン深さを d と指定しています。

説明: ステージ 1 マイグレーション設定ファイルは、マイグレーションするバージョンの数を d と指定しています。ステージ 1 マイグレーション・ツールは、 d 個のマイグレーション・セット (指定した構成マップのバージョンごとに 1 つずつ) を作成する必要があります。しかし、マイグレーション・ツールは n に示されている数のマイグレーション・セットしか作成していません。 v に示されている数は、マイグレーション・ツ

ルがライブラリー内で検出した、指定された構成マップのバージョン数です。 v が d より小さい場合は、構成マップのバージョン数が予想した数より少なかったことを示しています。この場合、 n と v は等しくなり、これは構成マップのすべてのバージョンからマイグレーション・セットが作成されたことを示しています。 v が d より大きい場合は、ライブラリー内に構成マップのバージョンがさらに多く存在することを示します。この場合、 n と d は等しくなり、これはバージョン深さの設定が満たされたことを示しています。

ユーザーの処置: なし。

HPT.EGL.0206.e マイグレーション・セット

`migrationSetName` にロード・エラーが発生しました。

説明: マイグレーション・セットをイメージにロードできませんでした。この原因としては、マイグレーション・セットに重複するパーツ名が存在することが考えられます。

ユーザーの処置: システム・トランスクリプトを調べて、エラーの原因を判別してください。問題を訂正して、ステージ 1 マイグレーションを再度実行してください。

HPT.EGL.0207.w 名前 `partName` に無効な空白があるため、`partType` パーツはマイグレーションから除外されました。このパーツは、アプリケーション `applicationName`、`versionName` にあります。

説明: VisualAge Generator では、パーツ名の末尾にブランクを含むパーツを作成できます。この場合、一方のパーツ名の末尾にブランクがあることを除いて同じ名前をもつパーツが 2 つ作成されることがよくあります。名前が少し異なっているので、これらは重複パーツではありません。ただし、外部ソース形式ファイル内では、パーツ名以外のソース・コードが異なっても、これらのパーツ名は同一です。ステージ 1 マイグレーション・ツールは、ブランクを含むパーツ名をマイグレーション・セットから省略します。これは、他の VAGen パーツがブランクで終わるパーツ名を参照できないからです。類似した名前のパーツが存在しない場合は、この手法によって、他のパーツが参照できないパーツがマイグレーションされないようになります。類似した名前のパーツが存在する場合は、この手法によって、ステージ 2 マイグレーション・ツールが正しいパーツ定義を EGL に変換します。

ユーザーの処置: なし。ただし、VisualAge Generator 内でパーツを検討して、類似した名前のパーツが存在するかどうか判別することをお勧めします。VAGen パー

ツ・ブラウザを使用し、*partName** を指定してマイグレーション・セット内のすべてのパーツを検索します。

HPT.EGL.0208.e データベース列

schemaName.tableName.columnName のデータは切り捨てられました。

説明: 1 つ以上の名前変更規則により、対応する SQL 列に収まらない長さの EGL プロジェクト名、パッケージ名、またはバージョン名が作成されました。

ユーザーの処置: より短い EGL プロジェクト名、パッケージ名、またはバージョン名が作成されるように、名前変更規則を変更します。また、DB2 表を変更して、SQL 列の長さを増やすこともできます。ただし、SQL 列の長さを増やす前に、マイグレーション後にこれらの長い名前をスクロールすることになっても構わないかどうか、また長い名前が EGL の制限を超えないかどうかを検討してください。名前変更規則または SQL 列を変更した後、ステージ 1 マイグレーションを再度実行します。

HPT.EGL.0211.e 元の VAGen 構成マップ名

configurationMapName から、空の派生 EGL プロジェクト名が作成されました。名前変更規則を変更してください。

説明: マイグレーション・ツールが、ユーザーによって指定された名前変更規則を使用して、提示された EGL プロジェクト名を作成しましたが、この名前には文字が含まれていません。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、有効な EGL プロジェクト名が得られるように構成マップの名前変更規則を変更します。名前変更規則を変更する際には、マッピング・コンテキストとして **both** を指定する名前変更規則の影響を必ず検討してください。

HPT.EGL.0212.e 元の VAGen アプリケーション名

applicationName から、空の EGL パッケージ名が作成されました。(Original VAGen application name *applicationName* - results in a derived empty EGL package name.) 名前変更規則を変更してください。

説明: マイグレーション・ツールが、ユーザーによって指定された名前変更規則を使用して、提示された EGL パッケージ名を作成しましたが、この名前には文字が含まれていません。

ユーザーの処置: ステージ 1 マイグレーション設定ファイルを編集して、有効な EGL パッケージ名が得られるようにアプリケーションの名前変更規則を変更します。名前変更規則を変更する際には、マッピング・コンテキストとして **both** を指定する名前変更規則の影響を必ず検討してください。

HPT.EGL.0213.i マイグレーション・セット

migrationSetName - versionName のメッセージ・テーブル・ユーティリティが *count* の新規関連行を追加しました

説明: この情報メッセージは、プログラム・パーツ内のメッセージ・テーブル接頭部に基いて、プログラムに関連するものとして追加されたメッセージ・テーブルの数を示します。メッセージ・テーブルはステージ 1 で関連するものとして追加されるため、ステージ 3 マイグレーション・ツールは EGL ビルド・パスを正しく設定することができ、また、プログラム・パーツ用の正しい **import** ステートメントを組み込むことができます。このメッセージは、ステージ 1 マイグレーション・ツールの実行時に「メッセージ・テーブル関連付けの修復 (Repair Message Table Associates)」でこのオプションを選択した場合にも、表示されます。

ユーザーの処置: なし。

VisualAge Generator から EGL マイグレーション・ツールへのメッセージ - ステージ 2

以下のメッセージは、ステージ 2 で生成されます。EGL 予約語に関して必要な名前変更を行うまで、メッセージには常に VAGen パーツ名が挿入されます。

IWN.MIG.0001.e 外部ソース形式ファイル *fileName* の構文解析中に例外が発生しました - 無効な外部ソース形式ヘッダー

説明: マイグレーション・ツールは、VisualAge Generator 4.5 からエクスポートされた外部ソース形式のみを処理します。指定された外部ソース形式ファイルの先頭行に、VisualAge Generator 4.5 外部ソース形式ファイルの正しいヘッダーがありません。

ユーザーの処置: 外部ソース形式ファイルを VisualAge Generator 4.5 にインポートします。これにより、現行パーツが VisualAge Generator 4.5 形式に変換されます。その後、VisualAge Generator 4.5 を使用してパーツをエクスポートし、マイグレーションを再実行します。

IWN.MIG.0002.e 外部ソース形式ファイル *fileName*, *partType*, *partName* の構文解析中の例外 - *exceptionText*

説明: VisualAge Generator からの外部ソース形式構文の解析中に問題が発生しました。この問題の考えられる原因は次のとおりです。

- 引用符のミスマッチ。例えば、データ項目の通貨フィールドで引用符を使用した場合など。
- 制御パーツ内のコメント区切りのミスマッチ。
- 使用しているロケールに対して無効な各国語文字。例えば、2 バイト・ロケール用に設定されていないワークステーション上で、中国語などの 2 バイト文字を使用する VAGen ソース・コードをマイグレーションしようとした場合など。

ユーザーの処置: VisualAge Generator 内でパーツを訂正し、外部ソース形式を再度エクスポートします。その後、ステージ 2 マイグレーション・ツールを実行して、ファイルを処理します。VisualAge Generator 内でパーツを訂正できない場合は、IBM サポートに連絡して援助を受けてください。このファイルの外部ソース形式のソースをあらかじめ用意しておいてください。

IWN.MIG.0003.e ファイル *fileName*, *partType*, *partName* の EGL への変換中に例外が発生しました - *exceptionText*

説明: EGL ソースの作成中に問題が発生しました。*exceptionText* が、発生した問題を具体的に示しています。

ユーザーの処置: IBM サポートに連絡して支援を受けてください。このファイルの外部ソース形式のソースをあらかじめ用意しておいてください。

IWN.MIG.0004.e 出力ファイル *fileName1* と UI レコードは、名前が同じです - UI レコードは *fileName2* に入ります。

説明: 単一ファイルのマイグレーション中は、*fileName1* は指定済みの出力ファイルです。外部ソース・フォーマットのファイルは、指定済み出力ファイルと同じ名前を持つ UI レコードを含んでいます。このファイルは他のいくつかのパーツも含んでいます。マイグレーション・ツールは、UI レコードを処理する前に指定済み出力ファイルに他のパーツを置きました。このツールは、続いて、*fileName2* という名前の EGL ファイルに UI レコードを置きました。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。

ユーザーの処置: *fileName1* に異なる名前を付けます。VGUI レコード名と同じ名前を *fileName2* に付けます。

IWN.MIG.0047.i マイグレーション・セット *Name_version* - マイグレーションが開始されました。

説明: これは、マイグレーション・ツールから出される、状況を示す情報メッセージです。

ユーザーの処置: なし。

IWN.MIG.0048.i マイグレーション・セット *Name_version* - マイグレーションが完了しました。

説明: これは、マイグレーション・ツールから出される、状況を示す情報メッセージです。

ユーザーの処置: なし。

IWN.MIG.0049.i EGL *projectName*, *packageName*, *fileName* の *partType* *partName* - マイグレーションが開始されました。

説明: これは、マイグレーション・ツールから出される、状況を示す情報メッセージです。*partType* は、Program、Map Group、または Table のいずれかの値になります。指定された *partName* の関連パーツが、同時にマイグレーションされます。関連パーツは、マイグレーション・データベース内の情報に基づいて、*partName* と同じファイル、あるいは異なるプロジェクト、パッケージ、またはファイルに入っています。プログラムのマイグレーションが開始されると、それぞれの関連したマップ・グループがマイグレーションされ、次にそれぞれの関連したテーブルがマイグレーションされます。最後に、残りの関連パーツ (レコード、共用項目、および関数) がマイグレーションされます。

ユーザーの処置: なし。

IWN.MIG.0050.i プログラム *programName* - その他の関連パーツのマイグレーションが開始されました。

説明: これは、マイグレーション・ツールから出される、状況を示す情報メッセージです。プログラムのマイグレーションが開始されると、それぞれの関連したマップ・グループがマイグレーションされ、次にそれぞれの関連したテーブルがマイグレーションされます。最後に、残りの関連パーツ (レコード、共用項目、および関数) がマイグレーションされます。メッセージ IWN.MIG.0050.i は、プログラムの残りの関連パーツのマイグレーションが開始されたときに出力されます。

ユーザーの処置: なし。

IWN.MIG.0051.e マイグレーション・セット *planName*, *partType*, *partName* の構文解析中に例外が発生しました - 無効な外部ソース形式ヘッダー。

説明: マイグレーション・ツールは、VisualAge Generator 4.5 からエクスポートされた外部ソース形式のみを処理します。指定されたパーツの外部ソース形式の先頭行に、VisualAge Generator 4.5 外部ソース形式ファイルの正しいヘッダーがありません。サンプルのステージ 1 マイグレーション・ツールを変更した場合、またはマイグレーション・データベースをロードするためのステージ 1 マイグレーション・ツールをユーザーが独自に作成した場合に、この問題が起こる可能性があります。

ユーザーの処置: 外部ソース形式ファイルを VisualAge Generator 4.5 にインポートします。これにより、現行パーツが VisualAge Generator 4.5 形式に変換されます。その後、ステージ 1 マイグレーション・ツールを使用してマイグレーション・セットをエクスポートします。

IWN.MIG.0052.e マイグレーション・セット *planName*, *partType*, *partName* の構文解析中に例外が発生しました - *exceptionText*。

説明: VisualAge Generator からの外部ソース形式構文の解析中に問題が発生しました。この問題の考えられる原因は次のとおりです。

- 引用符のミスマッチ。例えば、データ項目の通貨フィールドで引用符を使用した場合など。
- 制御パーツ内のコメント区切りのミスマッチ。
- 使用しているロケールに対して無効な各国語文字。例えば、2 バイト・ロケール用に設定されていないワークステーション上で、中国語などの 2 バイト文字を使用する VAGen ソース・コードをマイグレーションしようとした場合など。

ユーザーの処置: VisualAge Generator 内でパーツを訂正し、ステージ 1 マイグレーション・ツールを再実行してデータベースを訂正します。その後、ステージ 2 マイグレーション・ツールを再実行して、更新したパーツを処理します。VisualAge Generator 内でパーツを訂正できない場合は、IBM サポートに連絡して援助を受けてください。問題のあるパーツを含む小さなリポジトリ (.dat ファイル) または外部ソース・ファイルをあらかじめ用意しておいてください。

IWN.MIG.0053.e マイグレーション・セット *planName*, *partType*, *partName* の EGL への変換中に例外が発生しました - *exceptionText*。

説明: EGL ソースの作成中に問題が発生しました。*exceptionText* が、発生した問題を具体的に示しています。

ユーザーの処置: IBM サポートに連絡して支援を受けてください。このパーツの外部ソース形式のソースをあらかじめ用意しておいてください。

IWN.MIG.0054.e マイグレーション・セット *migrationSetName*, *partType*, *partName* の外部ソース形式が無効です。

説明: 指定されたパーツ用に保管されている外部ソース形式が無効です。マイグレーション・ツールは、指定されたマイグレーション・セット内でその他のパーツの処理を継続します。関連パーツを使用したマイグレーションのために、マイグレーション・ツールは指定されたパーツを使用不可と見なします。マイグレーション・ツールは、指定されたパーツに対して、意図的に無効な EGL をマイグレーション・データベースに格納します。格納される EGL は、EZE_UNKNOWN_PARTTYPE *partName* です。このため、EGL 検証により、「問題」ビューにエラー・メッセージが表示されるようになります。

ユーザーの処置: 指定されたパーツを VisualAge Generator 内で検討してください。パーツの外部ソース形式をエクスポートし、単一ファイル・モードでパーツのマイグレーションを試行します。問題を解決できない場合は、IBM サポートに連絡して支援を受けてください。問題のあるパーツを含む小さなリポジトリ (.dat ファイル) または外部ソース・ファイルをあらかじめ用意しておいてください。

IWN.MIG.0055.e マイグレーション停止 - エラー限度を超えました。

説明: 無効な外部ソース形式のパーツに関するエラーしきい値を超えました。マイグレーション・ツールは処理を停止します。

ユーザーの処置: メッセージ IWN.MIG.0054.e のすべての出現箇所を検討してください。ユーザー独自のツールを作成してマイグレーション・データベースをロードしている場合は、ツールが外部ソース形式のコードをマイグレーション・データベースにロードする方法に問題があることが考えられます。問題の原因の判別に役立つ照会については、547 ページの『付録 G. マイグレーション・データベース』を参照してください。

IWN.MIG.0060.e データベース・ドライバのロード中にエラーが発生しました。ドライバ:
driverName

説明: 指定されたデータベース・ドライバを見つけることができません。

ユーザーの処置: データベース・ドライバ名を訂正します。また、データベース・ドライバのロケーションが正しいことを確認してください。

IWN.MIG.0061.e データベースへの接続中にエラーが発生しました。データベース:
databaseName.errorText

説明: マイグレーション・ツールは、指定されたスキーマ名を使用して、指定されたデータベースに接続できません。 *errorText* フィールドに、接続が失敗した理由についてさらに詳しい情報があります。

ユーザーの処置: データベース名を訂正します。リモート・データベースに接続している場合は、データベースがローカル側でカタログ済みであることを確認してください。

IWN.MIG.0063.e データベース接続のクローズ中にエラーが発生しました。

説明: マイグレーションは正常に完了しましたが、マイグレーション・ツールはデータベース接続をクローズできませんでした。

ユーザーの処置: EGL 開発環境をシャットダウンしてから、データベースをバックアップしてください。

IWN.MIG.0070.e ユーザー出口メソッド
renameUserExitName (partName) が存在しません。

説明: 「ユーザー出口の名前変更」に対して指定した JAR ファイル、または JAR ファイル内のパッケージとクラスが見つかりませんでした。

ユーザーの処置: 「VAGen マイグレーションの設定」で「ユーザー出口の名前変更」について指定した「JAR ファイル・ロケーション」、「パッケージ名」、および

「クラス名」を確認してください。

IWN.MIG.0071.e ユーザー出口メソッド
renameUserExitName(partName) には、必要なメソッド・シグニチャーがありません。

説明: 「ユーザー出口の名前変更」を使用するには、シグニチャー *renameUserExit(String s, Connection c)* を含んだメソッドを組み込む必要があります。「VAGen マイグレーションの設定」で「ユーザー出口の名前変更」に関して指定した JAR ファイル、パッケージ、およびクラスに、このメソッドが存在しませんでした。

ユーザーの処置: クラス定義を検討し、必要なメソッド・シグニチャーを組み込んだことを確認します。また、「VAGen マイグレーションの設定」で「ユーザー出口の名前変更」に関して、「JAR ファイル・ロケーション」、「パッケージ名」、および「クラス名」が正しく指定されていることも確認してください。

IWN.MIG.0072.e ユーザー出口メソッド
renameUserExitName(partName) で Java 言語アクセス制御の実行が指定されていますが、基幹メソッドにアクセスできません。

説明: マイグレーション・ツールは、指定されたユーザー出口クラスの定義にアクセスできません。

ユーザーの処置: ユーザー出口の名前変更 クラスが public として定義されていて、指定されたパッケージ内にあることを確認します。

IWN.MIG.0073.e ユーザー出口メソッド
renameUserExitName(partName) は、例外をスローして予期せず終了しました。

説明: メソッド *renameUserExit(String s, Connection c)* は NULL 値を戻します。マイグレーションは継続します。マイグレーション・ツールは、オリジナルの VAGen パーツ名を使用します。

ユーザーの処置: 「ユーザー出口の名前変更」内のコードの周辺に **try** ブロックを配置します。このメッセージを回避するには、例外が発生した場合にオリジナルの VAGen パーツ名を戻します。

IWN.MIG.0080.i VAGen マイグレーション設定ファイル *pref_store.ini* が見つかりません。デフォルトが想定されます。

説明: VAGen マイグレーション設定ファイルがありません。マイグレーション・ツールは、設定 (例えば、「名前変更接尾部」と「ヘルプ・マップ接尾部」) の値

としてデフォルト値を使用します。この原因としては、マイグレーション中に新しいワークスペースを指定したために、設定が存在しないことが考えられます。設定ファイルは、次の場所にあります。

```
workspace-directory\metadata\plugins
\org.eclipse.core.runtime\settings
\com.ibm.etools.egl.vagenmigration.prefs
```

ユーザーの処置: マイグレーション設定のデフォルト値については、211 ページの『VAGen マイグレーションの設定』を参照してください。

IWN.MIG.0081.i ファイル *fileName* - マイグレーションが完了しました。

説明: マイグレーション・ツールは、指定されたファイルの処理を完了しました。

ユーザーの処置: ログ・メッセージを検討して、マイグレーションの結果を確認してください。

IWN.MIG.0082.e ファイル *fileName* - 必要パラメーターが指定されていません。

説明: 1 つ以上の必要パラメーターが指定されていません。 **-importFile** パラメーターは常に必要です。

-importFile パラメーターが外部ソース形式ファイルを指定している場合は、**-eglFile** パラメーターと **-package** パラメーターも指定する必要があります。

ユーザーの処置: バッチ・コマンド・ファイルを検討して、指定されていないパラメーターを判別してください。パラメーターを追加し、バッチ・コマンド・ファイルを再実行します。

IWN.MIG.0083.e ファイル *fileName* - パラメーター *parmName* に値が割り当てられていません。

説明: *parmName* は、以下のパラメーターのいずれかになります。

- **-importFile**
- **-eglFile**
- **-package**

パラメーター名には大/小文字の区別があります。

ユーザーの処置: バッチ・コマンド・ファイルを訂正し、再実行します。

IWN.MIG.0084.e ファイル *fileName* - パラメーター *parmName*、値 *value* が無効です。

説明: *parmName* は、以下のパラメーターのいずれかになります。

- **-importFile**
- **-eglFile**
- **-package**

パラメーター名には大/小文字の区別があります。

ユーザーの処置: バッチ・コマンド・ファイルを訂正し、再実行します。

IWN.MIG.0085.e ファイル *fileName* - パラメーター・リストの中で無効なパラメーターが渡されました。

説明: バッチ・コマンド・ファイルに問題があります。1 つ以上のパラメーターが誤って入力されました。有効なパラメーターは、**-importFile**、**-eglFile**、**-package**、および **-overwrite** のみです。

ユーザーの処置: バッチ・コマンド・ファイルを訂正し、再実行します。

IWN.MIG.0095.e 関数 *functionName* - EZESCRPT は、マイグレーション対象としてサポートされていません。

説明: 指定された関数は、EZESCRPT 特殊機能語を使用するステートメントを含んでいます。EGL にはこの EZE 関数の置換表現はありません。マイグレーション・ツールは、EZESCRPT を EZE_SCRPT に変換してこの関数のロジックを保持します。ただし、この関数を EGL で使用することはできません。

ユーザーの処置: EGL 関数を検討してください。この関数を使用するプログラムを生成または実行することはできません。

IWN.MIG.0101.e データ項目 *DataItemName* - *editRoutineName* の編集ルーチン・タイプを判別できません。関数であるものと見なされます。

説明: VisualAge Generator は、データ項目のマップ編集ルーチンとして、EZEC10、EZEC11、関数、またはテーブルをサポートします。EGL は、*DataItem* パーツの **validatorFunction** および **validatorDataTable** 両方のプロパティをサポートします。マイグレーション・ツールは、次のようにしてマップ編集ルーチンを変換します。

- EZEC10 と EZEC11 は、**validatorFunction** プロパティにマイグレーションされます。
- *editRoutineName* に指定されたパーツがマイグレーション時に使用可能で、このパーツが関数ならば、*editRoutineName* は **validatorFunction** プロパティにマイグレーションされます。また、**editRoutineName**

が 7 文字より長い場合も、マイグレーション・ツールは編集ルーチンを *validatorFunction* プロパティにマイグレーションします。これは、テーブル名が VisualAge Generator では 7 文字に制限されているからです。

- *editRoutineName* に指定されたパーツがマイグレーション時に使用可能で、このパーツがテーブルならば、*editRoutineName* は **validatorDataTable** プロパティにマイグレーションされます。また、編集メッセージが項目に対して指定されている場合も、マイグレーション・ツールは編集ルーチンを **validatorDataTable** プロパティにマイグレーションします。これは、VisualAge Generator が編集メッセージを EZEC10、EZEC11、またはテーブルのみと組み合わせで使用するからです。
- *editRoutineName* に指定されたパーツがマイグレーション時に使用不可で、*editRoutineName* が 7 文字以下であり、編集メッセージが指定されていない場合、マイグレーション・ツールは *editRoutineName* が関数であると想定し、**validatorFunction** プロパティにマイグレーションします。メッセージ IWN.MIG.0101.e はこの場合にのみ出されます。

ユーザーの処置: 指定された編集ルーチンが関数でなければ、EGL DataItem 定義を変更し、**validatorFunction** プロパティを **validatorDataTable** プロパティに変更します。その他の考慮事項については、83 ページの『共用データ項目のマップ編集ルーチン』にある、編集ルーチンに関する説明を参照してください。

IWN.MIG.0102.e パーツ *partName* は、共用データ項目 *DataItemName* を使用しています - プリミティブ定義にマイグレーションできません。型定義を使用します。

説明: レコード、テーブル、呼び出し先パラメーター・リスト、関数仮パラメーター・リスト、または関数ローカル・ストレージ内で共用データ項目が使用されているときに、VAGen の共用データ項目を EGL のプリミティブ定義にマイグレーションする設定を選択しました。*DataItemName* で指定した項目は、*partName* で指定したパーツ内で使用されています。しかし、マイグレーション時にデータ項目定義が使用不可でした。マイグレーション・ツールは、マイグレーションしたコードが有効になるように、データ項目名を型定義として使用します。

ユーザーの処置: 型定義を使用する必要がある場合は、**import** ステートメントを追加して、DataItem パーツを配置するパッケージをインポートしなければならない場合があります。プリミティブ定義を使用する必要がある場合は、正しい項目特性を使用するように、指定されたパーツを変更します。あるいは、マイグレーション・セット (または、単一ファイル・モードでマイグレーション

している場合は、外部ソース形式ファイル) に共用データ項目を組み込んで、再度マイグレーションを行います。

IWN.MIG.0103.w データ項目 *DataItemName* - 設定が原因で **evensql=y** が無視されます。

説明: 指定したデータ項目パーツは **evensql=y** の VAGen PACK (EGL 10 進数) 項目です。VAGen マイグレーション設定では、**evensql=y** を受け入れないことが指定されています。この設定に基づいて、マイグレーション・ツールは PACK 項目を次に大きい奇数精度 (最大 18) に変換します。マイグレーション・ツールは、次のようにして PACK 項目の EGL 精度を決定します。

- **evensql=n** が項目に指定されると、EGL 精度は常に (VAGen バイト * 2) - 1 と計算され、最大値は 18 です。
- **evensql=y** が項目に指定されると、EGL 精度は設定に基づいて計算されます。

- 「項目または変数の **evensql=y** を受け入れない」設定が選択されている場合、EGL 精度は、
 $(VAGenBytes * 2) - 1$

と計算され、最大値は 18 です。メッセージ IWN.MIG.0103.w はこの場合にのみ出されます。

- この設定が選択されていない (**evensql=y** を受け入れられる) 場合、EGL 精度は、
 $(VAGenBytes * 2) - 2$

と計算され、最大値は 18 です。メッセージ IWN.MIG.0103.w は、出されません。

ユーザーの処置: なし。ただし、SQL WHERE 文節または EGL **prepare** ステートメントにおけるこの項目の使い方を見直すことができます。この項目の定義が、SQL 表定義と正確に一致しないと、パフォーマンスに影響が出る可能性があります。詳しくは、268 ページの『VisualAge Generator 互換モードの使用の中止』および 79 ページの『偶数の長さの PACK データ項目』にある EVENSQLE に関する情報を参照してください。

IWN.MIG.0201.i レコード *recordName* はレベル 77 項目を含んでいます。追加レコード *level77RecordName* を作成します。

説明: VisualAge Generator は、作業用ストレージ・レコード内のレベル 77 項目をサポートします。EGL はレベル 77 項目をサポートしません。EGL は、独立データ項目の定義を許可しません。マイグレーション・ツールは、レベル 77 項目を含む作業用ストレージ・レコードを 2 つの分離した BasicRecord (1 つは非レベル

77 項目を収容し、もう 1 つはレベル 77 項目を収容)に分割します。作業用ストレージ・レコードがレベル 77 項目のみを含んでいる場合、マイグレーション・ツールはレベル 77 BasicRecord のみを作成します。プログラムがレベル 77 項目を含む 1 次作業用ストレージ・レコードを指定している場合、マイグレーション・ツールは、オリジナルの BasicRecord とレベル 77 BasicRecord の両方の宣言をプログラム定義に組み込みます。

ユーザーの処置: なし。プログラムとステートメントのマイグレーション時に *recordName* が使用できない場合の影響など、その他の考慮事項については、87 ページの『レコード内のレベル 77 項目』にある、レコード内のレベル 77 項目に関する説明を参照してください。

IWN.MIG.0202.i レコード *recordName* - *redefinedRecordName* を再定義します。

説明: *recordName* は、再定義されるレコードとして *redefinedRecordName* を指定する VAGen 再定義レコードです。*recordName* は、*redefinedRecordName* によって使用されているものと同じ物理ストレージに対して、異なる項目レイアウトを指定しています。EGL は、レコード・パーツ内の再定義情報を保持しません。この情報は、プログラム内でのみ保持されます。マイグレーション・ツールは、オリジナルの VAGen *redefinedRecordName* の情報を示すコメントを *recordName* に組み込みます。プログラムのマイグレーション時に、*recordName* が使用可能であり、その結果として VisualAge Generator 内でオーバーレイ定義が行われる場合、マイグレーション・ツールは *recordName* 宣言に *redefines* プロパティを組み込みます。

ユーザーの処置: なし。プログラムのマイグレーション時に *recordName* が使用できない場合の影響など、その他の考慮事項については、86 ページの『再定義レコード』にある、再定義レコードに関する説明を参照してください。

IWN.MIG.0203.w レコード *recordName* - 項目が含まれていません。

説明: VisualAge Generator は、項目を含まないレコード・パーツの保管を許容します。ただし、このレコードは無効なので、プログラム内では使用できません。EGL は、フィールドを含まないレコード・パーツを容認します。マイグレーション・ツールは、このレコードをマイグレーションします。

ユーザーの処置: このレコードが引き続き必要かどうか判断してください。必要な場合は、レコードを編集して 1 つ以上のデータ項目を追加します。必要でなければ、レコードを削除します。

IWN.MIG.0204.e レコード *recordName* - 代替仕様レコード *altspecRecord* は使用不可です。SQL 表名を判別できません。

説明: レコード *recordName* は、代替仕様レコード *altspecRecord* を指定しており、このレコードは *recordName* の項目構造を指定します。VisualAge Generator では、SQL レコードの代替仕様レコードは SQL 表名も指定します。EGL では、代替仕様レコードは **embed** キーワードを使用して構造のみを指定します。表名は、それぞれの SQL レコード・パーツ定義内で指定する必要があります。*recordName* のマイグレーション時に、代替仕様レコードとして指定されたレコードがマイグレーション中に使用できませんでした。マイグレーション・ツールは正しい表名を判別できず、**tableNames** プロパティを **###TABLES_NOT_FOUND###** に設定します。*recordName* の定義は無効です。

ユーザーの処置: *recordName* を編集し、VAGen 代替仕様レコード (*altspecRecord*) から **tableNames** および *tableNameVariables* プロパティをコピーします。**tableNames** プロパティは、実際の SQL 表名を指定します。**tableNameVariables** プロパティは、表名ホスト変数を指定します。*recordName* が実際の SQL 表名と SQL 表名ホスト変数を混合して参照する場合は、**tableNames** と **tableNameVariables** の両プロパティを使用できます。その他の考慮事項については、89 ページの『代替仕様レコード』を参照してください。

IWN.MIG.0205.e レコード *recordName* - 代替仕様レコード *altspecRecord* は使用不可です。SQL キー項目を判別できません。

説明: レコード *recordName* は、代替仕様レコード *altspecRecord* を指定しており、このレコードは *recordName* の項目構造を指定します。VisualAge Generator が SQL レコードのデフォルト選択条件を決定する際に、VisualAge Generator は代替仕様レコード内で **key=yes** を指定する項目を、*recordName* 内で指定されたキー項目 (存在する場合) と組み合わせます。キーは、レコード構造内で項目がリストされている順に組み合わせられます。EGL では、代替仕様レコードは **embed** キーワードを使用して構造のみを指定します。すべてのキー項目が、それぞれの SQL レコード・パーツ定義内で指定されている必要があります。*recordName* のマイグレーション時に、代替仕様レコードとして指定されたレコードがマイグレーション中に使用できませんでした。マイグレーション・ツールは正しいキー項目を判別できず、**keyItems** プロパティを **###KEYS_NOT_FOUND###** に設定し、その後に *recordName* からのキー項目 (存在する場合) を付けます。*recordName* の定義は無効です。

ユーザーの処置: *recordName* を編集し、**keyItems** プロパティを変更して、####KEYS_NOT_FOUND#### を項目名のリストに置き換えます。これらの項目名は、代替仕様レコード (*altspecRecord*) 内で **key=yes** を指定したものです。代替仕様レコードのキー項目を、*recordName* 内の VAGen 定義に指定されたキー項目と組み合わせ、**keyItems** プロパティがこれらの項目をレコード構造内での出現順と同じ順序でリストするようにします。代替仕様レコード内で項目が **key=yes** として指定され、かつ *recordName* 内でキー項目として指定されている場合は、**recordName** の *keyItems* の組み合わせリストにその項目を 1 回だけ組み込みます。その他の考慮事項については、89 ページの『代替仕様レコード』を参照してください。

IWN.MIG.0206.i SQL レコード *recordName* - 代替仕様レコードを指定しないキー項目 *keyItem* を含んでいます。

説明: VisualAge Generator 上では、代替仕様レコードを指定しなくても、キー項目を指定する SQL レコードを保管できます。ただし、この状態では、テストおよび生成時に VisualAge Generator はキー項目を無視します。キー項目は、代替仕様レコードも存在する場合に限って意味を持ちます。

ユーザーの処置: なし。このキー項目は、VisualAge Generator 内で無視されていました。マイグレーション・ツールは、マイグレーション時にこの項目を除去します。

IWN.MIG.0207.i レコード *recordName* - レベル 77 項目のみを含む代替仕様レコード *altspecRecord* を指定しています。embed キーワードは省略されます。

説明: *altspecRecord* によって示されたレコードは、レベル 77 項目のみを含む作業用ストレージ・レコードです。*recordName* が作業用ストレージ・レコードを代替仕様として指定する場合、VisualAge Generator は *altspecRecord* からの構造 (非レベル 77 項目) のみを使用します。*altspecRecord* の構造内に項目が存在しないので、マイグレーション・ツールは **embed** キーワードを省略します。

ユーザーの処置: なし。ただし、*recordName* は空のレコードなので削除できます。プログラム内で、*recordName* の参照を必ずすべて削除してください。

**IWN.MIG.0208.e レコード *recordName* - 代替仕様レコード *altspecRecord* が使用できません。
!itemColumnName 変数の SQL 列名を判別できません。**

説明: レコード *recordName* は、代替仕様レコード *altspecRecord* を指定しており、このレコードは *recordName* の項目構造を指定します。VisualAge Generator が SQL レコードのデフォルト選択条件を決定する際に、VisualAge Generator は *!itemColumnName* 変数を対応する SQL 列名に変換します。EGL の場合、*!itemColumnName* 変数はサポートされません。それぞれの SQL レコード・パーツ定義ごとに、デフォルト選択条件の中で SQL 列名を明示的に指定する必要があります。*recordName* のマイグレーション時に、代替仕様レコードとして指定されたレコードがマイグレーション中に使用できませんでした。マイグレーション・ツールは、デフォルト選択条件の中で、1 つ以上の *!itemColumnName* 変数に対応する正しい SQL 列名を判別できませんでした。マイグレーション・ツールは、EGL のデフォルト選択条件の中で *!itemColumnName* を使用します。*recordName* の定義は無効です。

ユーザーの処置: *recordName* を編集し、**defaultSelectCondition** プロパティを変更して、*!itemColumnName* 変数を VAGen 代替仕様レコード (*altspecRecord*) からの対応する SQL 列名に置き換えます。その他の考慮事項については、89 ページの『代替仕様レコード』にある、*!itemColumnName* 変数に関する説明を参照してください。

**IWN.MIG.0209.e レコード *recordName* - 代替仕様レコード *altspecRecord* に項目がありません。
embed キーワードが省略されます。**

説明: レコード *recordName* は、代替仕様レコード *altspecRecord* を指定しており、このレコードは *recordName* の項目構造を指定します。ただし、この代替仕様レコードにはデータ項目がありません。マイグレーション・ツールは、*recordName* の定義から **embed** キーワードを省略します。

ユーザーの処置: なし。ただし、*recordName* と *altspecRecord* を検討して、データ項目を組み込む必要があるか、またはこれら 2 つのレコードを削除できるかを判断する必要があります。プログラム内で、これらのレコードの参照を必ずすべて削除してください。

**IWN.MIG.0210.e レコード *recordName* -
!itemColumnName 変数の列名を判別できません。**

説明: 指定されたレコードのデフォルト選択条件は、1 つ以上の VAGen *!itemColumnName* 変数を使用しています。VAGen *!itemColumnName* 変数は、実際の SQL 列名に対応する SQL レコード定義内の項目名を指定します。VisualAge Generator は、テストおよび生成時に、*!itemColumnName* 変数の実際の SQL 列名を SQL レコードから判別します。EGL は、*!itemColumnName* 変数

をサポートしません。代わりに、EGL の場合は、変更された SQL ステートメントの中で実際の SQL 列名を使用する必要があります。*recordName* によって示されたレコードが VisualAge Generator では無効な場合は、メッセージ IWN.MIG.0210.e が出力されます。この場合、そのレコードは、レコードまたはその代替仕様レコード内で定義されていない *!itemColumnName* 変数を 1 つ以上使用しています。マイグレーション・ツールは、実際の SQL 列名への置換を実行できません。

ユーザーの処置: レコードを編集し、*!itemColumnName* 変数を正しい SQL 列名に変更します。

IWN.MIG.0211.w レコード *recordName*、データ項目 *DataItemName* - 設定が原因で *evensql=y* が無視されます。

説明: 指定されたレコードに、指定された非共用データ項目が含まれています。レコード定義は、この非共用データ項目が、*evensql=y* の VAGen PACK (EGL DECIMAL) 項目であることを指定しています。VAGen マイグレーション設定は、*evensql=y* を受け入れないことを指定しています。この設定に基づいて、マイグレーション・ツールは PACK 項目を次に大きい奇数精度 (最大 18) に変換します。マイグレーション・ツールは、次のようにして PACK 項目の EGL 精度を決定します。

- *evensql=n* が項目に指定されると、EGL 精度は常に (VAGen バイト * 2) - 1 と計算され、最大値は 18 です。
- *evensql=y* が項目に指定されると、EGL 精度は設定に基づいて計算されます。
 - 「項目または変数の *evensql=y* を受け入れない」設定が選択されている場合、EGL 精度は、
(*VAGenBytes* * 2) - 1

と計算され、最大値は 18 です。メッセージ IWN.MIG.0211.w はこの場合にのみ出力されます。

- この設定が選択されていない (*evensql=y* を受け入れる) 場合、EGL 精度は、
(*VAGenBytes* * 2) - 2

と計算され、最大値は 18 です。メッセージ IWN.MIG.0211.w は出力されません。

ユーザーの処置: なし。ただし、SQL WHERE 文節または EGL **prepare** ステートメントにおけるこの項目の使い方を直すことができます。この項目の定義が、SQL 表定義と正確に一致しないと、パフォーマンスに影響が出る可能性があります。詳しくは、268 ページの『VisualAge Generator 互換モードの使用の中止』および 79 ページの『偶数の長さの PACK データ項目』にある

EVENSQLE に関する情報を参照してください。

IWN.MIG.0212.e レコード *recordName* - 代替仕様レコード *altspecRecord* が使用できません。なんらかの項目名で *dliFieldName* プロパティに関する指定変更が必要であるのかどうかを判別できません。

説明: レコード *recordName* は、代替仕様レコード *altspecRecord* を指定しており、このレコードは *recordName* の項目構造を指定します。VisualAge Generator では、DL/I セグメント・レコード内のフィールド名は DL/I PSB 内の名前に一致しなければなりません。EGL では、フィールド名は予約語にすることはできず、また、# または @ シンボルで始めることもできません。EGL ではさらに、DL/I セグメント・レコードのフィールド名を、DL/I PSB で決められている 8 文字の制限より長くできます。DL/I PSB のフィールド名が EGL フィールド名に一致しない場合、EGL は、対応する EGL フィールド名のために DL/I PSB で使用される名前を提供するために *dliFieldName* プロパティを使用します。代替仕様のレコードが利用できないために、マイグレーション・ツールは、EGL 命名規則によりいずれかのフィールド名が名前変更されるかどうかを判別できません。

ユーザーの処置: EGL 命名規則によりレコード内のフィールド名の名前変更が必要であるかどうかを決定するために、*altspecRecord* のレコード定義を検討します。フィールド名が名前変更された場合、対応する *dliFieldName* プロパティを指定するために、*recordName* を編集し、各フィールド名変更用の **embed** キーワードをオーバーライドします。*dliFieldName* プロパティの値はオリジナルの VAGen フィールド名である必要があります。

フィールドに対するオーバーライドを行って **embed** キーワードをコーディングする方法に関する例については、322 ページの表 84 を参照してください。

IWN.MIG.0251.w UI レコード *recordName* は、予約語であるか、# または @ シンボルで始まっています。これは、*newRecordName* に名前変更されました。

説明: UI レコード *recordName* は、EGL 予約語と競合しているか、# または @ シンボルで始まっています。ステージ 2 マイグレーション・ツールは、マイグレーションの設定で指定された「名前変更接頭部」に基づいて、UI レコードを *newRecordName* に名前変更しました。ステージ 2 ツールには、VGUI レコード用の **alias** プロパティも含まれているため、EGL 生成の出力内の名前は、VisualAge Generator の出力内の名前と同一になります。EGL が VGUI レコード名とファイル

名との一致を必要とするため、ステージ 3 マイグレーション・ツールは、VGUI レコードのファイル名を *newRecordName.egl* に変更しました。単一ファイル・モードでマイグレーションを行う場合、マイグレーション・ツールは同じ変更を行います。

ユーザーの処置: なし。

IWN.MIG.0301.e テーブル名 *tableName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、テーブルの名前を自動的に変更しません。

ユーザーの処置: テーブルの名前とその参照すべてを変更する必要があります。この対象には、次の場所での参照が含まれます。

- プログラムの **use** 宣言ステートメント
- プログラムおよび関数のロジック・ステートメント
- データ項目 **validatorDataTable** プロパティ
- 書式フィールド **validatorDataTable** プロパティ
- VGUI レコード・フィールド **validatorDataTable** プロパティ

生成される **DataTable** の名前としてオリジナルのテーブル名を保持する必要がある場合は、**alias** プロパティをオリジナルの **DataTable** 名に設定します。**alias** プロパティを指定しない場合は、CICS プログラム定義など、EGL 以外による **DataTable** 名の参照すべてを必ず変更してください。

IWN.MIG.0302.w テーブル *tableName* - コンテンツが 1 行しかありません。MOVEA ステートメントのソースとしての、その使用法を確認してください。

説明: VisualAge Generator では、単一行のコンテンツを含んだテーブルが MOVEA ステートメントのソースとして使用される場合、そのソースはスカラーとして処理され、ターゲット配列はスカラー・ソースによって完全に初期化されます。これは、VisualAge Generator ドキュメンテーションに反します。このドキュメンテーションによると、テーブルは常に配列として処理される必要があります、その結果、ターゲット配列の最初の要素のみが初期化されることになることと示されています。EGL では、**for** 修飾子の付いた **move** は常に、ある配列から別の配列への移動として処理され、したがって、ターゲット配列の最初の要素のみが初期化されます。

ユーザーの処置: プログラムをレビューして、指定したテーブルを MOVEA ステートメントのソースとして使用したかどうか確認します。MOVEA ステートメントでテーブル名がソース・フィールドの修飾子として使用された場合、マイグレーション・ツールは、メッセージ

HPT.EGL.0710.e も発行します。テーブル名を修飾子として使用しないでテーブル内のフィールドを参照する MOVEA ステートメントを検索する場合には、マイグレーション・データベースに対して DB2 照会を実行します。詳しくは、554 ページの『特定のエラー・メッセージを支援する照会』を参照してください。VisualAge Generator でテーブルが MOVEA のソースとして使用されている場合は、テーブルのソース要素からターゲット配列を初期化するループを使用するようにプログラム・ロジックを変更します。

IWN.MIG.0401.e マップ・グループ (FormGroup) 名 *mapGroupName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、マップ・グループ (FormGroups) の名前を自動的に変更しません。

ユーザーの処置: FormGroup の名前と、その参照すべて (プログラムの **use** 宣言ステートメントでの参照など) を変更する必要があります。生成される FormGroup の名前としてオリジナルのマップ・グループ名を保持する必要がある場合は、**alias** プロパティをオリジナルのマップ・グループ (FormGroup) 名に設定します。

alias プロパティを指定しない場合は、CICS プログラム定義など、EGL 以外による FormGroup 名の参照すべてを必ず変更してください。

IWN.MIG.0402.e マップ・グループ *mapGroupName* - 複数の装置で同じ縦の長さや幅が指定されていますが、異なる浮動域が指定されています。装置: *devicesList*

説明: VisualAge Generator は、同じ装置サイズを持った装置タイプに対して、異なる浮動域サイズを指定することを許容します。EGL の場合は、それぞれの装置サイズに対して浮動域を 1 つだけ指定できます。マイグレーション・ツールは、それぞれの装置サイズごとに浮動域サイズをマイグレーションします。複数の VAGen 装置が変換によって同じ EGL 装置のサイズ仕様およびマージン仕様を持つようになった場合、マイグレーション・ツールは EGL に関して 1 つのエントリーのみを組み込みます。VAGen 装置が同じ EGL 装置のサイズに変換されたものの、異なるマージン仕様を持つようになった場合、マイグレーション・ツールは両方のエントリーを組み込みます。この FormGroup 定義は無効です。このメッセージは、VisualAge Generator では異なる浮動域情報を指定していた、同サイズの装置タイプのグループごとに繰り返されます。

ユーザーの処置: FormGroup を編集して、同サイズの装置のグループごとに 1 つの浮動域仕様を超える、すべての浮動域仕様を削除します。

IWN.MIG.0403.e **formGroup** *FormGroupName* -
FormGroup 内で書式をネストするための
編集が必要です。

説明: 単一ファイル・モードでマイグレーションを行う場合、マイグレーション・ツールは **FormGroup** 内で書式をネストしません。代わりにマイグレーション・ツールは、**FormGroup** に属する書式の名前を示す **EGL** の **use** ステートメントを挿入します。マイグレーション・ツールは、それぞれの書式の先頭と末尾に、所属する **FormGroup** を示すコメントを挿入します。

ユーザーの処置: **FormGroup** を含むファイルを編集して、書式が **FormGroup** 内でネストされるように書式を移動します。**FormGroup** 内の **use** ステートメントが、書式を移動する先を示しています。書式を **FormGroup** 内でネストした後、**use** 宣言ステートメントを除去します。

IWN.MIG.0501.e ヘルプ・マップ・グループ
mapGroupName は、変数フィールドのあ
るマップ *mapName* を含んでいます -
mapName は、プログラムのメインのマッ
プ・グループにある同じマップ名と競合し
ています。

説明: **VisualAge Generator** の場合は、プログラムのメインのマップ・グループとヘルプ・マップ・グループ内で同じマップ名を使用できます。**EGL** の場合は、プログラムの 2 つの **FormGroup** 内で重複する書式名は使用できません。重複する名前をもつ書式がプログラムによって使用されていなくても、この制限が適用されます。プログラムのヘルプ・マップ・グループ内のマップが、プログラムのメインのマップ・グループ内のマップと競合し、ヘルプ・マップが定数フィールドのみを含んでいる場合、マイグレーション・ツールはこれらのマップの名前を変更します。ヘルプ・マップ・グループ内のマップが変数フィールドを含んでいる場合は、その名前がメインのマップ・グループ内にあるマップ名と競合していても、マイグレーション・ツールはマップの名前を変更しません。これは、そのプログラムのメインのマップ・グループとしてそのヘルプ・マップ・グループを指定する他のプログラムが、そのマップを使用している可能性があるからです。

ユーザーの処置: ヘルプ **FormGroup** を編集し、書式の名前を変更します。また、この **FormGroup** を使用するすべてのプログラム内で、書式定義とこの書式への参照すべてを必ず変更します。その他の考慮事項については、97 ページの『マップ名とヘルプ・マップ名』にある、マップ名に関する説明を参照してください。

IWN.MIG.0502.e マップ・グループ *mapGroupName*、
マップ *mapName*、および変数フィールド
mapItemName - *editRoutineName* の編集ル
ーチン・タイプを判別できません。関数が
想定されます。

説明: **VisualAge Generator** は、マップ変数のマップ編集ルーチンとして、**EZEC10**、**EZEC11**、関数、またはテーブルをサポートします。**EGL** は、書式フィールドの **validatorFunction** 関数および **validatorDataTable** プロパティの両方をサポートします。マイグレーション・ツールは、次のようにしてマップ編集ルーチンを変換します。

- **EZEC10** と **EZEC11** は、**validatorFunction** プロパティにマイグレーションされます。
- *editRoutineName* に指定されたパーツがマイグレーション時に使用可能で、このパーツが関数ならば、*editRoutineName* は **validatorFunction** プロパティにマイグレーションされます。また、*editRoutineName* が 7 文字より長い場合も、マイグレーション・ツールは編集ルーチンを **validatorFunction** プロパティにマイグレーションします。これは、テーブル名が **VisualAge Generator** では 7 文字に制限されているからです。
- *editRoutineName* に指定されたパーツがマイグレーション時に使用可能で、このパーツがテーブルならば、*editRoutineName* は **validatorDataTable** プロパティにマイグレーションされます。また、編集メッセージが書式フィールドに対して指定されている場合も、マイグレーション・ツールは編集ルーチンを **validatorDataTable** プロパティにマイグレーションします。これは、**VisualAge Generator** が編集メッセージを **EZEC10**、**EZEC11**、またはテーブルのみと組み合わせるからです。
- *editRoutineName* に指定されたパーツがマイグレーション時に使用不可で、*editRoutineName* が 7 文字以下であり、編集メッセージが指定されていない場合、マイグレーション・ツールは *editRoutineName* が関数であると想定し、**validatorFunction** プロパティにマイグレーションします。メッセージ **IWN.MIG.0502.e** はこの場合にのみ出されます。

ユーザーの処置: 指定された編集ルーチンが関数でなければ、書式フィールドを変更し、**validatorFunction** プロパティを **validatorDataTable** プロパティに変更します。その他の考慮事項については、100 ページの『マップ変数フィールドと編集ルーチン』にある、編集ルーチンに関する説明を参照してください。

IWN.MIG.0503.w マップ・グループ *mapGroupName*、マップ *mapName* - 名前なし変数フィールドが、位置 (*row,column*) で定数フィールドに変換されました。

説明: VisualAge Generator は、名前なし変数フィールドがマップに存在することを許容します。プログラムは、これらの名前なし変数フィールドにアクセスできません。テストおよび生成時に、名前なし変数フィールドは定数に変換されます。1 つ以上のプロパティがデフォルト以外の値なので、マイグレーション・ツールはこの名前なし変数フィールドを定数に変換しました。

ユーザーの処置: 書式定義を検討し、定数フィールドがこのフィールドの正しいマイグレーションであることを確認してください。その他の考慮事項については、103 ページの『名前なしマップ変数フィールド』にある、名前なし変数フィールドに関する説明を参照してください。

IWN.MIG.0504.w マップ・グループ *mapGroupName*、マップ *mapName* - 名前なし変数フィールドが、位置 (*row,column*) から除去されました。

説明: VisualAge Generator は、名前なし変数フィールドがマップに存在することを許容します。プログラムは、これらの名前なし変数フィールドにアクセスできません。テストおよび生成時に、名前なし変数フィールドは定数に変換されます。すべてのプロパティが定数フィールドのデフォルト値を指定しているので、マイグレーション・ツールはこの名前なし変数フィールドを除去しました。EGL の場合、デフォルトのプロパティを使用する定数を書式に対して明示的に定義する必要はありません。

ユーザーの処置: 書式定義を検討し、このフィールドを除去することが正しいマイグレーションであることを確認してください。その他の考慮事項については、103 ページの『名前なしマップ変数フィールド』にある、名前なし変数フィールドに関する説明を参照してください。

IWN.MIG.0506.e マップ・グループ *mapGroupName*、マップ *mapName* - 行、列に無保護定数があります。**protect=skipProtect** に変更しました。

説明: VisualAge Generator の場合、表示マップとプリンター・マップの両方に無保護定数を使用できます。定数の場合、EGL は、**protect** プロパティが **skipProtect** または **protect** のいずれかに設定されていることを必要とします。マイグレーション・ツールは、**protect** プロパティをフィールドの **skipProtect** に設定します。

ユーザーの処置: **skipProtect** が受け入れ可能であれば、アクションは必要ありません。定数フィールドの **protect** プロパティを **skipProtect** に設定すると、ユーザーは直前の変数フィールドの末尾から入力を継続でき、追加された文字は次の無保護変数フィールドに配置されます。定数フィールドの **protect** プロパティを **protect** に設定すると、ユーザーは直前の変数フィールドの末尾から入力を継続することができなくなります。ユーザーが入力を継続するには、次の変数フィールドまでタブ・キーで移動する必要があります。

IWN.MIG.0507.w マップ・グループ *mapGroupName*、マップ *mapName* - **row=0, column=0** にある定数が、**row=1, column=1** に変更されました。

説明: VisualAge Generator は、マップ上で位置 **row=0, column=0** にある定数を許容します (ただし完全にはサポートしません)。**row=0, column=0** にあるフィールドは、属性情報を指定できません。EGL は、**row=0, column=0** にあるフィールドをサポートしません。**row=0, column=0** にあるフィールドは定数で、第 1 バイトはブランクに初期化されます。マイグレーション・ツールは、位置を **row=1, column=1** に変更し、定数値の第 1 バイトを削除します。マイグレーション・ツールは、フィールドの **color** や **highlight** などのフィールド表示プロパティを組み込みません。これは、この情報が外部ソース形式ファイルに記録されていないからです。

ユーザーの処置: この書式を使用するプログラムをテストして、表示の外観に変化があるかどうか判断してください。変化がある場合は、書式を編集して、望ましい外観が得られるようにフィールド表示プロパティを設定します。

IWN.MIG.0508.e マップ・グループ *mapGroupName*、マップ *mapName* - **row=0, column=0** にある定数を変更できません。

説明: VisualAge Generator は、マップ上で位置 **row=0, column=0** にある定数を許容します (ただし完全にはサポートしません)。**row=0, column=0** にあるフィールドは、属性情報を指定できません。EGL は、**row=0, column=0** にあるフィールドをサポートしません。**row=0, column=0** にあるフィールドは定数で、第 1 バイトはブランクに初期化されません。フィールドの位置を変更すると、定数が移動したり書式から除去されたりして外観が変化するので、マイグレーション・ツールはフィールドの位置を変更しません。マイグレーション・ツールは、フィールドの **color** や **highlight** などのフィールド表示プロパティを組み込みません。これは、この情報が外部ソース形式ファイルに記録されてい

ないからです。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。

ユーザーの処置: 書式を編集し、定数フィールドを変更して、フィールドを row=1, column = 1 に配置します。属性バイトが row=1, column=1 を占めるようになるので、必要に応じて、その補正のために定数フィールドを変更して 1 バイトを除去します。必ず、この書式を使用するプログラムをテストして、表示の外観に変化があるかどうか判別してください。変化がある場合は、書式を編集して、望ましい外観が得られるようにフィールド表示プロパティを設定します。

IWN.MIG.0509.e マップ・グループ *mapGroupName*、マップ *mapName* - row=0, column=0 にある変数を変更できません。

説明: このマップは、システム共通プロダクトまたは VisualAge Generator の古いバージョンで作成された可能性があります。VisualAge Generator 4.5 は、row=0, column=0 にある変数をサポートしません。row=0, column=0 にあるフィールドは、属性情報を指定できません。EGL は、row=0, column=0 にあるフィールドをサポートしません。row=0, column=0 にあるフィールドは、変数フィールドです。フィールドの位置を変更すると、フィールドが移動したり、データの第 1 バイトが失われたりするので、マイグレーション・ツールはフィールドの位置を変更しません。マイグレーション・ツールは、フィールドの **color** や **highlight** などの表示プロパティを組み込みません。これは、この情報が外部ソース形式ファイルに記録されていないからです。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。

ユーザーの処置: 書式を編集し、フィールドを変更して、フィールドを row=1, column = 1 に配置します。必要に応じて、row=1, column=1 に移動した属性バイトが原因でデータが失われないように、変数フィールドの近くにある他のフィールドを変更してください。必ず、この書式を使用するプログラムをテストして、表示の外観に変化があるかどうか判別してください。変化がある場合は、書式を編集して、望ましい外観が得られるようにフィールド表示プロパティを設定します。

IWN.MIG.0510.e マップ・グループ *mapGroupName*、マップ *mapName* - *mapName* がプログラム名と競合します。

説明: プログラムが、そのプログラムと同名のマップを含むマップ・グループまたはヘルプ・マップ・グループを使用しています。VisualAge Generator は、マップ名がプログラム名と同じであることを許容します。EGL は、書式名がプログラム名と同じであることを許容しません。マップ名がプログラム名と同じであり、マップに

変数フィールドがない場合、マイグレーション・ツールはプログラムのヘルプ・マップ・グループ内のマップ名を変更します。ただし、次の場合、マイグレーション・ツールはマップの名前を変更しません。

- マップが、プログラムのヘルプ・マップ・グループ内にある変数フィールドを含むマップである。
- マップがプログラムのメイン・マップ・グループにある。

ユーザーの処置: FormGroup を編集し、書式の名前を変更します。また、この FormGroup を使用するすべてのプログラム内で、書式定義とこの書式への参照すべてを必ず変更します。その他の考慮事項については、97 ページの『マップ名とヘルプ・マップ名』にある、マップ名に関する説明を参照してください。

IWN.MIG.0512.e マップ・グループ *mapGroupName*、マップ *mapName* - 重複したカーソルがフィールド *fieldName* から除去されました。

説明: VisualAge Generator Templates (VAGT) のカスタマイズによっては、マップ上に複数のカーソルが指定されます。この場合には、VisualAge Generator は重複したカーソルを容認します。テスト機能と生成の場合、VisualAge Generator は、カーソルを指定する最初のフィールドにカーソルを配置し、他のすべてのフィールドを無視します。この場合の最初のフィールドは、行および列の順による最初のフィールドであり、編集順による最初のフィールドではありません。EGL は、重複したカーソルを容認しません。マイグレーション・ツールは、カーソルを指定する最初のフィールド以外のすべてのフィールドから、**cursor** プロパティを除去します。配列要素上に複数のカーソルが指定されている場合、マイグレーション・ツールは、カーソルを指定する最初の要素以外の配列のすべての要素から、**cursor** プロパティを除去します。

ユーザーの処置: カーソルを指定する最初のフィールド上のカーソル位置が受け入れ可能である場合、アクションは必要ありません。

IWN.MIG.0601.w 関数 *functionName*、入出力オブジェクト *recordName* - **UPDATE** オプションのレコード・タイプを判別できません。非 SQL レコードが想定されます。

説明: SQL の場合、プログラム内に複数の UPDATE 関数または SETUPD 関数があれば、VisualAge Generator の REPLACE 関数に、対応する UPDATE 関数または SETUPD 関数の名前を指定する必要があります。EGL は、SQL ステートメントに **resultSetID** を使用して、**replace** ステートメントと、それに対応する

get ステートメントまたは **open** ステートメントの関係を指定します。*recordName* によって示されたレコードが、マイグレーション時に使用不可でした。マイグレーション・ツールは、UPDATE 関数が非 SQL レコードを対象にしていると想定し、*resultSetID* を組み込みません。

ユーザーの処置: プログラム内の同じレコードに対して複数の **get** ステートメントまたは **open** ステートメントが存在するために、EGL 検証でエラー・メッセージが表示される場合は、関数を編集して、**get forUpdate** ステートメントに *resultSetID* を追加します。*resultSetID* は、プログラム内で固有であることが必要です。*resultSetID* を固有にするためには、マイグレーション時に *resultSetID* として使用した「結果セット接尾部」の設定値を、関数名の後ろに付けて使用します。その他の考慮事項については、126 ページの『複数の UPDATE 関数または SETUPD 関数を使用する SQL 入出力』を参照してください。

IWN.MIG.0602.w 関数 *functionName* - 入出力オブジェクト *mapName*; のマップ・タイプを判別できません。表示マップが想定されます。

説明: VisualAge Generator は、表示マップとプリンター・マップの両方に DISPLAY 入出力オプションを使用します。EGL は、**display** ステートメントをテキスト書式のみで使用し、**print statement** ステートメントを印刷書式に使用します。VisualAge Generator 互換モードでは、**display** ステートメントを印刷書式にも使用することができます。*mapName* として指定されたマップが、マイグレーション時に使用不可でした。マイグレーション・ツールは、マップが表示マップであると想定し、EGL の **display** ステートメントにマイグレーションします。

ユーザーの処置: VisualAge Generator 互換モードを使用し続ける場合、またはマップが表示マップである場合には、アクションは必要ありません。マップが印刷マップであり、VisualAge Generator 互換モードの使用を中止する場合は、**print** ステートメントを使用するように関数を変更する必要があります。その他の考慮事項については、113 ページの『マップの DISPLAY 入出力オプション』を参照してください。

IWN.MIG.0603.e 関数 *functionName*、SQL 入出力オブジェクト *recordName* - SQL 表名を判別できません。

説明: VisualAge Generator は、テストおよび生成時に SQL レコードから SQL 表名を判別します。EGL の場合は、変更されたすべての SQL ステートメントに表名が組み込まれている必要があります。*recordName* によって示されたレコードが、マイグレーション時に使用不可

可でした。マイグレーション・ツールは、EGL 検証でエラー・メッセージが表示されるように、表名に EZE_UNKNOWN_SQLTABLE を使用します。さらにマイグレーション・ツールは、ステートメントの表ラベルを T1 に設定します。

ユーザーの処置: 関数を編集し、レコード定義に基づいて正しい表名と表ラベルを指定します。表名は、EGL レコード定義の *tableNames* および *tableNameVariables* のどちらか、または両方のプロパティにあります。その他の考慮事項については、509 ページの『付録 D. 「問題」ビューのメッセージ』にある EZE_UNKNOWN_SQLTABLE に関する説明を参照してください。

IWN.MIG.0604.e 関数 *functionName*、SQL 入出力オブジェクト *recordName* - *!itemColumnName* 変数の列名を判別できません。

説明: 変更された SQL ステートメントが、1 つ以上の VAGen *!itemColumnName* 変数を使用していました。VAGen *!itemColumnName* 変数は、実際の SQL 列名に対応する SQL レコード定義内の項目名を指定します。VisualAge Generator は、テストおよび生成時に、*!itemColumnName* 変数の実際の SQL 列名を SQL レコードから判別します。EGL は、*!itemColumnName* 変数をサポートしません。代わりに、EGL の場合は、変更された SQL ステートメントの中で実際の SQL 列名を使用する必要があります。*recordName* によって示されたレコードまたは代替仕様のレコードが、マイグレーション時に使用不可でした。マイグレーション・ツールは、変更された SQL ステートメント内で *!itemColumnNames* を使用して、可能なかぎり多くの情報を提供します。

ユーザーの処置: 関数を編集し、レコード定義に基づいて SQL 列名を指定します。それぞれの *!itemColumnName* ごとに、対応する項目を SQL レコード定義の中で見つけます。その項目の列名が、EGL の入出力ステートメント内で使用する必要がある列名です。その他の考慮事項については、125 ページの『SQL 入出力と *!itemColumnName*』を参照してください。

IWN.MIG.0605.w 関数 *functionName*、SQL 入出力オブジェクト *recordName* - SQLEXEC は *model=none* を指定し、SQL 文節を指定していません。

説明: SQLEXEC 入出力オプションは、モデル型として *none* を指定していますが、SQL 文節を含んでいません。VisualAge Generator で入出力オブジェクトが指定されていない場合、*recordName* は省略されます。実際には、VisualAge Generator はこの入出力オプションに関してノーオペレーションを生成します。マイグレーション

ン・ツールは、EGL のノーオペレーション・ステートメント (セミコロンのみ) を生成し、モデル型が `none` であったことを示す VAGen の情報コメントを組み込みます。VAGen 関数がエラー・ルーチンを指定している場合、マイグレーション・ツールは、そのエラー・ルーチンに適した `try...onException` ブロックを組み込みます。

ユーザーの処置: 関数を検討して、入出力ステートメントを除去するか、拡張するかを判断してください。

IWN.MIG.0607.e 関数 *functionName*、SQL 入出力オブジェクト *recordName* - SQL 入出力文節 *clauseName* を判別できません。

説明: 指定された *recordName* が、マイグレーション時に使用不可でした。マイグレーション・ツールは、以下のいずれかの状況では、指定された文節を作成できません。

- 関数が、変更された SQL を使用しているものの、一部の文節が欠落している。ある時期の VisualAge Generator では、変更された SQL 文節のみが関数とともに保管されていました。この場合、残りの文節は、関数の入出力オブジェクトとして指定されたレコード定義から VisualAge Generator によって作成されます。この状況でリストされる可能性がある *clauseNames* は、
SELECT、INTO、INSERTCOLNAME、VALUES、および FORUPDATEOF です。
- 関数がデフォルトの SQL を使用し、「**Execution time statement build**」オプションが指定されている。この場合、すべての文節は、関数の入出力オブジェクトとして指定されたレコード定義から VisualAge Generator によって作成されます。この状況でリストされる可能性がある *clauseNames* は、SELECT、INTO、INSERTCOLNAME、VALUES、FORUPDATEOF、SET、WHERE、および ORDERBY です。

マイグレーション・ツールは、スケルトン文節を作成し、EZE_UNKNOWN_SQL_clauseName を組み込みます。

ユーザーの処置: メッセージに示されているレコードを見つめます。関数を編集して、欠落している SQL 文節を組み込みます。欠落している SQL 文節をどのように指定する必要があるか判別するには、VAGen SQL ステートメント・エディターを使用して SQL 文節を表示します。詳細と起こりうる問題については、119 ページの『SQL 入出力と必須 SQL 文節の欠落』または 123 ページの『SQL 入出力と Execution time statement build』を参照してください。509 ページの『付録 D. 「問題」ビューのメッセージ』にある

EZE_UNKNOWN_SQL_clauseName に関する情報を参照してください。

IWN.MIG.0608.e 関数 *functionName*、SQL 入出力オブジェクト *recordName* - 代替仕様 *altspecRecordName* の SQL 入出力文節 *clauseName* を判別できません。

説明: 指定された *recordName* は、マイグレーション時に使用可能です。ただし、*recordName* は、マイグレーション時に使用できない代替仕様レコード

altspecRecordName を指定しています。マイグレーション・ツールは、以下のいずれかの状況では、指定された文節を作成できません。

- 関数が、変更された SQL を使用しているものの、一部の文節が欠落している。ある時期の VisualAge Generator では、変更された SQL 文節のみが関数とともに保管されていました。この場合、残りの文節は、関数の入出力オブジェクトとして指定されたレコード定義から VisualAge Generator によって作成されます。この状況でリストされる可能性がある *clauseNames* は、
SELECT、INTO、INSERTCOLNAME、VALUES、および FORUPDATEOF です。
- 関数がデフォルトの SQL を使用し、「**Execution time statement build**」オプションが指定されている。この場合、すべての文節は、関数の入出力オブジェクトとして指定されたレコード定義から VisualAge Generator によって作成されます。この状況でリストされる可能性がある *clauseNames* は、SELECT、INTO、INSERTCOLNAME、VALUES、FORUPDATEOF、SET、WHERE、および ORDERBY です。

マイグレーション・ツールは、スケルトン文節を作成し、EZE_UNKNOWN_SQL_clauseName を組み込みます。

ユーザーの処置: メッセージに示されている代替仕様レコードを見つめます。関数を編集して、欠落している SQL 文節を組み込みます。欠落している SQL 文節をどのように指定する必要があるか判別するには、VAGen SQL ステートメント・エディターを使用して SQL 文節を表示します。詳細と起こりうる問題については、119 ページの『SQL 入出力と必須 SQL 文節の欠落』または 123 ページの『SQL 入出力と Execution time statement build』を参照してください。509 ページの『付録 D. 「問題」ビューのメッセージ』にある
EZE_UNKNOWN_SQL_clauseName に関する情報を参照してください。

IWN.MIG.0609.e 関数 *functionName* - SSA 内のレコード *recordName* は使用できません。比較値項目 *itemName* の修飾を判別できません。

説明: 変更された DL/I ステートメントは非修飾の比較値項目を使用していました。デフォルトで、VisualAge Generator は、最初に現行の SSA と関連する DL/I セグメント・レコードの項目を検索します。現行の SSA に関連する指定された DL/I セグメント・レコード *recordName* は利用できません。このため、マイグレーション・ツールは、比較値項目の修飾を判別できませんでした。

ユーザーの処置: メッセージに示されているレコードを見つけ、検討します。この項目がレコード内にある場合、関数を編集し、欠落した比較値項目の修飾を組み込むようにします。項目がレコード内にない場合、使用する正しい修飾を判別するためにプログラム・ロジックを検討します。さらに、プログラムを生成した最後の時刻から、生成済み COBOL のソース・コードを検討できます。VisualAge Generator では、ある時点で比較値項目の修飾規則が変わりました。したがって、比較値項目の修飾が変動しているため、プログラムを最後に生成したとき以来リリースが変更されていないことが明らかでない限り、VisualAge Generator の現行リリースを使用してプログラムを再生成しないでください。

IWN.MIG.0610.e 関数 *functionName* - SSA のレコード *recordName* には、使用できない代替仕様レコード *altspecRecordName* があります。比較値項目 *itemName* の修飾を判別できません。

説明: 変更された DL/I ステートメントは非修飾の比較値項目を使用していました。デフォルトで、VisualAge Generator は、最初に現行の SSA と関連する DL/I セグメント・レコードの項目を検索します。現行の SSA に関連する指定された DL/I セグメント・レコード *recordName* は、マイグレーション中に使用できます。ただし、*recordName* は、マイグレーション時に使用できない代替仕様レコード *altspecRecordName* を指定しています。このため、マイグレーション・ツールは、比較値項目の修飾を判別できませんでした。

ユーザーの処置: メッセージに示されているレコードを見つけ、検討します。項目が代替仕様レコード内にある場合、関数を編集し、欠落した比較値項目の修飾を組み込むようにします。項目がレコード内にない場合、使用する正しい修飾を判別するためにプログラム・ロジックを検討します。さらに、プログラムを生成した最後の時刻から、生成済み COBOL のソース・コードを検討できます。VisualAge Generator では、ある時点で比較値

項目の修飾規則が変わりました。したがって、比較値項目の修飾が変動しているため、プログラムを最後に生成したとき以来リリースが変更されていないことが明らかでない限り、VisualAge Generator の現行リリースを使用してプログラムを再生成しないでください。

IWN.MIG.0611.e 関数 *functionName* - 比較値項目 *itemName* が、レコード *recordName* にあります。修飾を判別できません。

説明: 変更された DL/I ステートメントは非修飾の比較値項目を使用していました。デフォルトで、VisualAge Generator は、最初に現行の SSA と関連する DL/I セグメント・レコードの項目を検索します。マイグレーション・ツールは、現行の SSA に関連した DL/I セグメント・レコードを検索しましたが、項目を検出できませんでした。VisualAge Generator では、ある時点で比較値項目の修飾規則が変わりました。マイグレーション・ツールは、比較を修飾するためにどのレコードを使用すべきかを決定できません。

ユーザーの処置: 使用する正しい修飾を判別するためにプログラム・ロジックを検討します。さらに、プログラムを生成した最後の時刻から、生成済み COBOL のソース・コードを検討できます。比較値項目の修飾が変動しているため、プログラムを最後に生成したとき以来リリースが変更されていないことが明らかでない限り、VisualAge Generator の現行リリースを使用してプログラムを再生成しないでください。

IWN.MIG.0612.e 関数 *functionName* - SSA に関して無効な関係演算子です。正しい演算子を判別できません。

説明: 変更済み DL/I は、無効な関係演算子を使用していました。これは、関係演算子に適していない値が保管される原因となった、VisualAge Generator 内の問題によって発生している可能性があります。マイグレーション・ツールは正しい関係演算子を判別できません。マイグレーション・ツールは、関係演算子として EZE_UNKNOWN_RELOP を使用します。

注: 問題を引き起こす可能性の最も高い演算子は、不等号に使用される記号です。EGL SSA における不等号の記号は != です。

ユーザーの処置: VisualAge Generator で DL/I 呼び出しエディターを使用し、指定済み関数に関する SSA を検討します。演算子が関数の外部形式ファイルに誤って保管されていても、正しい演算子が DL/I Call Editor に表示されます。EGL で関数を編集し、EZE_UNKNOWN_RELOP を正しい値に変更します。

IWN.MIG.0613.w 関数 *functionName* - Execution Time Statement Build を使用する際に、入出力エラー・ルーチンがありません。

説明: VisualAge Generator では、指定された関数は、INQUIRY、UPDATE、SETINQ、または SETUPD 入出力オプションが指定されている場合に「**Execution Time Statement Build**」オプションを使用しますが、入出力エラー・ルーチンが指定されていません。この結果、生成時に SQL PREPARE ステートメントの後ろに OPEN が付けられ、続いて (INQUIRY または UPDATE の場合) 後ろに FETCH が付きます。SQL PREPARE ステートメントでソフト・エラーが発生した場合、処理は続行されます。マイグレーション・ツールは入出力オプションを EGL prepare ステートメントとそれに続く get (INQUIRY または UPDATE の場合) または open (SETINQ または SETUPD の場合) に変換します。しかし、VisualAge Generator 内に入出力エラー・ルーチンが存在しなかったため、マイグレーション・ツールは、**try...onException** ブロックを EGL 入出力ステートメントの周辺に組み込むことができず、したがって、EGL prepare ステートメントでソフト・エラーが発生した後も処理を続行するための追加のロジックを組み込むことができません。マイグレーションの結果として、EGL prepare ステートメントでソフト・エラーが発生した場合、処理は停止されます。

ユーザーの処置: prepare ステートメントでソフト・エラーが発生する可能性があり、処理を続行する必要がある場合、**try...onException** ブロックを EGL prepare ステートメントの周辺に配置し、適切なエラー処理ロジックを組み込んで、ソフト・エラーが発生した後も処理を続行できるようにします。

IWN.MIG.0614.w 関数 *functionName* - SQLEXEC は Execution Time Statement Build と一緒に使用します。

説明: VisualAge Generator では、指定された関数は、SQLEXEC 入出力オプションを使用し、「**Execution Time Statement Build**」オプションも指定します。この結果、生成時に SQL EXECUTE IMMEDIATE が得られ、これにより、PREPARE、EXECUTE、および DESTROY がこの単一 SQL ステートメントによって実行されます。マイグレーション・ツールは SQLEXEC を EGL prepare ステートメントとそれに続く EGL execute ステートメントに変換します。これが最も近い EGL になります。

ユーザーの処置: なし。ただし、この関数を使用するすべてのプログラムをテストし、動作とパフォーマンスが VisualAge Generator と同一であることを確認してください。

IWN.MIG.0701.e 関数 *functionName* - SET map PAGE ステートメントで使用されている *mapName* のマップ・タイプを判別できません。**converseLib.EZE_SETPAGE();** が使用されました。

説明: VisualAge Generator は、SET map PAGE を使用して表示マップの場合に画面を消去することや、印刷マップの場合にページ替えを行うことを示します。EGL は、**converseLib.clearScreen()** 関数をテキスト書式のみに対して使用し、**converseLib.pageEject()** 関数を印刷書式に対して使用します。*mapName* として指定されたマップが、マイグレーション時に使用不可でした。マイグレーション・ツールは、マップ・タイプに関する推測は行いません。代わりにマイグレーション・ツールは、**converseLib.EZE_SETPAGE()** プレースホルダーを使用して、EGL 検証でエラー・メッセージが表示されるようにします。マイグレーション・ツールは、オリジナルのマップ名をコメントとして組み込みます。

ユーザーの処置: 関数を確認し、**converseLib.clearScreen()** または **converseLib.pageEject()** が正しい選択であるかどうか判別します。その他の考慮事項については、133 ページの『SET map PAGE ステートメント』を参照してください。

IWN.MIG.0702.e 関数 *functionName* - テーブル *tableName* が欠落しているために、RETR ステートメントの戻す列名を判別できません。

説明: 戻す列が RETR ステートメントに指定されていない場合は、指定されたテーブルの 2 列目に基づいて、VisualAge Generator が戻す列名を自動的に判別します。RETR に対する EGL の置換表現は、if ステートメントと、その後に続く代入ステートメントです。戻す列名は、代入ステートメントの中で明示的に指定されている必要があります。*tableName* によって示されたテーブルが、マイグレーション時に使用不可でした。マイグレーション・ツールは、EGL 検証でエラー・メッセージが表示されるように、**EZE_UNKNOWN_RETURN_COLUMN** を使用します。この問題がプログラムのフロー・ステートメント内で発生した場合は、関数名の代わりにプログラム名がメッセージに出力されます。

ユーザーの処置: 関数を編集し、戻す列をテーブル定義に基づいて正しく指定します。テーブルの 2 列目が、VisualAge Generator 内で使用されるデフォルトの戻す列です。その他の考慮事項については、132 ページの『RETR ステートメント』を参照してください。

IWN.MIG.0703.e 関数 *functionName* - テーブル

tableName が欠落しているために、RETR ステートメントの検索列名を判別できません。

説明: 検索列が RETR ステートメントに指定されていない場合は、指定されたテーブルの 1 列目に基づいて、VisualAge Generator が検索列名を自動的に判別します。RETR に対する EGL の置換表現は、**if** ステートメントと、その後に続く代入ステートメントです。検索列名は、**if** ステートメントの中で明示的に指定されている必要があります。*tableName* によって示されたテーブルが、マイグレーション時に使用不可でした。マイグレーション・ツールは、EGL 検証でエラー・メッセージが表示されるように、EZE_UNKNOWN_SEARCH_COLUMN を使用します。この問題がプログラムのフロー・ステートメント内で発生した場合は、関数名の代わりにプログラム名がメッセージに出力されます。

ユーザーの処置: 関数を編集し、検索列をテーブル定義に基づいて正しく指定します。テーブルの 1 列目が、VisualAge Generator 内で使用されるデフォルトの検索列です。その他の考慮事項については、132 ページの『RETR ステートメント』を参照してください。

IWN.MIG.0704.e 関数 *functionName* - テーブル

tableName が欠落しているために、FIND ステートメントの検索列名を判別できません。

説明: 検索列が FIND ステートメントに指定されていない場合は、指定されたテーブルの 1 列目に基づいて、VisualAge Generator が検索列名を自動的に判別します。FIND に対する EGL の置換表現は、**if** ステートメントと、その後に続く関数呼び出しステートメントです。検索列名は、**if** ステートメントの中で明示的に指定されている必要があります。*tableName* によって示されたテーブルが、マイグレーション時に使用不可でした。マイグレーション・ツールは、EGL 検証でエラー・メッセージが表示されるように、EZE_UNKNOWN_SEARCH_COLUMN を使用します。この問題がプログラムのフロー・ステートメント内で発生した場合は、関数名の代わりにプログラム名がメッセージに出力されます。

ユーザーの処置: 関数を編集し、検索列をテーブル定義に基づいて正しく指定します。テーブルの 1 列目が、VisualAge Generator 内で使用されるデフォルトの検索列です。その他の考慮事項については、131 ページの『FIND ステートメント』を参照してください。

IWN.MIG.0706.e 関数 *functionName* - IF、WHILE、または TEST DUP ステートメント内で使用されている *recordName* のレコード・タイプを判別できません。 EZE_DUPLICATE を使用しました。

説明: VisualAge Generator は、非 SQL レコードと SQL レコードの両方に対して、DUP と UNQ の両方の検査をサポートします。SQL レコードの場合、DUP と UNQ は同一です。非 SQL レコードの場合、EGL は、**duplicate** と **unique** の両方をサポートします。SQL レコードの場合、EGL は **unique** のみをサポートします。*recordName* によって示されたレコードが、マイグレーション時に使用できませんでした。マイグレーション・ツールは、EGL 検証でエラー・メッセージが表示されるように、DUP を EZE_DUPLICATE にマイグレーションします。

注: マイグレーション・ツールは、TEST ステートメントを **if** ステートメントにマイグレーションします。

ユーザーの処置: 関数を編集し、EZE_DUPLICATE を次のいずれかの値に変更します。

- SQL レコードの場合は **unique**
- 非 SQL レコードの場合は **duplicate**

その他の考慮事項については、138 ページの『入出力エラー値 UNQ および DUP』を参照してください。

IWN.MIG.0707.e 関数 *functionName* - IF、WHILE、または TEST NULL ステートメント内で項目 *itemName* が使用されるときに、この項目がレコードまたはマップのどちらにあるか判別できません。EZE_NULL を使用しました。

説明: VisualAge Generator は、マップ項目と SQL 項目の両方に対して、NULL の検査をサポートします。マップ項目の NULL の検査は、ブランクの検査と同等です。SQL 項目の NULL の検査を行うと、NULL 標識変数が検査され、データベース内で列が NULL かどうか判別されます。同等な EGL ステートメントにより、書式フィールドが **blanks** かどうか、および SQL フィールドが **null** かどうか検査されます。*itemName* に示された項目が、マイグレーション時に使用不可でした。マイグレーション・ツールは、EGL 検証でエラー・メッセージが表示されるように、NULL を EZE_NULL にマイグレーションします。

注: マイグレーション・ツールは、TEST ステートメントを **if** ステートメントにマイグレーションします。

ユーザーの処置: 関数を編集し、EZE_NULL を次のいずれかの値に変更します。

- 書式フィールドの場合は **blanks**
- SQL フィールドの場合は **null**

その他の考慮事項については、136 ページの『SQL 項目とマップ項目が NULL かどうかの検査』を参照してください。

IWN.MIG.0708.w 関数 *functionName* - IF、WHILE、または TEST 以外のステートメント内で EZESYS を使用しています。以前の VAGen 値が使用されます。

説明: VisualAge Generator は、IF、WHILE、および TEST 以外のステートメント内で EZESYS の使用をサポートします。マイグレーション・ツールは、ステートメント・タイプに基づいて EZESYS をマイグレーションします。IF、WHILE、および TEST の各ステートメントの場合、マイグレーション・ツールは、EZESYS を **sysVar.systemType** に変換し、さらに値を新しい EGL 値に変換します。IF、WHILE、TEST 以外のステートメントの場合は、マイグレーション・ツールは **custPrefixEZESYS** に変換します。ここで **custPrefix** は、マイグレーション用に設定した「名前変更接頭部」設定です。プログラムをマイグレーションするとき、「以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)」マイグレーション設定を選択解除すると、マイグレーション・ツールは、**custPrefixEZESYS** の宣言、および **custPrefixEZESYS** をオリジナルの VAGen 値に初期化するステートメントを組み込みます。オリジナルの VAGen 値がこのステートメントで使用されます。

ユーザーの処置: 関数を検討し、オリジナルの VAGen 値を使用するか、新しい EGL 値を使用するかを判断してください。新しい EGL 値を使用する場合は、**custPrefixEZESYS** を **sysVar.systemType** に変更してください。

オリジナルの VAGen 値を使用する予定で、マイグレーション中に、「以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)」マイグレーション設定を選択した場合は、指定された関数を使用するすべてのプログラムに **custPrefixEZESYS** の宣言および初期化ステートメントを追加する必要があります。オリジナルの VAGen 値を使用する予定で、「以前の EZESYS 値を初期化しない (Do not initialize old EZESYS values)」マイグレーション設定を選択解除した場合は、変更は必要ありません。**custPrefixEZESYS** の宣言および初期化ステートメントは、既にすべてのマイグレーション済みプログラムに組み込まれています。

IWN.MIG.0710.e 関数 *functionName* - テーブル

***tableName* 用の MOVEA、しかし、テーブルには 1 行しか含まれていません。**

説明: MOVEA ステートメントは、テーブルを、ソース・フィールドの修飾子として指定しています。VisualAge Generator では、単一行のコンテンツを含んだテーブルが MOVEA ステートメントのソースとして使用される場合、そのソースはスカラーとして処理され、ターゲット配列はスカラー・ソースによって完全に初期化されます。これは、VAGen ドキュメンテーションに反します。このドキュメンテーションによると、テーブルは常に配列として処理される必要があり、その結果、ターゲット配列の最初の要素のみが初期化されることになると示されています。EGL では、**for** 修飾子の付いた **move** は常に、ある配列から別の配列への移動として処理され、したがって、ターゲット配列の最初の要素のみが初期化されます。この問題がプログラムのフロー・ステートメント内で発生した場合は、関数名の代わりにプログラム名がメッセージに出力されます。

ユーザーの処置: プログラム・ロジックを調べて、ターゲット配列全体の初期化を目的としていたことを確認します。そうである場合は、ループを使用してテーブルのソース要素からターゲット配列を初期化するように、プログラム・ロジックを変更します。

IWN.MIG.0711.w 関数 *functionName* - 修飾子

***tableName* 用の MOVEA、しかし、修飾子が使用できません。**

説明: MOVEA ステートメントは、ソース・フィールド用の修飾子を指定しています。VisualAge Generator では、単一行のコンテンツを含んだテーブルが MOVEA ステートメントのソースとして使用される場合、そのソースはスカラーとして処理され、ターゲット配列はスカラー・ソースによって完全に初期化されます。これは、VAGen ドキュメンテーションに反します。このドキュメンテーションによると、テーブルは常に配列として処理される必要があり、その結果、ターゲット配列の最初の要素のみが初期化されることになると示されています。EGL では、**for** 修飾子の付いた **move** は常に、ある配列から別の配列への移動として処理され、したがって、ターゲット配列の最初の要素のみが初期化されます。修飾子が使用できないため、マイグレーション・ツールは、そのテーブルが、単一行のコンテンツしか含んでいないテーブルなのかどうかを判別できません。この問題がプログラムのフロー・ステートメント内で発生した場合は、関数名の代わりにプログラム名がメッセージに出力されます。

ユーザーの処置: メッセージが、マップ・グループのマイグレーション中に発行された場合は、このメッセージ

を無視してもかまいません。メッセージが、プログラムのマイグレーション中に発行された場合は、指定された修飾子の定義を調べて、レコード、マップ、または複数行を含んだテーブルなのかどうかを判別します。そうである場合は、アクションは不要です。修飾子が、単一行のコンテンツしか入っていないテーブルである場合は、メッセージ IWN.MIG.0710.e のユーザー応答を参照して、問題の解決方法の詳細を調べてください。

IWN.MIG.0801.e プログラム名 *programName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、プログラムの名前を自動的に変更しません。

ユーザーの処置: プログラムの名前と、それに対するすべての参照 (**call**、**transfer**、および **show** の各ステートメントでの参照、およびリンクエッジ・オプションのパーツでの参照など) を変更する必要があります。また、このプログラムに対応するバインド制御パーツ、またはリンク・エディット・パーツの名前も変更します。生成されるプログラムの名前としてオリジナルのプログラム名を保持する必要がある場合は、**alias** プロパティを指定できます。**alias** プロパティを指定しない場合は、CICS プログラム定義とトランザクション定義など、EGL 以外によるプログラム名の参照すべてを必ず変更してください。

IWN.MIG.0802.w プログラム *programName* - 暗黙項目を許容しています。暗黙項目の定義はマイグレーションによって作成されません。

説明: VisualAge Generator のプログラムは、暗黙データ項目の許容を指定できます。暗黙データ項目を許容するプログラムが、未定義の項目を実際に使用すると、テストおよび生成時に VisualAge Generator が定義を自動的に作成します。EGL は、暗黙項目を許容しません。マイグレーション・ツールは、暗黙定義を自動的に作成しません。

ユーザーの処置: VisualAge Generator 内でプログラムを検証して、暗黙項目が使用されているかどうか判別してください。使用されている場合、VisualAge Generator は暗黙項目の定義を検証メッセージに示します。EGL でプログラム定義を編集し、対応する EGL プリミティブ・フィールド宣言を追加します。フィールドを格納するためのレコードを作成する必要はありません。プリミティブ・フィールド宣言は、プログラムに直接追加できます。マイグレーションのステージ 1 を実行する前に暗黙項目を作成するために役立つホワイト・ペーパーについては、19 ページの『参照』を参照してください。

IWN.MIG.0804.w プログラム *programName* - CLOSE 入出力オプションに対して使用される入出力オブジェクト *partName* のパーツ型を判別できません。レコードが想定されます。

説明: VisualAge Generator では、テストまたは生成時に入出力オブジェクトが自動的に組み込まれます。

CLOSE 入出力オプションは、レコードと印刷マップの両方に使用できます。EGL では、入出力ステートメント内で使用されるレコードは、プログラム内で明示的に宣言されている必要があります。書式は明示的に宣言されませんが、FormGroup の **use** 宣言が必要です。指定されたプログラム内で CLOSE 入出力オプションが使用されており、指定された *partName* が CLOSE の入出力オブジェクトとして使用されています。ただし、指定された *partName* がマイグレーション時に使用不可でした。マイグレーション・ツールは、パーツがレコードであると想定し、データ宣言を組み込みます。

ユーザーの処置: マイグレーション・ツールの予測が誤っている場合は、EGL 検証でエラー・メッセージが「問題ビュー」に表示されます。プログラムを編集し、レコード宣言を除去し、書式に関する **use** 宣言を追加します。

IWN.MIG.0805.w プログラム *programName* - 実行モードが指定されていません。非セグメント化が想定されます。

説明: ある時期の VisualAge Generator では、実行モードがプログラム・パーツとともに保管されていませんでした。実行モードは、メインのトランザクション・プログラムのみに適用されます。指定された *programName* はメインのトランザクション・プログラムですが、実行モードが外部ソース形式に含まれていません。マイグレーション・ツールは、実行モードが非セグメント化であると想定し、EGL ソース内の **segmented** プロパティを NO に設定します。

ユーザーの処置: プログラムを非セグメント化モードで実行する場合は、アクションは必要ありません。プログラムをセグメント化モードで実行する必要がある場合は、プログラムを編集し、**segmented** プロパティを YES に変更します。

IWN.MIG.0806.w プログラム *programName*、テーブル *tableName* の **use 宣言 - 設定が原因で **deleteAfterUse=yes** が省略されました。**

説明: 指定されたプログラムに、指定されたテーブルの **use** 宣言が含まれています。システム共通プロダクトおよび VisualAge Generator のいくつかのリリースでは、**keep after use** フラグによって、テーブルのメモリーがプログラムによって解放される時期が決定されていま

す。VisualAge Generator の keep after use フラグは、通常、テーブルの use 宣言の EGL deleteAfterUse プロパティにマイグレーションされます。しかし、ご使用の VAGen マイグレーション設定は、マイグレーション・ツールでは、deleteAfterUse プロパティを組み込まないことが指定されています。

ユーザーの処置: なし。ご使用の VAGen プログラムが、VisualAge Generator 4.5 フィックスパック 4 以降を使用して生成されている場合は、振る舞いに差はありません。詳しくは、268 ページの『VisualAge Generator 互換モードの使用の中止』にある deleteAfterUse に関する情報を参照してください。

IWN.MIG.0807.e プログラム *programName* - PSB

psbName は使用できません。PSB の DLI セグメント・レコードが判別できません。

説明: VisualAge Generator では、プログラムの PSB で指定された DLI セグメント・レコードはテスト時、または生成時にすべて自動的に組み込まれます。これは、DLI セグメント・レコードがデフォルトの SSA を作成する時に使用されるか、または変更された SSA で明示的に使用される可能性があるからです。EGL では、DLI セグメント・レコードは、デフォルトの SSA 使用されるか、または変更された SSA で明示的に使用される場合、プログラムで明示的に宣言される必要があります。指定された *psbName* は、マイグレーション時に使用できません。従って、マイグレーション・ツールは、なんらかの DLI セグメント・レコードがプログラムのデータ宣言のリストに追加される必要があるかどうかを判別することができません。

ユーザーの処置: この関数またはプログラムの未確定または未解決の DLI セグメント・レコードに関するメッセージがないかどうか、EGL「問題」リストを調べます。メッセージがない場合、DLI セグメント・レコードのための宣言が既にプログラムに (最も高い可能性として DLI セグメント・レコードに対する I/O ディレクトリの結果として) 組み込まれています。メッセージがある場合、プログラムを編集して、DLI セグメント・レコードのための宣言を追加します。

IWN.MIG.0808.e 呼び出し先プログラム *programName*

- PSB *psbName* は使用できません。PCB タイプが判別できません。

説明: VisualAge Generator では、パラメーターとしてプログラムに引き渡される PCB を示すために、EZEDLPCB[n] (*n* は数値リテラル) が使用されます。PCB が I/O、代替、データベース、または GSAM PCB であるかどうかに関係なく、VisualAge Generator は PCB を使用します。EZEDLPCB[0] は常に I/O PCB であり、VAGen PSB パーツには明示的にリストされま

せん。PCB が実行時に使用されるときに、DLI PCB が正しいタイプでない場合にはエラーが発生します。EGL では、プログラムの PSBRecord からの PCB 名がパラメーター名として使用されます。PCB タイプは、レコード名 (IO_PCBRecord、

ALT_PCBRecord、DB_PCBRecord、または GSAM_PCBRecord) を使用して適切な型定義を与えて、各プログラム・パラメーターごとに明示的に指定しなければなりません。マイグレーション・ツールは EZEDLPCB[0] の型定義として常に IO_PCBRecord を使用します。プログラム *programName* は、*n* が 0 より大きい 1 つまたは複数の EZEDLPCB[n] パラメーターを指定します。ただし、指定された *psbName* は、マイグレーション時に使用できません。したがって、マイグレーション・ツールはパラメーター・リストの PCB 用に組み込むタイプを判別することができません。マイグレーション・ツールは、パラメーター・リストのすべての PCB 用に EZE_UNKNOWN_PCB_TYPE を使用します。

ユーザーの処置: 指定された PSBRecord を見つけます。プログラムを編集して型定義を変更し、指定された EGL PSBRecord で、対応する PCB タイプに基づいて正しい xxxx_PCBRecord を指定します。

IWN.MIG.0809.e 呼び出し先プログラム *programName*

- PSB *psbName* は使用できません。PCB マッピングが判別できません。

説明: VisualAge Generator では、パラメーターとしてプログラムに引き渡される PCB を示すために、EZEDLPCB[n] (*n* は数値リテラル) が使用されます。VisualAge Generator は、VAGen PSB パーツの対応する PCB に EZEDLPCB[n] を自動的に関連付けます。EZEDLPCB[0] は常に I/O PCB であり、VAGen PSB パーツには明示的にリストされません。DLI PSB が実行時に予想通りの数の PCB を持っていないと、エラーが発生します。EGL では、プログラムの PSBRecord からの PCB 名がパラメーター名として使用されます。EGL PSBRecord 内の対応する位置とパラメーター・リスト内の各 PCB を明示的に関連させるために、**pcbParms** プロパティが使用されます。プログラム *programName* は、1 つまたは複数の EZEDLPCB[n] パラメーターを指定しますが、指定された *psbName* はマイグレーション中は使用できません。したがって、マイグレーション・ツールは **pcbParms** プロパティに組み込む PCB の数を判別できません。マイグレーション・ツールは、**pcbParms** プロパティの値として EZE_UNKNOWN_PCB_MAPPING を使用します。

ユーザーの処置: 指定された PSBRecord を見つけます。プログラムを編集して、**pcbParms** プロパティを変更し、PCB パラメーターと、指定された EGL

PSBRecord の PCB との間のマッピングを提供します。

IWN.MIG.0810.e 呼び出し先プログラム *programName*
- パラメーター・リストが、**PSB**
psbName に存在する **PCB** 数値よりも大
きな数値を参照しています。**PCB** タイ
プ、および **PCB** マッピングが完了しませ
ん。

説明: VisualAge Generator では、パラメーターとして
プログラムに引き渡される **PCB** を示すために、
EZEDLPCB[n] (*n* は数値リテラル) が使用されます。
VisualAge Generator は、VAGen PSB パーツの対応する
PCB に EZEDLPCB[n] を自動的に関連付けます。関連
付けは、**PCB** が I/O、代替、データベース、または
GSAM **PCB** であるかどうかに関係なく行われます。
EZEDLPCB[0] は常に I/O **PCB** であり、VAGen PSB
パーツには明示的にリストされません。DL/I PSB が予
想通りの数の **PCB** を持っていないか、または DL/I
PCB が正しいタイプでない場合、ランタイム・エラー
が発生します。EGL では、プログラムの PSBRecord
からの **PCB** 名がパラメーター名として使用されます。
PCB タイプは、レコード名 (IO_PCBRecord、
ALT_PCBRecord、DB_PCBRecord、または
GSAM_PCBRecord) を使用して適切な型定義を与えて、
各プログラム・パラメーターごとに明示的に指定しなけ
ればなりません。マイグレーション・ツールは
EZEDLPCB[0] の型定義として常に IO_PCBRecord を使
用します。さらに、EGL PSBRecord 内の対応する位置
と、パラメーター・リストにある各 **PCB** を明示的に関
連させるために、**pcbParms** プロパティが使用されま
す。プログラム *programName* は、1 つまたは複数の
EZEDLPCB[n] パラメーターを指定していますが、*n* の
値のいくつかは、指定された *psbName* の **PCB** の数よ
りも大きくなっています。したがって、マイグレーシ
ョン・ツールはパラメーター・リスト内のいくつかの
PCB をインクルードする型定義を判別できません。マ
イグレーション・ツールは、指定された *psbName* にあ
る **PCB** に対応しないパラメーター・リスト内の **PCB**
については、EZE_UNKNOWN_PCB_TYPE を使用しま
す。さらに、マイグレーション・ツールは **pcbParms** プ
ロパティに組み込む **PCB** の数を判別できません。マ
イグレーション・ツールは、*n* の最も高い値まで、その
値を含めて、すべての EZEDLPCB[n] パラメーターにつ
いて **PCB** マッピング情報を作成します。しかし、この
リストは指定済み *psbName* の使用可能な **PCB** に一致
しません。

ユーザーの処置: 指定された PSBRecord を見つけま
す。PSBRecord とプログラム・ロジックを検討し、ど
ちらが正しいのかを判別します。追加の **PCB** を
PSBRecord に追加します。プログラムを編集してパラメ
ーターの型定義を変更し、指定された EGL PSBRecord

で、対応する **PCB** タイプに基づいて正しい
xxxx_PCBRecord を指定します。同様に、**pcbParms** プ
ロパティを変更し、**PCB** パラメーターと、指定され
た EGL PSBRecord 内の **PCB** との間の正しいマッピ
ングを提供します。

IWN.MIG.0811.w プログラム *programName* - マップ
を使用しないものと思われます。コメント
化された **FormGroup** *FormGroupName* の
ステートメントを使用します。

説明: VisualAge Generator では、メイン・トランザク
ション・プログラムまたは呼び出し先トランザクシ
ョン・プログラムは、マップを使用しない場合でも、常に
マップ・グループを指定する必要があります。こうした
状態では、マップ・グループ・パーツが存在する必要は
ありません。EGL では、プログラムは、実際に書式を
使用しなければ、**FormGroup** を指定する必要はありませ
ん。マイグレーション・ツールはプログラムがマップ・
グループを指定していることを判別しましたが、I/O オ
ブジェクト、マップ・ステートメントのある XFER で
の呼び出し先パラメーター、またはプログラムの最初の
マップとして、マップを使用するためには表示されませ
ん。したがって、マイグレーション・ツールは、EGL
プログラム内の **FormGroup** に関する **use** ステートメン
トをコメント化しました。また、ヘルプ **FormGroup** に
関する **use** ステートメントがある場合、マイグレーシ
ョン・ツールは、このステートメントもコメント化しま
した。

ユーザーの処置: なし。ただし、このプログラムに関す
る、マイグレーション・セットまたは外部ソース形式フ
ァイルにインクルードされていない追加関数がある場
合、これらの関数では指定されたマップ・グループから
のマップを使用する可能性があります。**FormGroup** 内の
書式を使用する必要がある場合、EGL プログラムを変
更し、**FormGroup** およびヘルプ **FormGroup** に関する
use ステートメントをアンコメントします。書式グル
ープ内に複数の書式が存在する場合は、**use** 宣言を変更し
て、プログラムが使用する **FormGroup** 内で特定の書式
が指定されるようにすることが必要になる場合があります。
特定の書式をリストすると、EGL 内での未解決ま
たは未確定の参照の回避に役立ちます。

IWN.MIG.0901.w **PSB** *psbName* に、同一のデータベー
ス名 *databaseName* を持った複数の **PCB**
が存在します。

説明: VisualAge Generator では、DL/I 入出力関数に関
して、データベース名か、または VAGen PSB からの
PCB 番号を指定することによって、どの **PCB** を使用す
るかを指定できます。VAGen PSB 内の複数の **PCB** に
ついて、同一のデータベース名を指定することができま

す。VAGen PSB 内に PCB 名はありません。実際のデータベース名は、テスト、生成、またはランタイムでは、決して使用されません。DL/I PSB には、PCBNAME パラメーターを組み込む必要はありません。EGL でのデバッグの場合、DL/I PSB 内のデータベース PCB ごとに PCBNAME パラメーター (または PCB 名を指定するラベル) が必要です。PCBNAME パラメーターは、EGL PSBRecord 内の対応する PCB の **pcbName** プロパティと一致する必要があります。マイグレーション・ツールは、VAGen PSB からのオリジナルのデータベース名に、ご使用の「データベース PCB 接尾部」マイグレーション設定を連結して、EGL PCB レコードの名前を作成します。マイグレーション・ツールはまた、EGL PCB レコードの **pcbName** プロパティを、VAGen PSB からのオリジナルのデータベース名に設定します。一般に、ご使用の DL/I PSB に PCBNAME を追加する場合、VAGen データベース名を PCBNAME として使用できます。ただし、PCBNAME は、DL/I PSB 内で固有でなければなりません。そのため、マイグレーション・ツールは、同一のデータベース名を持った複数の PCB が VAGen PSB 内に存在するときには常に、このメッセージを発行します。この場合、マイグレーション・ツールによって設定された **pcbName** プロパティを、すべての PCB で使用することはできません。

ユーザーの処置: ご使用の DL/I PSB を変更して PCBNAME パラメーターに組み込む場合は、固有の PCB 名を使用する必要があります。ご使用の DL/I PSB 内で PCB 名を設定した後、**pcbName** プロパティの EGL PSBRecord を更新して、ご使用の DL/I PSB 内の実際の PCB 名を反映させる必要があります。

IWN.MIG.1001.e 生成オプション・パーツ *partName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、プログラムの名前を自動的に変更しません。プログラムで *programName.opt* という名前の特殊な生成オプション・パーツが使用される可能性があるので、マイグレーション・ツールは生成オプション・パーツの名前も変更しません。

ユーザーの処置: プログラム名を変更する場合は、対応する生成オプション・パーツの名前を必ず変更してください。

IWN.MIG.1002.w 生成オプション・パーツ *partName* - **/dbms=odbc** は **dbms="DB2"** にマイグレーションされます。

説明: 指定された生成オプション・パーツは、VAGen 生成オプション **/dbms=odbc** を指定しています。EGL

は、DB2 または Oracle のみをサポートします。マイグレーション・ツールは、EGL ビルド記述子パーツ内で、**/dbms=odbc** を **dbms="DB2"** に変換します。EGL は、JDBC ドライバーを使用して DB2 のサポートを提供します。ご使用のデータベース用の JDBC ドライバーが存在すれば、ビルド記述子オプション **dbms="DB2"** をデータベース・タイプとして使用できる可能性があります。

注: EGL では、Oracle は、Java 生成を使用する場合にのみ、サポートされます。

ユーザーの処置: ODBC サポートを使用していたさまざまな VAGen プログラムをマイグレーション、生成、およびテストして、ご使用の JDBC ドライバーの環境で必要な機能すべてが正しく動作することを確認してください。詳しくは、277 ページの『SQL サポートに関する違い』を参照してください。

IWN.MIG.1003.e 生成オプション・パーツ *partName* - **/system=systemType** はサポートされていません。

説明: 指定された生成オプション・パーツは、VAGen の **/system** 生成オプションを含んでおり、EGL によってサポートされないランタイム環境を指定しています。マイグレーション・ツールは、EGL ビルド記述子パーツ内で、**/system** 生成オプションをコメントに変換します。

ユーザーの処置: このビルド記述子パーツが他のビルド記述子パーツによって使用されているかどうか判別してください。使用されていない場合は、このビルド記述子パーツを削除できます。また、EGL がこのランタイム環境を将来サポートする予定がある場合は、ビルド記述子パーツを参照用に保持しておくこともできます。

IWN.MIG.1004.w 生成オプション・パーツ *partName* - **/system=systemType** を使用するには、**destPort** が設定されている必要があります。

説明: 示された生成オプション・パーツは **/system** 生成オプションを含み、COBOL ランタイム環境を指定しています。EGL ビルド・プロセスを実行するには、**destPort** ビルド記述子オプションを使用して宛先ポートを指定する必要があります。

ユーザーの処置: 生成オプション・パーツに対応するビルド記述子パーツを変更して、**destPort** ビルド記述子オプションを組み込みます。**destPort** の値を指定するときは、以下の環境を考慮します。

- z/OS 環境の場合、**destPort** のデフォルト値はありません。**destPort** ビルド記述子オプションを追加する

必要があり、その値が z/OS ビルド・サーバーを開始する JCL で使用する値と一致しなければなりません。z/OS ビルド・サーバーを開始するサンプル JCL (ジョブ制御言語) はポート 5555 を使用します。

- iSeries 環境の場合、**destPort** のデフォルト値はありません。**destPort** ビルド記述子オプションを追加する必要があります。値は、iSeries ビルド・サーバーによって使用される値と一致しなければなりません。
- VSE 環境の場合、**destPort** のデフォルト値は 21 です。**destPort** ビルド記述子オプションを指定する必要があるのは、値が 21 と異なる場合のみです。

IWN.MIG.1099.e 制御パーツ *partName* - *symparm* *symparmName* はサポートされません。

説明: 指定された制御パーツは、EGL によってサポートされない *symparm* を使用または設定しています。マイグレーション・ツールは、オリジナルの VAGen *symparm* 名を使用して、*symparm* を「現状のまま」マイグレーションします。ただし、EGL では、**SymbolicParameter** は生成時に設定されません。

ユーザーの処置: 制御パーツを変更して、**SymbolicParameter** のデフォルト値を設定します。あるいは、指定された **SymbolicParameter** を今後は使用しないように制御パーツを変更します。

IWN.MIG.1101.e リンケージ・テーブル・パーツ *partName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、制御パーツの名前を自動的に変更しません。

ユーザーの処置: リンケージ・オプション・パーツ名を予約語でないものに変更します。リンケージ・オプション・パーツ名を変更する場合は、このリンケージ・オプション・パーツを参照するビルド記述子パーツすべてを必ず変更してください。

IWN.MIG.1102.e リンケージ・テーブル *partName* - */contable=BINARY* はサポートされません。変更する必要があります。

説明: VisualAge Generator は、リンケージ・テーブル・パーツ内で */contable=BINARY* をサポートします。EGL はこの値をサポートしません。マイグレーション・ツールは、EGL リンケージ・オプション・パーツ内の **conversionTable** プロパティを "BINARY" に設定します。この値は無効ですが、生成時まで検出されません。

ユーザーの処置: **conversionTable** プロパティの値を、EGL によってサポートされる値に変更する必要があります。

あります。EGL の **conversionTable** プロパティ、および使用可能なオプションについて詳しくは、オンライン・ヘルプのリンケージ・オプション・パーツに関する情報を参照してください。

IWN.MIG.1103.e リンケージ・テーブル・パーツ

partName -

/remoteComType=CICSECI はサポートされません。デフォルトで **CICSECI** に設定されます。

説明: VisualAge Generator は、リンケージ・テーブル・パーツ内で */remoteComType=CICSECI* をサポートします。EGL はこの値をサポートしません。マイグレーション・ツールは、EGL のリンケージ・オプション・パーツに **remoteComType="CICSECI"** を組み込みます。この値は有効ですが、使用する予定の値とは異なる可能性があります。CICSECI を使用する場合は、**ctgPort** および **ctgLocation** プロパティを設定する必要があります。

ユーザーの処置: CICSECI を使用する場合は、リンケージ・オプション・パーツを変更して、**remoteComType** として CICSECI を指定するエントリーの **ctgPort** と **ctgLocation** の値を設定します。CICSECI を使用しない場合、EGL の **remoteComType** プロパティ、および EGL で使用できるオプションについて詳しくは、オンライン・ヘルプのリンケージ・オプション・パーツに関する情報を参照してください。

IWN.MIG.1104.e リンケージ・テーブル・パーツ

partName -

/remoteComType=communicationType はサポートされません。変更する必要があります。

説明: VisualAge Generator は、リンケージ・テーブル・パーツ内で */remoteComType=communicationType* をサポートします。EGL は、この通信プロトコルをサポートしません。マイグレーション・ツールは、EGL のリンケージ・オプション・パーツで **remoteComType** プロパティを "*communicationType*" に設定します。この値は無効であり、変更する必要があります。

ユーザーの処置: 使用する通信プロトコルを決定します。その後、パーツを編集し、**remoteComType** を EGL によってサポートされる値に変更します。EGL の **remoteComType** プロパティ、および EGL で使用可能なオプションについて詳しくは、オンライン・ヘルプのリンケージ・オプション・パーツに関する情報を参照してください。

IWN.MIG.1201.e リソース関連パーツ *partName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、制御パーツの名前を自動的に変更しません。

ユーザーの処置: リソース関連パーツ名を予約語でないものに変更します。リソース関連パーツ名を変更する場合は、このリソース関連パーツを参照するビルド記述子パーツすべてを必ず変更してください。

IWN.MIG.1202.e リソース関連パーツ *partName* - */filetype=fileType* はサポートされません。変更する必要があります。

説明: VisualAge Generator は、一部のワークステーション環境では */filetype=BTRIEVE* と */filetype=MFCOBOL* をサポートします。EGL は、これらのファイル・タイプをサポートしません。マイグレーション・ツールは、EGL リソース関連パーツに **filetype** の情報を組み込みます。値が無効です。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。

ユーザーの処置: **filetype** 値を、EGL によってサポートされる値に変更する必要があります。EGL の **filetype** プロパティ、および使用可能なオプションについて詳しくは、オンライン・ヘルプのリソース関連パーツに関する情報を参照してください。

IWN.MIG.1203.e リソース関連パーツ *partName* - */system* は *targetSystem* ですが、これはサポートされません。*/filetype fileType* の情報に基づいてマイグレーションしました。

説明: リソース関連パーツには、指定された *targetSystem* を使用するエントリーが含まれています。このターゲット・システムは、EGL ではサポートされません。マイグレーション・ツールは、*fileType* に基づいてリソース関連エントリーをマイグレーションします。例えば、*targetSystem* が *mvs** で、*fileType* が *transient* である場合、マイグレーション・ツールは、EGL リソース関連エントリーを作成し、EGL **system** を *mvs** に設定します。これは無効です。EGL 検証により、「問題」ビューにエラー・メッセージが表示されます。エントリーを訂正する場合は、有効な EGL **system** (この例では *zoscsics*) を指定します。

ユーザーの処置: 「問題」ビューにエラーがある場合は、有効なターゲット・システムを指定して、リソース関連パーツのエントリーを訂正します。

IWN.MIG.1301.e リンク・エディット・パーツ *partName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、プログラムの名前を自動的に変更しません。プログラム名は対応するリンク・エディット・パーツと一致している必要があるので、マイグレーション・ツールはリンク・エディット・パーツの名前変更も行いません。

ユーザーの処置: プログラム名を変更する場合は、対応するリンク・エディット・パーツの名前を必ず変更してください。

IWN.MIG.1401.e バインド制御パーツ *partName* は予約語です。名前を変更する必要があります。

説明: マイグレーション・ツールは、プログラムの名前を自動的に変更しません。プログラム名は対応するバインド制御パーツと一致している必要があるので、マイグレーション・ツールはバインド制御パーツの名前変更も行いません。

ユーザーの処置: プログラム名を変更する場合は、対応するバインド制御パーツの名前を必ず変更してください。

VisualAge Generator から EGL マイグレーション・ツールへのメッセージ - ステージ 3

以下は、ステージ 3 で出されるメッセージです。

IWN.MIG.0030.i マイグレーション・セット

Name_version -- インポート分析が開始されました。

説明: これは、マイグレーション・ツールから出される、状況を示す情報メッセージです。

ユーザーの処置: なし。

IWN.MIG.0031.i マイグレーション・セット

Name_version -- インポート分析が完了しました。

説明: これは、マイグレーション・ツールから出される、状況を示す情報メッセージです。

ユーザーの処置: なし。

IWN.MIG.0032.i マイグレーション・セット

Name_version -- ステージ 3 に対しては処理されません。

説明: これは、マイグレーション・ツールから出される、状況を示す情報メッセージです。指定したマイグレーション・セットのバージョンはステージ 3 の実行中に処理されませんでした。これは、マイグレーション・セットの別のバージョンがワークスペースにインポートされるためです。例えば、マイグレーション・セットの最新バージョンをインポートするように要求したものの、その指定したバージョンが指定したマイグレーション・セットの最新バージョンではない場合などです。

ユーザーの処置: なし。

IWN.MIG.0033.e EGL ソース・ファイル *fileName* - *exceptionText* のマージ中に例外が発生しました。

説明: ステージ 3 のマイグレーション中に、「既存ファイルを上書き」のオプションの指定を解除しました。現行のマイグレーション・セットからの新規パーツは、既存ファイルにマージされることになっています。しかし、マイグレーション・ツールは、指定されたファイルのマージを実行できませんでした。

ユーザーの処置: IBM サポートに連絡して支援を受けてください。ステージ 3 のマイグレーション中に使用したワークスペースとマイグレーション・データベースを提供できるように準備しておいてください。

付録 D. 「問題」ビューのメッセージ

未確定状態では、マイグレーション・ツールがマイグレーション時に作成する正しい EGL 構文を常に判別できるとは限りません。未確定状態は通常、関連パーツがマイグレーション時に使用できない場合に起こります。これらの場合、マイグレーション・ツールは、EGL 検証がエラー・メッセージを「問題」ビューに表示したりコンパイル・エラーが発生したりするように、意図的に無効な EGL 構文を作成することがあります。下の表に、エラーを引き起こす特定のテキスト・ストリングを示します。具体的な EGL エラー・メッセージまたはコンパイラ・メッセージは異なる場合がありますが、左側の欄にリストされているテキスト・ストリングが、エラーとしてフラグを立てられた EGL ステートメントの近くに表示されます。マイグレーション・ツールがこれらのテキスト・ストリングを組み込むとき、ツールはマイグレーション・ログにもメッセージを出します。

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
###KEYS_NOT_FOUND###	<p>問題: 現行の SQL レコードには、別のレコードの構造が埋め込まれています。マイグレーション時に、embed キーワードによって指定されたレコードを使用できませんでした。keyItems プロパティには、VAGen の現行 SQL レコードに関して指定されたキー項目が含まれていますが、埋め込まれているレコードのキーが欠落しています。</p> <p>解決策: embed キーワードによって指定されている EGL レコードを見つけます。###KEYS_NOT_FOUND### のテキストを、埋め込まれる SQL レコードにリストされているキーに置き換えます。埋め込まれているレコードのキーを現行レコードのキー項目と組み合わせて、埋め込まれているレコードのレコード構造内での出現順にこれらのフィールドが並ぶようにします。現行レコードのキー項目が、埋め込まれているレコードの keyItems プロパティでも指定されている場合は、EGL の keyItems プロパティにこのフィールドを 1 回だけ組み込みます。</p>
###TABLES_NOT_FOUND###	<p>問題: 現行の SQL レコードには、別のレコードの構造が埋め込まれています。マイグレーション時に、embed キーワードによって指定されたレコードを使用できませんでした。</p> <p>解決策: embed キーワードによって指定されている EGL レコードを見つけて、tableNames と tableNameVariables プロパティを現行の SQL レコードにコピーします。</p>

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
EZE_DUPLICATE	<p>問題: VAGen の IF、WHILE、または TEST の各ステートメントで指定されているレコードを、マイグレーション時に使用できませんでした。</p> <p>解決策: EGL の if または while ステートメントで指定されている EGL レコードを見つけます。EZE_DUPLICATE を以下のいずれかの値に設定します。</p> <ul style="list-style-type: none"> • 非 SQL レコードの場合は duplicate • SQL レコードの場合は unique
EZE_NULL	<p>問題: マイグレーション・ツールは、VAGen の IF、WHILE、または TEST の各ステートメントに指定されている項目が、SQL レコードまたはマップのどちらにあるか判別できません。</p> <p>解決策: プログラムを検討して、フィールドが SQL レコードまたは書式のどちらにあるか判別します。EZE_NULL を以下のいずれかの値で置き換えます。</p> <ul style="list-style-type: none"> • SQL フィールドの場合は null • 書式フィールドの場合は blanks
EZE_SCRIPT	<p>問題: VisualAge Generator の GUI では、EZESCRPT は Java または Smalltalk のメソッドを呼び出すために使用されます。EGL には対応するシステム関数がありません。マイグレーション・ツールは EZE_SCRIPT を使用して、EGL に変換できないステートメントを示します。</p> <p>解決策: この関数を変更するか、または使用されていない関数を含むプロジェクトにこの関数を移動します。</p>
EZE_SETPAGE();	<p>問題: VAGen SET map PAGE ステートメントで指定されたマップを、マイグレーション時に使用できませんでした。</p> <p>解決策: EZE_SETPAGE() ステートメントに付随する、// VAGen Info のコメントで指定されている書式を見つけます。EZE_SETPAGE を以下のいずれかの関数に変更してください。</p> <ul style="list-style-type: none"> • テキスト書式の場合は converseLib.clearScreen() • 印刷書式の場合は converseLib.pageEject()

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
EZE_UNKNOWN_PARTTYPE	<p>問題: マイグレーション・データベースに保管されている外部ソース形式が無効でした。マイグレーション・ツールは、パーツ型を判別できず、パーツを EGL 構文に変換できませんでした。</p> <p>解決策: EZE_UNKNOWN_PARTTYPE ステートメントに指定されているパーツは無効です。この問題が少数のパーツのみに対して起こる場合は、VisualAge Generator から外部ソース形式をエクスポートし、単一ファイル・モードでこれらのパーツのマイグレーションを試行してください。</p> <p>ユーザー独自のツールを作成してマイグレーション・データベースをロードしている場合は、ツールが外部ソース形式のコードをマイグレーション・データベースにロードする方法に問題があることが考えられます。問題の原因の判別に役立つ照会については、547 ページの『付録 G. マイグレーション・データベース』を参照してください。</p>
EZE_UNKNOWN_PCB_MAPPING	<p>問題: マイグレーション中に、プログラム用に指定された PSB パーツが使用できませんでした。マイグレーション・ツールは、pcbParms プロパティに指定する値を判別できませんでした。</p> <p>解決策: プログラム内で、psb という名前の変数のデータ宣言を見つけてください。psb に対して指定される型定義は、プログラムの EGL PSBRecord パーツの名前です。プログラムの pcbParms プロパティを変更し、入力 PCB パラメーターを、EGL PSBRecord パーツ内の対応する PCB にマップします。詳しくは、(502 ページの) メッセージ IWN.MIG.0809.e を参照してください。</p>
EZE_UNKNOWN_PCB_TYPE	<p>問題: マイグレーション時に、プログラムに関して指定された PSB パーツが使用できなかったか、または含まれていた PCB の数がプログラムのパラメーター・リストで指定された数を下回っていました。マイグレーション・ツールは、プログラムの PCB パラメーターに使用する型定義を判別できませんでした。パラメーター・リスト内のそれぞれの PCB は、pcbn です。ここで、<i>n</i> は、EGL PSBRecord パーツ内の PCB に対応する数値リテラルです。</p> <p>解決策: プログラム内で、psb という名前の変数のデータ宣言を見つけてください。psb に指定される型定義は、プログラムの EGL PSBRecord パーツの名前です。プログラムの PCB パラメーターを変更して、EGL PSBRecord 内の対応する PCB の型定義に基づいて正しい型定義レコード (IO_PCBRecord、ALT_PCBRecord、DB_PCBRecord、または GSAM_PCBRecord) を指定します。詳しくは、(502 ページの) メッセージ IWN.MIG.0808.e を参照してください。</p>

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
EZE_UNKNOWN_QUALIFIER	<p>問題: 現在のセグメント検索指数 (SSA) に、修飾されていない EGL ホスト変数 (VAGen 比較値項目) が含まれています。SSA の DL/I セグメント・レコードまたはその代替仕様レコードが、マイグレーション中に使用できませんでした。あるいは、レコードは使用可能でしたが、比較値項目が含まれていませんでした。マイグレーション・ツールは、EGL ホスト変数の修飾子を判別できませんでした。</p> <p>解決策: 現在の SSA の DL/I セグメント・レコードまたはその代替仕様を見つけてください。ホスト変数がそのレコード内の項目であるかどうかを判別します。レコード内の項目である場合は、ホスト変数の修飾子を DL/I セグメント・レコード名に変更します。レコード内の項目でない場合は、使用する正しい修飾子を決定してください。正しい修飾子の判別方法についての詳細は、(497 ページの) メッセージ IWN.MIG.0611.e を参照してください。</p>
EZE_UNKNOWN_RELOP	<p>問題: 変更された DL/I ステートメントで、無効な関係演算子が使用されました。これは、関係演算子に適していない値が保管される原因となった、VisualAge Generator 内の問題によって発生している可能性があります。マイグレーション・ツールは、正しい関係演算子を判別できませんでした。</p> <p>解決策: VisualAge Generator で DL/I Call Editor を使用して、指定された関数の SSA を検討します。演算子が関数の外部形式ファイルに誤って保管されていても、正しい演算子が DL/I Call Editor に表示されます。EGL で関数を編集し、EZE_UNKNOWN_RELOP を正しい値に変更します。</p> <p>注: 問題を引き起こす可能性の最も高い演算子は、不等号に使用される記号です。EGL SSA における不等号の記号は != です。</p>
EZE_UNKNOWN_RETURN_COLUMN	<p>問題: VAGen の RETR ステートメントに指定されている VAGen テーブルが、マイグレーション時に使用不可でした。</p> <p>解決策: 代入ステートメントで指定されている EGL DataTable を見つけ、EZE_UNKNOWN_RETURN_COLUMN を DataTable の 2 列目の名前で置き換えます。</p>
EZE_UNKNOWN_SEARCH_COLUMN	<p>問題: VAGen の FIND ステートメントまたは RETR ステートメントに指定されている VAGen テーブルが、マイグレーション時に使用不可でした。</p> <p>解決策: if ステートメントで指定されている EGL DataTable を見つけ、EZE_UNKNOWN_SEARCH_COLUMN を DataTable の 1 列目の名前で置き換えます。</p>
EZE_UNKNOWN_SQLTABLE	<p>問題: 入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。マイグレーション・ツールは、EGL 入出力ステートメントの正しい tables 文節を判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つければ、tableNames または tableNameVariables レコード・プロパティ、あるいはその両方から正しいテーブル文節を判別します。</p>

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
EZE_UNKNOWN_SQL_FORUPDATEOF	<p>問題: VisualAge Generator は、SQL の入出力オプション UPDATE または SETUPD に関して、デフォルトの FOR UPDATE OF 文節を作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールは、EGL の入出力ステートメントに適した FOR UPDATE OF 文節を判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、そのレコード内のフィールドのリストから正しい FOR UPDATE OF 文節を判別します。VisualAge Generator のデフォルトの FOR UPDATE OF 文節は、レコードから得られた列名を、レコード内でのフィールドのリスト順と同じ順序で並べたリストですが、以下の場合には列を省略しています。</p> <ul style="list-style-type: none"> レコードの EGL keyItems プロパティにリストされている列名 EGL の isReadOnly = YES プロパティを使用して指定された列名 <p>入出力ステートメントで指定されているレコードが別の SQL レコードを埋め込む場合は、2 つのレコードを使用して、以下の方法で列名を判別します。</p> <ul style="list-style-type: none"> embed キーワードによって指定されているレコードを使用して、列の順序と isReadOnly = YES プロパティを判別します。 入出力ステートメントで指定されているレコード (埋め込み側レコード) を使用して、keyItems プロパティを判別します。 <p>FOR UPDATE OF 文節を EGL の prepare ステートメント内で使用する場合は、列名のリストを二重引用符で囲みます。</p>
EZE_UNKNOWN_SQL_INSERTCOLNAME	<p>問題: VisualAge Generator は、SQL の ADD 入出力オプションに対してデフォルトの列のリストを作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールは EGL の add ステートメントの正しい列名リストを判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、そのレコード内のフィールドのリストから正しい列のリストを判別します。VisualAge Generator のデフォルトの列名リストは、レコードから得られた列名を、レコード内でのフィールドのリスト順と同じ順序で並べたリストですが、EGL の isReadOnly = YES プロパティを使用して指定された列名を省略しています。入出力ステートメントで指定されているレコードが別のレコードを埋め込んでいる場合は、embed キーワードによって指定されているレコード名を使用して、列の順序と isReadOnly = YES プロパティを判別します。この列名のリストは、EGL の prepare ステートメントでは使用されません。</p>

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
EZE_UNKNOWN_SQL_INT0	<p>問題: VisualAge Generator は、SQL の入出力オプション INQUIRY、SETINQ、UPDATE、または SETUPD の INTO 文節に関して、デフォルトのデータ項目のリストを作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールは、EGL の入出力ステートメントの正しい INTO 文節を判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、INTO 文節のための正しいフィールドのリストを判別します。VisualAge Generator のデフォルトのフィールド・リストは、レコード内でのフィールドのリスト順と同じ順序でレコードのフィールドを並べたリストです。入出力ステートメントで指定されているレコードが別のレコードを埋め込んでいる場合は、embed キーワードによって指定されたレコードを使用して、フィールドの順序を判別してください。INTO 文節が EGL prepare ステートメントで使用されることはありません。</p>
EZE_UNKNOWN_SQL_ORDERBY	<p>問題: デフォルトの SQL が使用され、「Execution time statement build」オプションが指定されたときに、VisualAge Generator が SQL の SETINQ 入出力オプションの SQL ORDER BY 文節のための列位置のデフォルト・リストを作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールが、EGL 入出力ステートメントのための正しい ORDER BY 文節を判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、ORDER 文節の正しい列位置のリストを判別します。VisualAge Generator での列位置のデフォルト・リストは、ASC オプションが後続く EGL の keyItems プロパティ内の各項目に対応するフィールド位置のリストです。このレコードの最初のフィールドが位置 1 と見なされます。入出力ステートメントで指定されているレコードが別のレコードを埋め込んでいる場合は、埋め込まれているレコードではなく、その入出力ステートメントで指定されているレコードの EGL keyItems プロパティを使用して、ORDER BY 文節を判別します。EGL の prepare ステートメントの ORDER BY 文節を、以下のフォーマットで作成してください。</p> <p>"order by keyField1Position, keyField2Position asc"</p>

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
EZE_UNKNOWN_SQL_SELECT	<p>問題: VisualAge Generator は、SQL の入出力オプション INQUIRY、SETINQ、UPDATE、または SETUPD の SELECT 文節に関して、デフォルトの列のリストを作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールは、EGL の入出力ステートメントの正しい SELECT 文節を判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、SELECT 文節の正しい列名のリストを判別します。 VisualAge Generator でのデフォルトの列名リストは、レコードから得られた列名を、レコード内でのフィールドのリスト順と同じ順序で並べたリストです。入出力ステートメントで指定されているレコードが別のレコードを埋め込んでいる場合は、embed キーワードによって指定されたレコードを使用して、列の順序を判別してください。SELECT 文節を EGL の prepare ステートメント内で使用する場合は、列名のリストを二重引用符で囲みます。</p>
EZE_UNKNOWN_SQL_SET	<p>問題: SQL の SQLEXEC 入出力オプションがデフォルトの SQL とともに使用され、model オプションが UPDATE に設定され、Execution time statement build オプションが指定されていたときに、VisualAge Generator が SQL SET 文節の列と値のデフォルト・リストを作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールが、EGL 入出力ステートメントの正しい SET 文節を判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、SET 文節の正しい列と値のリストを判別します。 VisualAge Generator における列と値のデフォルトのリストは、そのレコードの列とそれに対応するフィールド名のリストで、それらの順序は、そのレコード内でリストされている列とフィールドの順序と同じです。ただし、以下の場合には列とそれに対応するフィールドを省略しています。</p> <ul style="list-style-type: none"> レコードの EGL keyItems プロパティにリストされている列名 EGL の isReadOnly = YES プロパティを使用して指定された列名 <p>入出力ステートメントで指定されているレコードが別の SQL レコードを埋め込む場合は、2 つのレコードを使用して、以下の方法で列名を判別します。</p> <ul style="list-style-type: none"> embed キーワードによって指定されているレコードを使用して、列の順序と isReadOnly = YES プロパティを判別します。 入出力ステートメントで指定されているレコード (埋め込み側レコード) を使用して、keyItems プロパティを判別します。 <p>prepare ステートメントの SET 文節を以下のフォーマットで作成します。</p> <pre>" set columnName1 = " + fieldName1 + " , columnName2 = " + fieldName2</pre>

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
EZE_UNKNOWN_SQL_SQLEXEC	<p>問題: VisualAge Generator で、SQLEXEC 入出力オプションに関して以下のいずれかの状態が発生しました。</p> <ul style="list-style-type: none"> • レコードも SQL 文節も指定されていなかった。 • レコードは指定されていたが、Model オプションが NONE に設定されていた。 <p>この関数は、VisualAge Generator では無効です。したがって、マイグレーション・ツールは、どの入出力ステートメントが作成されることになっているのかを判別できません。</p> <p>解決策: VAGen ソース・コードを検討して、この関数の実際の意図を判別します。</p>
EZE_UNKNOWN_SQL_VALUES	<p>問題: VisualAge Generator は、SQL の ADD 入出力オプションの値を指定するために、デフォルトのデータ項目のリストを作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールは EGL の add ステートメントの VALUES 文節の正しいフィールド名リストを判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、VALUES 文節の正しいフィールドのリストを判別します。</p> <p>VisualAge Generator のデフォルトのフィールド名リストは、レコード内でのフィールドのリスト順と同じ順序でレコードのフィールドを並べたリストですが、EGL の isReadOnly = YES プロパティを使用して指定されたフィールドを省略しています。入出力ステートメントで指定されているレコードが別のレコードを埋め込んでいる場合は、embed キーワードによって指定されているレコード名を使用して、フィールドの順序と isReadOnly = YES プロパティを判別します。VALUES 文節が EGL prepare ステートメントで使用されることはありません。</p>

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
<p>EZE_UNKNOWN_SQL_WHERE ここで、VAGen の入出力オプションは、以下のオプションのいずれかです。</p> <ul style="list-style-type: none"> • INQUIRY • UPDATE • model オプションが UPDATE または DELETE のいずれかに設定された SQLEXEC 	<p>問題: デフォルトの SQL が使用され、Execution time statement build オプションが指定されていたときに、VisualAge Generator が、SQL の INQUIRY、UPDATE、または SQLEXEC 入出力オプションに関する SQL WHERE 文節を作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールが、EGL 入出力ステートメントの正しい WHERE 文節を判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、EGL レコードの defaultSelectCondition および keyItems プロパティから正しい WHERE 文節を判別します。prepare ステートメントの WHERE 文節を以下のようにして作成します。</p> <ul style="list-style-type: none"> • defaultSelectCondition が存在し、キー項目が存在していない場合は、defaultSelectCondition から WHERE 文節を作成します。例えば、次のようになります。 <code>" where " + recordDefaultSelectCondition</code> • defaultSelectCondition は存在していないが、1 つ以上のフィールドが EGL の keyItems プロパティにリストされている場合は、すべてのキー項目を使用して WHERE 文節を作成します。例えば、次のようになります。 <code>" where keyColumn1 = " + keyField1 + " and keyColumn2 = " + keyField2</code> • defaultSelectCondition が存在し、1 つ以上のフィールドが EGL の keyItems プロパティにリストされている場合は、デフォルトの選択条件とすべてのキー項目を使用して、WHERE 文節を作成します。例えば、次のようになります。 <code>" where " + recordDefaultSelectCondition + " and keyColumn1 = " + keyField1 + " and keyColumn2 = " + keyField2</code> • WHERE 文節に defaultSelectCondition プロパティからの情報が含まれている場合は、keyItems プロパティの場合に示したように、列情報と演算子をテキスト・リテラルに変換し、ホスト変数名からセミコロンを省略します。 • それ以外は、いずれの場合も WHERE 文節を省略します。

表 165. EGL 構文エラーまたは検証エラーを引き起こす VAGen マイグレーション・テキスト (続き)

EGL 構文内の VAGen マイグレーション・テキスト	問題と解決策
<p>EZE_UNKNOWN_SQL_WHERE ここで、VAGen の入出力オプションは、以下のオプションのいずれかです。</p> <ul style="list-style-type: none"> • SETINQ • SETUPD 	<p>問題: デフォルトの SQL が使用され、Execution time statement build オプションが指定されていたときに、VisualAge Generator が、SQL の SETINQ、または SETUPD 入出力オプションに関する WHERE 文節を作成しました。入出力オブジェクトとして指定されている SQL レコードが、マイグレーション時に使用不可でした。このため、マイグレーション・ツールが、EGL 入出力ステートメントの正しい WHERE 文節を判別できませんでした。</p> <p>解決策: 入出力ステートメントで指定されているレコードを見つけて、EGL レコードの defaultSelectCondition および keyItems プロパティから正しい WHERE 文節を判別します。prepare ステートメントの WHERE 文節を以下のようにして作成します。</p> <ul style="list-style-type: none"> • defaultSelectCondition が存在し、キー項目がないかまたは複数存在する場合は、defaultSelectCondition から WHERE 文節を作成します。例えば、次のようになります。 <code>" where " + recordDefaultSelectCondition</code> • defaultSelectCondition は存在していないが、1 つのフィールドのみが EGL の keyItems プロパティにリストされている場合は、そのキー項目を使用して WHERE 文節を作成します。例えば、次のようになります。 <code>" where keyColumn >= " + keyField</code> • defaultSelectCondition が存在し、1 つのフィールドのみが EGL の keyItems プロパティにリストされている場合は、デフォルトの選択条件とそのキー項目を使用して WHERE 文節を作成します。例えば、次のようになります。 <code>" where " + recordDefaultSelectCondition + " and keyColumn = " + keyField</code> • WHERE 文節に defaultSelectCondition プロパティからの情報が含まれている場合は、keyItems プロパティの場合に示したように、列情報と演算子をテキスト・リテラルに変換し、ホスト変数名からセミコロンを省略します。 • それ以外は、いずれの場合も WHERE 文節を省略します。

付録 E. 「問題」ビューの IWN.xxx メッセージ

IWN.SYN、IWN.VAL、および IWN.XML のメッセージの中には、完全に EGL 内で開発したコードの場合より、VisualAge Generator からマイグレーションした EGL ソース・コードの場合に出される可能性が高いものがあります。ここでは、マイグレーションされたコードに対して特別な意味のあるメッセージをリストします。

ファイルに非常に多くのエラーがある場合、以下の順序でエラーを解決するのが最適です。

- 無効な構文に対する IWN.SYN メッセージ。通常これらのメッセージは、以下のいずれかの結果として発行されます。
 - プログラム、FormGroup、書式、または DataTable の名前として EGL 予約語が使用されていて、マイグレーション・ツールでそれを名前変更できない。
 - マイグレーション・ツールで使用される意図的な無効構文。これらのエラーの解決方法について詳しくは、509 ページの『付録 D. 「問題」ビューのメッセージ』を参照してください。
- 解決不可能、またはあいまいなパーツに対する IWN.VAL メッセージ。
IWN.VAL.3260.e、IWN.VAL.6619.e、および IWN.VAL.6620.e などのメッセージを含みます。
- 別のレコード、書式、DataTable と同じ名前のフィールドが存在していることを示す、あるステートメントに関する IWN.VAL 警告メッセージで、
IWN.VAL.6570.w、IWN.VAL.6571.w、および IWN.VAL.6621.w などのメッセージがあります。
- その他のメッセージ。

注: EGL はファイルごとに最大で 40 のメッセージを作成します。したがって、追加メッセージを表示するには、一部のメッセージを解決してファイルを保存し、ワークスペースを再構築する必要がある場合があります。

.egl ファイルの IWN.VAL メッセージ

IWN.VAL.3012.e 同じ *recordName* という名前が、関数、プログラム、またはライブラリー *programName* で変数宣言、パラメーター宣言、使用宣言、または定数宣言としても使用されています。

説明: VisualAge Generator は、パラメーター・リストとテーブルおよび追加レコード・リストに同じレコード名を指定することを容認しました。この状態で、VisualAge Generator は、テーブルおよび追加レコード・リスト内のレコードを無視しました。

ユーザーの処置: プログラムを編集し、レコードの宣言を除去します。

IWN.VAL.3260.e タイプ *partName* は解決できません。

説明: 指定されたパーツが見つかりません。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているように、コンテキストに基づいて変化します。

ユーザーの処置: 意味は、以下のコンテキストに基づいて変化します。

- *partName* が、プログラムのレコード宣言リストで型定義として使用されるレコードである。
partName_level77Suffix という名前のレコードがあるかどうかを確認してください。ここで、*_level77Suffix* は

マイグレーションのステージ 2 で指定した設定です。VisualAge Generator は、プログラムのテーブルおよび追加レコード・リストにレベル 77 項目のみを含む作業用ストレージのレコードを容認しています。ただし、これらのレベル 77 項目はプログラムで参照できません。プログラムの 1 次作業用ストレージ・レコード内のレベル 77 項目のみがプログラムで参照できます。レコードにはレベル 77 項目のみが含まれているため、マイグレーション・ツールはレベル 77 レコードのみを作成しました。このような場合、プログラムを編集して、オリジナルのレコード名の宣言を削除します。項目が VAGen プログラムで参照されなかったため、対応するレベル 77 レコードを追加しません。

- *partName* が、ヘルプ FormGroup として使用されている。VAGen プログラムはヘルプ・マップ・グループを指定していますが、プログラムが対話するマップはヘルプ・マップを指定していません。プログラムを編集します。ヘルプ FormGroup の **use** ステートメントを除去するか、またはヘルプ FormGroup を含むパッケージに関する **import** ステートメントを追加します。
- *partName* が、PSBRecord 内の **segmentRecord** または **parentRecord** プロパティとして使用されている。VisualAge Generator は、PSB で DL/I セグメント・レコードが欠落している場合、そのセグメントがプログラムで実際に DL/I 入出力用として使用されるか、あるいはより低いレベルのセグメント用にデフォルトの SSA をビルドするために使用されない限り、エラー・メッセージを発行しません。EGL では、PSB で参照される DL/I セグメント・レコードはすべて定義されている必要があります。PSB を使用するすべてのプログラムの PSBRecord およびロジックを検討します。欠落している DL/I セグメント・レコードを指定する PCB をプレースホルダーとしてのみ使用し、非 EGL プログラムに移動する場合は、PCB を変更して、階層情報を除去することを考慮してください。あるいは、欠落している DLISegment レコード定義を作成またはインポートします。

IWN.VAL.4925.e *programName* の変数宣言 *recordName* を解決できませんでした。

説明: プログラムが DL/I を使用している場合、指定された *recordName* は、プログラムの PSBRecord で指定された DL/I セグメントの名前になっている場合があります。VisualAge Generator では、PSB においてプログラムに関して指定されている DL/I セグメントは、すべてプログラムの関連パーツと見なされます。これにより、デフォルト SSA で使用される DL/I セグメントは、すべてプログラムで使用可能になります。

VisualAge Generator は、DL/I セグメント・レコードが

欠落している場合、そのセグメントが実際に DL/I 入出力用として使用されるか、あるいはより低いレベルのセグメント用にデフォルトの SSA をビルドするために使用されない限り、エラー・メッセージを発行しません。プログラムで PSB が指定されている場合、マイグレーション・ツールは、入出力オブジェクトとして使用されるか、または入出力オブジェクトへの階層パスで使用されるすべての DL/I セグメントに関するデータ宣言を自動的に組み込みます。

ユーザーの処置: プログラムを検討して、DL/I セグメント・レコードが必要であるかどうかを判別します。レコードが使用されておらず、より低いレベルのセグメントのデフォルト SSA のビルドに必要なでない場合は、プログラムを編集して、レコードの宣言を除去します。

IWN.VAL.4930.e プログラム *programName* 内の **use** 宣言 *FormGroupName* の値が無効です。
FormGroup、**DataTable**、**列挙**、または**ライブラリー・パーツ**を使用する必要があります。

説明: 指定された *FormGroupName* が欠落しているか、または予約語です。

ユーザーの処置: 欠落している *FormGroup* をマイグレーションします。*FormGroup* 名が予約語の場合は、*FormGroup* 名、およびその *FormGroup* を参照しているすべてのプログラムを変更します。

IWN.VAL.5004.e **DataTable** *tableName* は *n1* 列で定義されていますが、その内容は *n2* 列で定義されています。

説明: **DataTable** で有効に定義された列の数が、内容のリストに一致していません。以下のいずれかの状態が発生した可能性があります。

- 1 つ以上のフィールドが型定義で定義されますが、その型定義を解決することはできません。
- VisualAge Generator で、数値フィールドの小数点としてコンマが使用されました。EGL では常にピリオドを使用します。マイグレーション構文の設定「**10 進数のコンマを小数点に変更**」を選択した場合、マイグレーション・ツールはコンマを小数点に変換します。

ユーザーの処置: フィールドの 1 つに解決できない型定義がある場合は、最初にその問題を訂正し、その **DataTable** を含むプロジェクトを再ビルドします。それでもまだ問題があり、VisualAge Generator で小数点としてコンマを使用した場合は、設定を確認してください。**DataTable** を含むファイルから **import** ステートメントを保存し、次に単一ファイル・モードを使用して再度このテーブルをマイグレーションしてテーブルの内容

を訂正します。次に、このファイルに **import** ステートメントを追加します。

IWN.VAL.5052.e *programName* - レコードは **null** としか比較できません。

説明: 一般に、このエラーが発生するのは、レコード、書式、または *DataTable* に、レコードと同じ名前を持つフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前はフィールドに解決されます。EGL では、名前はレコードに解決されますが、レコードをこのタイプの比較に使用することはできません。すべての VAGen レコードは EGL 構造化レコードにマイグレーションされることに注意してください。

ユーザーの処置: ネーム解決により問題が発生する場合は、フィールドを含んでいるレコード、書式、または *DataTable* の名前でフィールド名を修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.5085.e *programName* - 要素 *operandName* は式に使用できません。

説明: 一般に、このエラーが発生するのは、レコード、書式、または *DataTable* に、別のレコード、書式、または *DataTable* と同じ名前を持つフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前は演算式のフィールドに解決されます。EGL では、名前はレコード、書式、または *DataTable* に解決されますが、これらは演算式には使用できません。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいるレコード、書式、または *DataTable* の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.5089.e *programName* - *operandName1* は、*operandName2* との比較には使用できません。

説明: 一般に、このエラーが発生するのは、レコード、書式、または *DataTable* に、別のレコード、書式、または *DataTable* と同じ名前を持つフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前はこのタイプの比較用のフィールドに解決されます。EGL では、名前はレコード、書式、または *DataTable* に解決されますが、これらはこのタイプの比較には使用できません。ネーム解決

の問題は、*operandName1* または *operandName2* のいずれかに関する名前の競合により発生することがあります。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName1* または *operandName2* を、そのフィールドを含んでいるレコード、書式、または *DataTable* の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.5090.e *programName* - **in** 条件内のオペランド *operandName* は、項目かまたはリテラルである必要があります。

説明: 一般に、このエラーが発生するのは、レコード、書式、または *DataTable* に、別のレコード、書式、または *DataTable* と同じ名前を持つフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前は **FIND**、**RETR**、**IF x IN array**、または **WHILE x IN array** ステートメントのフィールドに解決されます。EGL では、名前はレコード、書式、または *DataTable* に解決されますが、これらは **if x in** または **while x in** ステートメントには使用できません。VAGen **FIND** および **RETR** ステートメントはどちらも、EGL の **if x in structuredFieldArray** ステートメントにマイグレーションされることに注意してください。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいるレコード、書式、または *DataTable* の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.5093.e *programName* - **is/not** 条件のオペランド *operandName* はテキスト書式フィールドである必要があります。

説明: 一般に、このエラーが発生するのは、別のレコード、書式、または *DataTable* と同じ名前フィールドが書式に存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前は **IF x IS CURSOR** や **IF x NOT DATA** などのステートメントのマップ・フィールドに解決されます。EGL では、名前はレコード、書式、または *DataTable* に解決されますが、これらはこのタイプの **if** ステートメントには使用できません。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいる書式の名前で修飾します。ネーム解決が問題であることを確

認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.5094.e *programName* - *operandName* は、現行の **is/not** 式では無効です。可変のテキスト書式フィールド、**CHAR**、**MBCHAR**、**DBCHAR**、および **UNICODE** は、ニーモニック・ブランクと一緒に使用する場合に有効な型です。

説明: 一般に、このエラーが発生するのは、レコード、書式、または DataTable に、別のレコード、書式、または DataTable と同じ名前を持つフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前は **IF x IS BLANKS** ステートメントのフィールドに解決されます。EGL では、名前はレコード、書式、または DataTable に解決されますが、これらは **if x is blanks** ステートメントには使用できません。VAGen のニーモニック **BLANK**、**BLANKS**、および **NULLS** は EGL のニーモニック **blanks** にマイグレーションされることに注意してください。VAGen のニーモニック **NULL** は、比較がマップ・フィールドに関するものである場合は EGL のニーモニック **blanks** にマイグレーションされ、比較が SQL フィールドに関するものである場合には **null** にマイグレーションされます。詳しくは、136 ページの『SQL 項目とマップ項目が NULL かどうかの検査』を参照してください。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいるレコード、書式、または DataTable の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.5095.e *programName* - **is/not** 条件のオペランド *operandName* は、ニーモニック **null**と一緒に使用する場合には無効です。

説明: 一般に、このエラーが発生するのは、SQL レコードに、別のレコード、書式、または DataTable と同じ名前のフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前は **IF x IS NULL** ステートメントのフィールドに解決されます。EGL では、名前はレコード、書式、または DataTable に解決されますが、これらは **if x is null** ステートメントには使用できません。VAGen のニーモニック **NULL** は、比較がマップ・フィールドに関するものである場合は EGL のニーモニック **blanks** にマイグレーションされ、比較が SQL フィールドに関するものである場合には **null** にマイグレーションされ

ることに注意してください。詳しくは、136 ページの『SQL 項目とマップ項目が NULL かどうかの検査』を参照してください。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいるレコードまたは書式の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.5101.e *mainFunctionName* - このステートメントの位置で指定した場合、**xxxxx** システム・ワードは無効です。

説明: 指定された **main** 関数を使用するプログラムが、他の関数を呼び出しています。関数呼び出しチェーンに含まれる関数の 1 つが、指定されたシステム・ワードをステートメント内で使用しています。マイグレーション・ツールは、VAGen の **EZE** ワードを置換する EGL システム・ワードを常に修飾します。**xxxxx** システム・ワードが **sysLib**、**sysVar**、**mathLib**、**strLib**、**vgLib**、**vgVar**、**converseLib**、**converseVar**、**dliLib**、または **dliVar** で修飾されていない場合、最も可能性の高い原因として、以下のものが考えられます。

- VAGen プログラムが暗黙データ項目を許容しており、**xxxxx** の定義がマイグレーション時に自動的に作成された。EGL は、暗黙データ項目を許容しません。マイグレーション・ツールはまた、暗黙データ項目の定義を自動的に作成しません。
- レコード、マップ、またはテーブルがマイグレーション・セットに含まれていなかったため、マイグレーション・ツールが必要な **import** ステートメントをプログラムに組み込むことができなかった。

ユーザーの処置: VAGen プログラムが暗黙項目を許容していたかどうか確認します。許容していた場合は、VisualAge Generator 内でプログラムを検証します。VAGen の「メッセージの表示 (View Messages)」リスト内のメッセージに、暗黙データ項目の正しい定義が表示されます。プログラムの宣言セクションに、データ項目の定義を追加します。マイグレーションのステージ 1 を実行する前に暗黙項目を作成するために役立つホワイト・ペーパーについては、19 ページの『参照』を参照してください。

VAGen プログラムが暗黙項目を許容していなかった場合は、VisualAge Generator 内でプログラムの関連リストを作成します。この関連リストから、VAGen 参照ツールを使用して、指定されたデータ項目を検索します。

「参照 (References)」ツールの結果は、どのレコード、マップ、またはテーブルがマイグレーション・セットから欠落しているかを知る手掛かりになります。

IWN.VAL.5143.e *programName* - *operandName* は、現行の **is/not** 式では無効です。ニーモニック数字と一緒に使用することができる型は、**CHAR**、**MBCHAR**、**UINCODE**、および **STRING** のみです。

説明: 一般に、このエラーが発生するのは、レコード、書式、または **DataTable** に、別のレコード、書式、または **DataTable** と同じ名前を持つフィールドが存在している場合です。**VisualAge Generator** では、ネーム解決はコンテキストに依存するため、名前は **IF x IS NUMERIC** ステートメントのフィールドに解決されます。**EGL** では、名前はレコード、書式、または **DataTable** に解決されますが、これらは **if x is numeric** ステートメントには使用できません。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいるレコード、書式、または **DataTable** の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ **IWN.VAL.6621.w** のユーザー応答を参照してください。

IWN.VAL.5168.e *xxxxx* は、**is/not** 式の中で使用した場合には無効です。

説明: 指定された値が無効です。マイグレーション済み **VAGen** コードに関する意味は、ユーザー応答で記述されているように、コンテキストに基づいて変化します。

ユーザーの処置: 意味は、以下のコンテキストに基づいて変化します。

- 値が **EZE_DUPLICATE** または **EZE_NULL** である場合、関連パーツがマイグレーション・セットにインクルードされていないため、マイグレーション・ツールはこのステートメントをマイグレーションする方法を判別できません。詳細および解決方法については、509 ページの『付録 D. 「問題」ビューのメッセージ』を参照してください。

IWN.VAL.5177.e *programName* - **is/not** 条件内のオペランド *operandName* は、**SQLRecord** 内の項目である必要があります。

説明: 一般に、このエラーが発生するのは、**SQL** レコードに、別のレコード、書式、または **DataTable** と同じ名前のフィールドが存在している場合です。**VisualAge Generator** では、ネーム解決はコンテキストに依存するため、名前は **IF x IS TRUNC** ステートメントなどのステートメントのフィールドに解決されます。**EGL** では、名前はレコード、書式、または **DataTable** に解決されますが、これらを **if x is trunc** ステートメントなどのステートメントに使用することはできません。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいる **SQL** レコードの名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ **IWN.VAL.6621.w** のユーザー応答を参照してください。

IWN.VAL.5196.e *programName* - **for** のカウント *countName* が無効です。**for** のカウントは、整数項目またはリテラルである必要があります。

説明: 一般に、このエラーが発生するのは、レコード、書式、または **DataTable** に、別のレコード、書式、または **DataTable** と同じ名前を持つフィールドが存在している場合です。**VisualAge Generator** では、ネーム解決はコンテキストに依存するため、名前は **MOVEA** ステートメント内のカウント用のフィールドに解決されます。**EGL** では、名前はレコード、書式、または **DataTable** に解決されますが、これらは **move** ステートメントの **for count** としては使用できません。

ユーザーの処置: ネーム解決により問題が発生する場合は、*countName* を、そのフィールドを含んでいるレコード、書式、または **DataTable** の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ **IWN.VAL.6621.w** のユーザー応答を参照してください。

IWN.VAL.5340.e プロパティ: 位置。 *formName* 内のフィールド *fieldName* に対するこのプロパティの値は無効です。値は形式 [**row**, **column**] である必要があり、ここでの **row** および **column** は正の整数です。

説明: **VisualAge Generator** は **row=0**, **column=0** にあるマップ・フィールドを許容します。このマップ・フィールドでは、いずれの属性も指定することができず、マップ上の前のフィールドと同じ属性が使用されます。**EGL** ではすべてのフィールドに属性バイトが含まれている必要があります。**row=0**, **column=0** にある定数フィールドが空白で始まる場合、マイグレーション・ツールは先頭文字を削除して位置を **row=1**, **column=1** に変更することによってフィールドの位置を調整します。このフィールドは、先頭文字としてブランクを含まない定数フィールド (*fieldName*) か、または変数フィールドのいずれかです。

ユーザーの処置: **EGL** エディターを使用して書式を変更します。フィールドが書式に合うように、フィールドの長さ **fieldLen**、**position**、**value**、およびその他のプロパティを必要に応じて調整します。表示プロパティを追加して、フィールドの外観が **VisualAge Generator** での外観と同じになるようにします。これらの変更を行

った後、 EGL フォーム・エディターを使用して書式に対して予想される変更を行うことができます。

IWN.VAL.6501.e *sqlStatement* SQL の入出力ステートメントは *sqlClause* 文節を必要としますが、この文節が欠落しています。

説明: メッセージ IWN.VAL.6506.e を参照してください。

ユーザーの処置: メッセージ IWN.VAL.6506.e を参照してください。

IWN.VAL.6503.e *xxxxx* SQL 入出力ステートメントに、適切でない文節があります。 *yyyyy* は、 *zzzzz* 文節の前に表示されます。

説明: メッセージ IWN.VAL.6506.e を参照してください。

ユーザーの処置: メッセージ IWN.VAL.6506.e を参照してください。

IWN.VAL.6506.e *xxxxx* SQL 入出力ステートメントには、 *yyyyy* 文節をただ 1 つ指定できます。

説明: このメッセージが出される可能性がある状況は、いくつかあります。

- VisualAge Generator の場合は、SQL キーワードを列名として使用できます。EGL の場合は、一部の SQL キーワードを使用できません。
- VisualAge Generator では、ある SQL 文節は括弧で囲みません。EGL では、これらの文節を括弧で囲む必要があります。例えば、副選択およびそれに関連した FROM、WHERE、GROUP BY、および ORDER BY 文節は、括弧で囲まなければなりません。

ユーザーの処置: 副選択を使用していない場合は、300 ページの『特殊な処理を必要とする SQL 予約語』で、SQL キーワードのリスト、および列名に関する問題を解決するために使用できる手法について参照してください。

副選択を使用している場合は、副選択とそれに関連する FROM、WHERE、GROUP BY、および ORDER BY 文節の周囲に括弧を追加します。例えば、以下の SQL ステートメントを考えてみましょう。

```
with #sql{
  SELECT EMPNO, COUNT(*) WORKDEPT
  FROM EMPLOYEE T1
  WHERE WORKDEPT LIKE '%E%'
  GROUP BY WORKDEPT
  UNION ALL
```

```
SELECT EMPNO
FROM EMP_ACT
WHERE PROJNO IN('MA2100', 'MA2112')
}
```

問題を修正するためには、以下の更新済みコードのように SQL ステートメントを変更して、2 番目の SELECT に括弧を追加します。

```
with #sql{
  SELECT EMPNO, COUNT(*) WORKDEPT
  FROM EMPLOYEE T1
  WHERE WORKDEPT LIKE '%E%'
  GROUP BY WORKDEPT
  UNION ALL
  ( SELECT EMPNO
  FROM EMP_ACT
  WHERE PROJNO IN('MA2100', 'MA2112'))
}
```

IWN.VAL.6507.e *xxxxx* SQL 入出力ステートメントは、 *yyyyy* 文節を許可しません。

説明: メッセージ IWN.VAL.6506.e を参照してください。

ユーザーの処置: メッセージ IWN.VAL.6506.e を参照してください。

IWN.VAL.6531.e *sqlClause* SQL 文節を空にすることはできません。

説明: メッセージ IWN.VAL.6506.e を参照してください。

ユーザーの処置: メッセージ IWN.VAL.6506.e を参照してください。

IWN.VAL.6541.e *programName* - 引き渡し側のレコード ID *operandName* はレコード変数である必要があります。

説明: 一般に、このエラーが発生するのは、入出力オブジェクトに、プログラム内の別のレコードと同じ名前を持つフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、*operandName* は XFER または DXFR ステートメントのレコードに解決されます。EGL では、入出力オブジェクト内のフィールドは、プログラム内で使用されるレコードよりも優先されます。さらに、**show** ステートメント (マップまたは UI レコードを含む VAGen XFER) は、EGL では入出力ステートメントです。

マイグレーション・ツールは、以下のようにして VAGen ステートメントを変換します。

- マップまたは UI レコードを含まない XFER は、EGL の **transfer to transaction** ステートメントにマイグレーションします。

- マップを指定した XFER および UI レコードを指定した XFER は、EGL の **show** ステートメントにマイグレーションします。
- DXFR は、EGL の **transfer to program** ステートメントにマイグレーションします。

ユーザーの処置: 問題の原因がネーム解決である場合は、名前がレコードに解決されるように、修飾子として EGL のキーワード **this** を指定します (例えば、**this.operandName**)。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.6570.w *programName* - 項目アクセス
fieldName がレコード、書式、または **DataTable** に解決されました。レコード、書式、または **DataTable** *containerName* に、*fieldName* と呼ばれる項目があります。

説明: 一般に、この警告が発生するのは、レコード、書式、または **DataTable** (*containerName*) に、別のレコード、書式、または **DataTable** と同じ名前を持つフィールド (*fieldName*) が存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前は、ステートメントに応じて、フィールド、レコード、マップ、または **DataTable** に解決されます。EGL では、名前は関数パラメーター項目、関数のローカル・ストレージ項目、レコード、書式、または **DataTable** に解決され、これらは有効ですが、必ずしも VisualAge Generator の場合と同じ名前に解決されるとは限りません。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいるレコード、書式、または **DataTable** の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.6571.w *programName* - 項目アクセス
fieldName は、レコード、書式、または **DataTable** *containerName* 内の項目に解決されました。プログラムが認識している *fieldName* という名前のレコード、書式、または **DataTable** があります。

説明: 一般に、この警告が発生するのは、入出力オブジェクト (*containerName*) に、プログラム内の別のレコード、書式、または **DataTable** と同じ名前を持つフィールド (*fieldName*) が存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存する

ため、名前は、ステートメントに応じて、フィールド、レコード、マップ、または **DataTable** に解決されます。EGL では、名前は有効な入出力オブジェクト内のフィールドに解決されますが、必ずしも VisualAge Generator の場合と同じ名前に解決されるとは限りません。

ユーザーの処置: この意味は、コンテキストに基づいて変化します。他の可能性を検討する前に必ず、ネーム解決の違いを考慮してください。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.6583.e *programName* - 配列参照
arrayName[*subscriptName*] 内の添え字
subscriptName は、整数項目または整数リテラルである必要があります。

説明: 一般に、このエラーが発生するのは、レコード、書式、または **DataTable** に、別のレコード、書式、または **DataTable** と同じ名前を持つフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、名前は添え字のフィールドに解決されます。EGL では、名前はレコード、書式、または **DataTable** に解決されますが、これらは添え字としては許可されていません。

ユーザーの処置: ネーム解決により問題が発生する場合は、*subscriptName* を、そのフィールドを含んでいるレコード、書式、または **DataTable** の名前で修飾します (例えば *containerName.fieldName*)。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.6619.e *programName* - *variableName* は解決できません。

説明: *variableName* はレコード名、書式名、またはレコード、書式、**DataTable**、あるいは関数内のフィールドである可能性があります。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: 意味は、以下のコンテキストに基づいて変化します。

- VAGen プログラムが暗黙の項目の定義を許可しており、*variableName* が非修飾のフィールド名であるかどうかを判別します。許容していた場合は、VisualAge Generator 内でプログラムを検証します。VAGen 検証メッセージがプログラムで使用される暗黙の項目の定義を指定します。EGL プログラムを編集し、必要な基本タイプの定義に対するガイドとして VAGen 検証メッセージを使用することにより、暗黙の項目の変数

定義を追加します。マイグレーションのステージ 1 を実行する前に暗黙項目を作成するために役立つホワイト・ペーパーについては、19 ページの『参照』を参照してください。

- プログラムが DL/I を使用しているかどうか、および指定された *variableName* が、プログラムの PSBRecord で指定された DL/I セグメントの名前であるかどうかを判別します。メッセージが **add** や **get** などの DL/I 入出力ステートメントを指している場合、より低いレベルのセグメント用にデフォルトの SSA をビルドするためには DL/I セグメント・レコードが必要です。プログラム・ロジックを検討して、デフォルト SSA をビルドするために DL/I セグメント・レコードが必要であるかどうかを判別します。必要である場合は、プログラムを編集して、指定されたレコード用に宣言を追加します。
- プログラムが DL/I を使用しているかどうか、および修飾子が EZE_UNKNOWN_QUALIFIER であるかどうかを判別します。VisualAge Generator は、ある時点で DL/I 比較値項目の修飾子の判別方法を変更しました。マイグレーション・ツールは、使用する修飾子を判別できませんでした。詳しくは、509 ページの『付録 D. 「問題」ビューのメッセージ』の EZE_UNKNOWN_QUALIFIER 情報、メッセージ IWN.MIG.0611.e (497 ページ)、および 127 ページの『DL/I I/O および比較値項目』を参照してください。
- プログラムが DL/I を使用しているかどうか、および指定された *variableName* が **dliVar.name** であるかどうかを判別します。VisualAge Generator では、プログラムが PSB を指定していない場合でも EZE DL/I 状況ワード (EZEDL* ワード) の使用が許可されています。EGL では、プログラムが PSBRecord を宣言している場合にのみ **dliVar** での変数の使用が許可されています。プログラムを編集して、PSBRecord に関する宣言を追加するか、または **dliVar** ライブラリーで使用している変数を除去します。
- マイグレーション・セットの一部のパーツをマイグレーションしなかった場合、マイグレーション・ツールは適切な **import** ステートメントを組み込むことができません。パーツがワークスペースに存在する場合、エラーを含むファイルに **import** ステートメントを追加する必要が生じる場合があります。
- *variableName* が修飾された名前である場合 (例えば X.Z)、修飾子 (X) は未確定です。VisualAge Generator では、フィールド名はレコード、テーブル、またはマップ内で一度しか出現できないため、許可されている修飾子は、レコード、テーブル、またはマップ名のみです。EGL では、レコードまたは DataTable 内の複数のサブストラクチャーで同じフィールド名を使用することができるため、レコードまたは DataTable 内の

フィールドを修飾子にすることもできます。プログラムが、あるレコードまたは DataTable 内のフィールドと同じレコード名、DataTable 名、または書式名を持っているかどうかを判別します。例えば、Z という名前のフィールドを含む X という名前の書式と、X という名前のフィールドを含む Y という名前のレコードがあり、フィールド X には A という名前のサブフィールドがあるとしてします。VAGen では、X は書式 X のみを意味することができるので、X.Z は有効な参照です。EGL では、X は書式 X か、またはレコード Y 内のフィールド X のいずれかの可能性があるため、X.Z は未確定です。X が未確定であるため、Z が X という名前の書式内でのみ使用されていても、X が解決されるまでは X.Z を解決することはできません。EGL キーワード **this** を修飾子として使用してみてください (例えば、**this.X.Z** とします)。あるいは、書式の名前を変更するか、またはレコード (Y) 内のフィールド (X) の名前を変更します。DataTable がレコード内のフィールドと同じ名前を持っている場合にも同じ状態が発生します。

- *variableName* が修飾されていない場合は、メッセージ IWN.VAL.6621.w を参照して、他に可能性がないか確認してください。

IWN.VAL.6620.e *programName* - 変数アクセス *xxxxx* は未確定です。

説明: *variableName* は、レコード、書式、DataTable、または関数内のフィールドである場合があります。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: 意味は、以下のコンテキストに基づいて変化します。

- *variableName* が修飾されない項目名である場合、**call** ステートメントまたはシステム関数呼び出しに関して問題が発生しているかどうかを判別します。修飾されない項目名が **CALL** ステートメントまたは EZE 関数呼び出しに使用されている場合、VisualAge Generator は、プログラムの 1 次作業用ストレージ・レコード内のレベル 77 項目を優先します。EGL には、同じ優先順位は用意されていません。*variableName* で指定されているフィールドが **inputRecord** プログラム・プロパティに対応するレベル 77 レコードに存在する場合は、**call** ステートメントまたはシステム関数呼び出しを変更して、レベル 77 レコード名でその項目を修飾します。ただし、この関数を呼び出すプログラムすべてが、同じレベル 77 レコードを使用していることを確認する必要があります。

- エラーのある関数をチェックして、EGL の **show** ステートメント (マップを指定した VAGen の XFER、または UI レコードを指定した XFER) が存在しているかどうかを判別します。マップを指定した VAGen の XFER および UI レコードを指定した XFER は、入出力ステートメントとは見なされません。EGL の **show** ステートメントは入出力ステートメントです。EGL の **show** ステートメントが存在していると、関数内のすべての ステートメントに関するネーム解決規則が変更され、ネーム解決が未確定状態になってしまう可能性があります。以下の状態について考えてください。
 - 関数のオリジナルの入出力オブジェクト内のフィールドは、EGL の **show** ステートメントで使用される書式または VGUIRecord 内のフィールドと同じ名前が付けられます。VAGen では、名前はオリジナルの入出力オブジェクト内のフィールドに解決されます。EGL では、オリジナルの入出力オブジェクト内のフィールドと、**show** ステートメントのフィールド入出力オブジェクトのフィールドは、ネーム解決に関してはどちらも同じカテゴリーに入ります。名前をオリジナルの入出力オブジェクト内のフィールドに解決する必要がある場合は、そのフィールド名をオリジナルの入出力オブジェクトの名前で修飾します (例えば、*recordName.xxxxx*)。
 - 関数のオリジナルの入出力オブジェクト内のフィールドは、EGL の **show** ステートメントで使用される書式または VGUIRecord と同じ名前が付けられます。VAGen では、名前はオリジナルの入出力オブジェクト内のフィールドに解決されます。EGL では、フィールド名、および **show** ステートメントで使用される書式または VGUIRecord は、ネーム解決に関しては同じカテゴリーに入るため、名前が未確定になります。名前をオリジナルの入出力オブジェクト内のフィールドに解決する必要がある場合は、そのフィールド名をオリジナルの入出力オブジェクトの名前で修飾します (例えば、*recordName.xxxxx*)。
- variableName* が関数呼び出しの場合、名前がレコード、書式、または DataTable 内のフィールドの名前と競合する場合があります。VisualAge Generator では、ネーム解決はコンテキストに依存するため、共用されていないフィールド名が関数名と同じになる可能性があります。EGL ではこれは許可されていません。関数およびその関数が使用されている箇所をすべて名前変更します。あるいは、その関数呼び出しを、その関数を含むパッケージの名前で修飾します。例えば、次のようになります。

packageName.functionName(argumentList);

IWN.VAL.6621.w *programName* - オペランド

operandName1 が書式、レコード、または DataTable に解決され、オペランド

operandName2 がプリミティブ項目に解決されました。VAGen では、どちらのオペランドも同じ項目に解決されていた可能性があります。

説明: 一般に、このエラーが発生するのは、レコード、書式、または DataTable に、別のレコード、書式、または DataTable と同じ名前を持つフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、*operandName1* はフィールドに解決されます。EGL では、名前はレコード、書式、または DataTable に解決されます。ネーム解決規則の違いの詳細については、537 ページの『メッセージの参照情報 - ネーム解決規則および名前の修飾規則』を参照してください。ステートメントのコンテキストに応じて、ネーム解決のこの違いが原因となり、EGL のさまざまなメッセージが出される可能性があります。ネーム解決の問題は、*operandName1* または *operandName2* のいずれかに関する名前の競合により発生することがあります。

ユーザーの処置: 特定のメッセージおよび意味は、ステートメントのコンテキスト、およびフィールドの名前がレコード、書式、または DataTable と同じであるかどうかによって変化します。一般には、この問題を解決するには、以下の技法を試してください。

- エラーのある関数をチェックして、EGL の **show** ステートメント (マップを指定した VAGen の XFER、または UI レコードを指定した XFER) が存在しているかどうかを判別します。マップを指定した VAGen の XFER および UI レコードを指定した XFER は、入出力ステートメントとは見なされません。EGL の **show** ステートメントは入出力ステートメントです。EGL の **show** ステートメントが存在していると、関数内のすべての ステートメントに関するネーム解決規則が変更され、以下のいずれかの方法でネーム解決が変更される可能性があります。
 - VisualAge Generator 内のレコード、書式、またはテーブルが EGL の **show** ステートメントの入出力オブジェクト内のフィールドに解決される。
 - VisualAge Generator のオリジナルの入出力オブジェクト内のフィールドが、**show** ステートメントで使用される書式または VGUIRecord に解決される。
- この問題が、別のレコード、書式、または DataTable と同じ名前を持つレコード、書式、または DataTable 内のフィールドによるものであることを確認します。VisualAge Generator では、以下の手順を実行します。

1. プログラムを生成して、それが有効なプログラムであることを確認します。
 2. 関連ツールを使用して、プログラムの関連パーツをすべて見つけます。
 3. 関連パーツ・リストに対して参照ツールを実行します。参照ツールに関して以下の情報を指定します。
 - 「**検索対象**」を「**テキスト**」検索に設定します。
 - 「**テキスト**」ストリングを「*operandName1*」に設定します。
 - 「**検索の範囲**」を「**リスト内のすべてのパーツ**」に設定します。
 4. 参照ツールの結果は、*operandName1* が項目の名前であるかどうか、およびレコード、マップ、またはテーブルの名前であるかどうかを判別するのに役立ちます。この結果は、*operandName1* の指定可能な修飾子を判別するのにも役立ちます。
operandName2 に対してこのプロセスを繰り返します。
 5. 指定可能な修飾子が複数ある場合は、537 ページの『メッセージの参照情報 - ネーム解決規則および名前修飾規則』の VAGen 修飾規則を検討して、正しい修飾子を判別してください。エラーが発生しているステートメントが **call** ステートメントまたは関数呼び出しである場合は、VAGen が生成した COBOL プログラムをチェックして、VisualAge Generator がこの名前をどのように解決したのかを判別する必要がある場合があります。
- 以下のいずれかの方法で EGL ソース・コードを変更します。
 - 名前を関数パラメーター・リストまたはローカル・ストレージ内のレコードまたは項目変数に解決する必要がある場合は、関数パラメーター・リストまたはローカル・ストレージの名前を変更することを検討してください。レコードまたは項目タイプの定義は変更しないでください。この技法では、変更はエラーが発生する関数のみに制限されています。必ず、関数内の変数へのすべての参照を変更してください。
 - 名前を VAGen レベル 77 項目に解決する必要がある場合は、EGL フィールド名を、マイグレーション時に指定したレベル 77 の接尾部を含めて、レコードの名前で修飾します (例えば、*recordName_level77Suffix.operandName1*)。ただし、この関数を呼び出すプログラムすべてが、同じレベル 77 レコードを使用していることを確認する必要があります。
 - 名前をその他のフィールドに解決する必要がある場合は、そのフィールドを含んでいるレコード、書式、または DataTable の名前でそのフィールド名を修飾します (例えば、*recordName.operandName1*)。
 - 名前をレコードまたは書式に解決する必要がある、そのステートメントを含んでいる関数内に、レコードまたは書式と同じ名前のフィールドを含む入出力オブジェクトがある場合は、EGL のキーワード **this** を修飾子として指定します (例えば、*this.operandName1*)。
-
- IWN.VAL.6650.e** *programName - targetName* はレコードであるため、代入ソースはレコードであるか、**CHA**、**HEX**、または **MBCHAR** に評価される必要があります。
- 説明:** 一般に、このエラーが発生するのは、レコード、書式、または DataTable に、レコードと同じ名前のフィールドが存在している場合です。VisualAge Generator では、ネーム解決はコンテキストに依存するため、ソースが数値リテラルまたは数値フィールドである場合には、*targetName* はフィールドに解決されます。EGL では、名前はレコードに解決されます。
- ユーザーの処置:** ネーム解決により問題が発生する場合は、*targetName* を、そのフィールドを含んでいるレコード、書式、または DataTable の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。
-
- IWN.VAL.6653.e** *programName - primitiveType1* および *primitiveType2* は、式 *expressionText* では互換タイプではありません。
- 説明:** マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。
- ユーザーの処置:** 意味は、以下のコンテキストに基づいて変化します。
- *primitiveType1* が **HEX** で、変数が **mathLib** 関数 **ceiling**、**floor**、**compareNum**、**precision**、または **round** の結果である場合、EGL は、これらの関数の結果としての **HEX** をサポートしません。プログラム・ロジックを変更して、プリミティブ型が *primitiveType2* によって指定される結果変数を使用するようにしてください。
 - このエラーは、レコード、書式、または DataTable に、別の書式または DataTable と同じ名前を持つフィールドが存在している場合にも発生する可能性があります。VisualAge Generator では、ネーム解決はコン

テキストに依存するため、*primitiveType1* で表される変数はフィールドに解決されます。EGL では、名前は書式または *DataTable* に解決されます。ネーム解決により問題が発生する場合は、フィールドを含んでいるレコード、書式、または *DataTable* の名前を変数名を修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ *IWN.VAL.6621.w* のユーザー応答を参照してください。

IWN.VAL.6665.e *programName* - 移動のソース
operandName が無効です。ソースは、レコード、書式、項目、リテラル 1、または定数である必要があります。

説明: 一般に、このエラーが発生するのは、レコード、書式、または *DataTable* に、別の *DataTable* と同じ名前のフィールドが存在している場合です。*VisualAge Generator* では、ネーム解決はコンテキストに依存するため、*operandName* はフィールドに解決されます。EGL では、名前は *DataTable* に解決されます。

ユーザーの処置: ネーム解決により問題が発生する場合は、*operandName* を、そのフィールドを含んでいるレコード、書式、または *DataTable* の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ *IWN.VAL.6621.w* のユーザー応答を参照してください。

IWN.VAL.6676.e *programName* - 移動のターゲット
targetName[subscriptName] が無効です。ターゲットは、ソース・スカラーと互換性のある型の項目配列である必要があります。

説明: 一般に、このエラーが発生するのは、レコード、書式、または *DataTable* に、別のレコード、書式、または *DataTable* と同じ名前を持つフィールドが存在している場合です。*VisualAge Generator* では、ネーム解決はコンテキストに依存するため、ソースとターゲット (*targetName*) はフィールドに解決されます。EGL では、ソースまたはターゲットはレコード、書式、または *DataTable* に解決されます。ネーム解決の問題は、ソース・フィールドまたはターゲット・フィールドのいずれかに関する名前競合により発生することがあります。

ユーザーの処置: ネーム解決により問題が発生する場合は、ソースまたはターゲット (*targetName*) を、そのフィールドを含んでいるレコード、書式、または *DataTable* の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ *IWN.VAL.6621.w* のユーザー応答を参照してください。

IWN.VAL.6695.e *functionName* - この項目参照では、状態 *XXXXX* は許可されていません。

説明: マイグレーション済み *VAGen* コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: 意味は、以下のコンテキストに基づいて変化します。

- 状態が *PROTECT*、*SKIP*、*INVISIBLE*、*BLINK*、*INITIALATTRIBUTES*、または色である場合、このフィールドは印刷書式に存在します。*VisualAge Generator* は、プリンター書式に関してこれらの属性の設定を容認しています。EGL は許容しません。関数を変更してステートメントを削除します。また、テキスト書式と印刷書式の両方に同じ関数を使用されている場合は、印刷書式に使用する関数のコピーを作成するか、または現在の関数を呼び出すテキスト書式関数にステートメントを移動する必要があります。
- 状態が *EMPTY* である場合は、書式またはレコードがマイグレーション時に使用不可であったかどうか判別してください。EGL は独立データ項目の定義を許可し、型定義が見つからない場合は、定義が項目に対するものであると想定します。EGL はレコードや書式のフィールドに関する *set empty* の使用をサポートしません。欠落したレコードや書式を定義またはマイグレーションします。

IWN.VAL.6697.e *programName* - 状態
formFieldStateName は、レコード参照に関しては許可されていません。

説明: 一般に、このエラーが発生するのは、レコードと同じ名前を持つフィールドが書式にある場合です。*VisualAge Generator* では、ネーム解決はコンテキストに依存するため、*SET* ステートメントが *CURSOR* や色などのプロパティを対象にしている場合には、名前はマップ上のフィールドに解決されます。EGL では、名前はレコードに解決されます。

ユーザーの処置: ネーム解決により問題が発生する場合は、*set* ステートメントで指定されているフィールドを、書式の名前で修飾します。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ *IWN.VAL.6621.w* のユーザー応答を参照してください。

IWN.VAL.6716.e *programName* - 引数 *argumentName* は関数 *functionName* の *inOut* パラメーター *parameterName* に渡すことができません。タイプ *type1* および *type2* が参照について互換ではありません。

説明: *functionName* は呼び出し関数の名前です。

inOut パラメーターに渡される引数は、参照について互換性を持つ必要があります。参照に互換性があるとは、引数タイプとパラメーター・タイプが完全に一致する必要があることを意味します。このエラーは、以下のいずれかの理由で発生することがあります。

- ケース VisualAge Generator は、引数とパラメーターに異なるタイプを使用することをサポートしません (ただし、場合によってはそれを容認します)。
- レコード、書式、または DataTable に、別のレコード、書式、または DataTable と同じ名前を持つフィールドがあります。

マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: 意味は、以下のコンテキストに基づいて変化します。

- *functionName* がシステム関数で、EGL がレコードまたは書式に解決する。これにより、VisualAge Generator が、たとえ意味をなさなくても、レコードまたは書式を容認している状態になっている場合があります。例えば、VisualAge Generator は、レコードを EZE 数学関数の引数として渡し、書式をストリング関数に引数として渡すことを容認しています。プログラム・ロジックを検討して、その意図を判別してください。
- *functionName* がシステム関数で、EGL がフィールドに解決するが、このフィールドが正しい型ではない。VisualAge Generator は、パラメーター型と互換性のない、いくつかの引数型を容認しています。EGL は型に互換性があることを要求します。引数に対してサブストラクチャー (または親フィールド) を追加して、正しい定義を持つフィールドを指定します。例えば、**sysLib.startTransaction** の 3 番目の引数に問題が発生し、その 3 番目の引数が INT 型のフィールドである場合、CHAR 型の親フィールドを追加して、3 番目の引数として使用します。CHAR 型のフィールドを親フィールドとして指定すると、レコードが初期化されるときに、必ずサブストラクチャーの INT 型フィールドがバイナリー・ゼロに初期化されます。
- 呼び出された関数がシステム・ユーザー関数で、EGL がフィールドに解決するが、このフィールドが正しい型ではない。この状態では、呼び出された関数を変更して、幅広い引数型および引数長を許容する **number** などの関数パラメーター型を使用するようにできる場合もあります。それ以外には、呼び出された関数と、その関数が呼び出されたすべての場所を検討する方法があります。呼び出された関数を、それぞれ異なるパラメーターが定義された、複数の関数に分割しなければならない場合があります。

- 別のレコード、書式、または DataTable と同じ名前を持つフィールドが存在するかどうかを検討してください。ネーム解決が問題であることを確認する方法、および指定可能な修飾子の判別の方法については、メッセージ IWN.VAL.6621.w のユーザー応答を参照してください。

IWN.VAL.6731.e *programName* - 引数 *argumentName* は関数 *functionName* の **in** または **out** パラメーター *parameterName* に渡すことができません。タイプ *type1* および *type2* は、代入互換性がありません。

説明: *functionName* は呼び出し関数の名前です。in または out パラメーターに渡された引数は、代入互換性がある必要があります。VisualAge Generator は、代入互換性のないいくつかのタイプを容認しています。EGL では互換性のないタイプは許可されていません。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: メッセージ IWN.VAL.6716.e のユーザー応答を参照してください。

IWN.VAL.6736.e *programName* - 式 *argumentName* は変換操作では無効です。この式はレコードに評価されるか、あるいは **blob** または **clob** 以外にプリミティブ型を持つ必要があります。

説明: *functionName* は呼び出し関数の名前です。引数は固定長レコードであるか、またはプリミティブ型である必要があります。マイグレーション・ツールはすべての VAGen レコードを構造化レコードに変換します。VisualAge Generator は、代入互換性のないいくつかのタイプを容認しています。EGL では互換性のないタイプは許可されていません。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: メッセージ IWN.VAL.6716.e のユーザー応答を参照してください。

IWN.VAL.6741.e *programName* - 関数呼び出し *functionName* の引数 *argumentName* が無効です。この引数はプリミティブ型の **char**、**mbchar**、**dbchar**、**hex**、**num**、または **unicode** である必要があります。

説明: *functionName* は呼び出し関数の名前です。引数は、リストされているプリミティブ型に限定されています。VisualAge Generator は、代入互換性のないいくつかのタイプを容認しています。例えば、VisualAge

Generator は、VAGen の文書によれば無効とされていても、レコードを引数としてストリング関数に渡すことを容認しています。EGL では互換性のないタイプは許可されていません。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: メッセージ IWN.VAL.6716.e のユーザー応答を参照してください。

IWN.VAL.6742.e *programName* - 関数呼び出し
functionName の引数 *argumentName* が無効です。この引数はプリミティブ型の **char**、**dbchar**、**hex**、**num**、**bin**、**int**、**smallint**、**bigint**、**pacf**、**money**、または **decimal** である必要があります。

説明: *functionName* は呼び出し関数の名前です。引数は、リストされているプリミティブ型に限定されています。VisualAge Generator は、代入互換性のないいくつかのタイプを容認しています。EGL では互換性のないタイプは許可されていません。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: メッセージ IWN.VAL.6716.e のユーザー応答を参照してください。

IWN.VAL.6743.e *programName* - 引数 *argumentName* を関数 *functionName* の緩いパラメーター *parameterName* に渡すことはできません。この引数はプリミティブ型の *type1* である必要があります。

説明: *functionName* は呼び出し関数の名前です。引数は、リストされているプリミティブ型に限定されています。VisualAge Generator は、代入互換性のないいくつかのタイプを容認しています。EGL では互換性のないタイプは許可されていません。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: *functionName* が **strLib.byteLen()** の場合は、すべてのプリミティブ型をサポートする **sysLib.bytes()** に置き換えることができる場合があります。それ以外の可能性については、メッセージ IWN.VAL.6716.e のユーザー応答を参照してください。

IWN.VAL.6744.e *programName* - 引数 *argumentName* を関数 *functionName* の緩いパラメーター *parameterName* に渡すことはできません。この引数は数値のプリミティブ型である必要があります。

説明: *functionName* は呼び出し関数の名前です。引数

は、リストされている数値のプリミティブ型に限定されています。VisualAge Generator は、代入互換性のないいくつかのタイプを容認しています。例えば、VisualAge Generator は、書式とレコードを数学関数に引数として渡したり、索引と長さをストリング関数に引数として渡したりすることを容認しています。EGL では互換性のないタイプは許可されていません。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているように、コンテキストに基づいて変化します。

ユーザーの処置: メッセージ IWN.VAL.6716.e のユーザー応答を参照してください。

IWN.VAL.6746.e 関数参照 *functionName* を解決できません。

説明: マイグレーション済み VAGen コードの意味は、「ユーザー応答」で説明されているように、コンテキストに基づいて変化します。

ユーザーの処置: 意味は、以下のコンテキストに基づいて変化します。

- このメッセージがエラーとして発行される (接尾部 "e")。指定された関数を見つけることができません。マイグレーション・セットに組み込まなかったパーツがある場合、マイグレーション・ツールは、編集ルーチンが関数であり、**validatorFunction** プロパティにマイグレーションされた、と誤って「推測」した可能性があります。*functionName* によって指定されたパーツを見つけてください。そのパーツが **DataTable** である場合は、エラーのあるファイルを編集して、**validatorFunction** プロパティを **validatorDataTable** に変更してください。そのパーツが関数である場合は、エラーのあるファイルに **import** ステートメントを追加して、その関数を含むパッケージを指すようにしなければなりません。
- このメッセージが警告として発行される (接尾部 "w")。VisualAge Generator では、関数が **DataItem** パーツの編集ルーチンとして指定されている場合、その **DataItem** が UI レコードで共用データ項目として使用されている場合にのみエラーが発生します。EGL では、**DataItem** パーツが型定義で使用されているかどうか (またはそれがどこで使用されているか) に関係なく、**validatorFunction** が使用可能である必要があります。この状態で警告メッセージを消去するには、**DataItem** パーツから **validatorFunction** プロパティを除去するか、またはエラーのあるファイルに **import** ステートメントを追加して、その関数を含むパッケージを指すようにします。

IWN.VAL.6751.e *programName* - 関数呼び出しのターゲットは、関数またはデリゲートである必要があります。

説明: 関数の名前が、レコード、書式、または DataTable 内のフィールドの名前と競合しています。VisualAge Generator では、ネーム解決はコンテキストに依存するため、共用されていないフィールド名が関数名と同じになる可能性があります。EGL ではこれは許可されていません。

ユーザーの処置: 関数およびその関数が使用されている箇所をすべて名前変更します。あるいは、その関数呼び出しを、その関数を含むパッケージの名前で修飾します。例えば、次のようになります。

```
packageName.functionName(argumentList);
```

IWN.VAL.7553.e *functionName* - *systemFunctionName* の引数 *n* は、ストリング項目、ストリング定数、またはストリング・リテラルであることが必要です。

説明: *functionName* は呼び出し関数の名前です。引数は、リストされている型に限定されています。VisualAge Generator は、代入互換性のないいくつかのタイプを容認しています。EGL では互換性のないタイプは許可されていません。マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: メッセージ IWN.VAL.6716.e のユーザー応答を参照してください。

IWN.VAL.7696.e DataTable *DataTableName* を項目 *type1* の *validatorDataTable* として使用することはできません。MatchValidTable 型および MatchInvalidTable 型のテーブルの場合、最初の列の型は項目の型に一致している必要があります。型 *type1* および *type2* が一致していません。

説明: VisualAge Generator は、データ項目の型がテーブルの最初の列の型に一致していない場合でも、そのテーブルをデータ項目用の編集ルーチンとして使用することを容認しています。EGL ではこの状態は許可されていません。

ユーザーの処置: DataItem パーツから *validatorDataTable* プロパティを除去するか、またはテーブルの最初の列の型を変更してください。

IWN.VAL.7740.e *programName* - *variableName* は読み取り専用であるため、これに対して代入を行うことはできません。

説明: *variableName* は、**sysVar.userID** などの EGL システム変数の 1 つです。マイグレーション・ツールは、EGL システム変数を常に EGL ライブラリー名で修飾します。*variableName* が修飾されておらず、大文字の場合、これは VisualAge Generator で自動的に作成される暗黙項目である可能性があります。EGL では暗黙項目は許可されていません。

ユーザーの処置: VisualAge Generator でプログラムを検証し、そのプログラムが暗黙項目を許可しているかどうか、および *variableName* がそのプログラム用に作成された暗黙データ項目の 1 つであるかどうかを判別してください。そうであれば、EGL プログラムを変更して、*variableName* に関する変数宣言を追加してください。EGL における項目の正しい定義を判別するには、VAGen 検証メッセージの情報を使用します。マイグレーションのステージ 1 を実行する前に暗黙項目を作成するために役立つホワイト・ペーパーについては、19 ページの『参照』を参照してください。

IWN.VAL.7755.w 到達不能コード

説明: **exit program**、**exit stack**、**return** などのステートメントの後には、1 つ以上のステートメントが続きます。**exit program**、**exit stack**、または **return** の後のステートメントは、実行されることはありません。

ユーザーの処置: 変更は不要です。ただし、警告を除去したい場合は、到達不能コードをコメント化するか削除してください。

IWN.VAL.7757.e 配列イニシャライザーの要素数は、項目 *fieldName* の発生回数を超えないようにする必要があります。発生サイズ *number2* に対して *number1* 個の要素が検出されました。

説明: VGUI レコードの場合は、「実行」ボタン配列または「**submitBypass**」ボタン配列の値のリストが配列のサイズとしては大きすぎます。VisualAge Generator は、生成時に余分な値を無視します。EGL では、配列サイズがリストの値の数と一致している必要があります。

ユーザーの処置: VGUI レコードを編集し、追加の値を除去します。

IWN.VAL.7789.e *propertyName* 項目 *fieldName* を乗算が発生する項目にすることはできません。

説明: VisualAge Generator は、UI レコードに対しては発生項目を配列として容認しますが、配列以外のものと同様に処理します。EGL は、乗算が発生する項目を **numElementsItem** プロパティで指定することを許可していません。

ユーザーの処置: VGUI レコードを編集して、指定された *fieldName* を変更し、それが配列にならないようにします。乗算が発生する親フィールドの外にフィールドを移動しなければならない場合があります。

IWN.VAL.7868.e *programName* - *stateName* の状態が DL/I セグメント・レコードでは無効です。

説明: DL/I レコードの場合、**if** または **while** ステートメントにおける *stateName* は無効です。VisualAge Generator は、DL/I では意味のない *stateNames* (例えば、FMT) をいくつか容認しています。EGL では、この *stateName* は許可されていません。

ユーザーの処置: プログラム・ロジックを見直して、そのプログラムの実際の意図を判別してください。このステートメントを変更するか、またはコメント化します。

.eglbld ファイルの IWN.VAL メッセージ

IWN.VAL.3999.e (10 の代替メッセージのうちの 1 つ目) XML 検証エラー - 属性 "xxxxx" は、すでに要素 "yyyyy" について指定されていました。

説明: 属性 xxxxx は、同じ制御パーツ内で複数回指定されています。XML パーサーは、.eglbld ファイルの処理を停止します。この結果、このエラーによって他のエラーが隠される可能性があります。このエラーによって .eglbld ファイルの処理が終了するため、.eglbld ファイル内の、エラーの時点の後で定義される未解決のパーツに関してエラーが発生することがあります。

ユーザーの処置: テキスト・エディターを使用して、現行の .eglbld ファイルを編集します。制御パーツ内で xxxxx が 1 つのみ指定されるように、制御パーツを変更します。.eglbld ファイルを保管すると、「問題」ビューのメッセージが更新されます。XML パーサーは、.eglbld ファイル内で最初に重複する属性の箇所で処理を停止するので、ファイル全体の構文解析を行うには、その前にいくつかのエラーを解決する必要があることがあります。

IWN.VAL.3999.e (10 の代替メッセージのうちの 2 つ目) XML 検証エラー - 要素タイプ "xxxxx" の内容は、"(listOfValues)" と一致している必要があります。

説明: VisualAge Generator 内では、リソース関連に対して指定された値のうち 1 つ以上が有効な値です。この値は、EGL によってサポートされていません。マイグレーション・ツールは、この値が無効であってもマイグレーションするので、EGL 検証は、ユーザーに問題の解決を促すためのエラー・メッセージを「問題」ビューに表示します。

ユーザーの処置: EGL のオンライン・ヘルプで xxxxx の有効なオプションを確認してください。使用するオプションを決定したら、必要な変更を行うために、テキスト・エディターでビルド記述子ファイルを開く必要があります。

IWN.VAL.3999.e (10 の代替メッセージのうちの 3 つ目) XML 検証エラー - 値 yyyyy を持つ属性 xxxxx には、リスト zzzzz からの値が必要です。

説明: VAGen 値すべてが、EGL に対応する置換表現を持っているわけではありません。

ユーザーの処置: マイグレーションによって値がどのように変換されるかについて詳しくは、419 ページの『制

御パーツ』のリンケージ・テーブルおよびリソース関連の表を参照してください。EGL で使用可能な選択については、EGL 文書を参照してください。

IWN.VAL.3999.e (10 の代替メッセージのうちの 4 つ目) XML 検証エラー - 要素タイプ "yyyyy" に関して、属性 "xxxxx" を宣言する必要があります。

説明: VAGen 値すべてが、EGL に対応する置換表現を持っているわけではありません。VisualAge Generator では有効な値の組み合わせが、EGL では無効になる場合があります。例えば、次のようになります。

- リンケージ・オプション・パーツでは、**library** 属性は **localCall** 項目に関しては無効です。EGL では、生成された Java コードからネイティブの C++ DLL への呼び出しは、同じマシン上で実行されている場合であっても、リモート呼び出しと見なされます。
- リンケージ・オプション・パーツでは、**remotePgmType** 属性は **localCall** 項目に関しては無効です。VAGen では、**remoteAppType** オプションはローカル呼び出しに対して無視されます。EGL では、XML 妥当性検査はさらに限定されます—有効な属性のみが許可されます。

ユーザーの処置: マイグレーションによって値がどのように変換されるかについて詳しくは、419 ページの『制御パーツ』のリンケージ・テーブルおよびリソース関連の表を参照してください。EGL で使用可能な選択については、EGL 文書を参照してください。

IWN.VAL.3999.e (10 の代替メッセージのうちの 5 つ目) XML 検証エラー - 要素タイプ "xxx" の後には、属性指定、">"、または ">" を続ける必要があります。

説明: リソース関連パーツ内のターゲット環境情報が無効です。これは、オリジナルの VAGen リソース関連項目に、/system=xxx*yyy または /system=xxx* が含まれている場合に発生する可能性があります。VAGen ではワイルドカード * は有効ですが、EGL では無効です。マイグレーション・ツールは、この項目を、有効である可能性のあるすべての EGL ターゲット・システムに変換するわけではありません。

ユーザーの処置: 有効な EGL ランタイム環境になるようこの項目を変更します。該当するすべての EGL ランタイム環境について、必要な回数だけこの項目を繰り返します。例えば、オリジナルの VAGen 項目が /system=mvs* の場合は、EGL 環境の ZOSBATCH と ZOSCICS についてこの項目を繰り返します。XML パー

サーは、ターゲット環境では * を含む最初のリソース関連項目で処理を停止するため、ファイル全体を構文解析するには、その前にいくつかのエラーを解決しておく必要があります。

IWN.VAL.3999.e (10 の代替メッセージのうちの 6 つ目) XML 検証エラー - 要素の内容は、整形形式の文字データまたはマークアップで構成されている必要があります。

説明: リソース関連パーツ内のターゲット環境情報が無効です。これは、オリジナルの VAGen リソース関連項目に /system=*xxx または /system=* が含まれている場合に発生する可能性があります。VAGen ではワイルドカード * は有効ですが、EGL では無効です。マイグレーション・ツールは、この項目を、有効である可能性のあるすべての EGL ターゲット・システムに変換するわけではありません。

ユーザーの処置: 有効な EGL ランタイム環境になるようこの項目を変更します。該当するすべての EGL ランタイム環境について、必要な回数だけこの項目を繰り返します。例えば、オリジナルの VAGen 項目が *cics の場合、有効な EGL ランタイム環境は zoscics のみです。XML パーサーは、ターゲット環境では * を含む最初のリソース関連項目で処理を停止するため、ファイル全体を構文解析するには、その前にいくつかのエラーを解決しておく必要があります。

IWN.VAL.3999.e (10 の代替メッセージのうちの 7 つ目) XML 検証エラー - 要素タイプ "yyyyy" には属性 "xxxxx" が必須であり、この属性を指定する必要があります。

説明: VisualAge Generator には、必須でない属性がいくつかありました。EGL では、yyyyy によって指定されている要素が定義されている場合、xxxxx によって指定される属性は必須です。

ユーザーの処置: テキスト・エディターを使用して必須属性を追加します。EGL で使用可能な選択については、EGL 文書を参照してください。

IWN.VAL.3999.e (10 の代替メッセージのうちの 8 つ目) XML 検証エラー - 要素タイプ elementType を宣言する必要があります。

説明: VAGen 値すべてが、EGL に対応する置換表現を持っているわけではありません。VisualAge Generator では有効な値の組み合わせが、EGL では無効になる場合があります。例えば、リソース関連ファイルでは、値 BTRIEVE は AIX 環境のファイル・タイプとして有効でした。EGL では、値 BTRIEVE は無効です。

ユーザーの処置: テキスト・エディターを使用してパー

ツを訂正します。マイグレーションによって値がどのように変換されるかについて詳しくは、419 ページの『制御パーツ』のリンケージ・テーブルおよびリソース関連の表を参照してください。EGL で使用可能な選択については、EGL 文書を参照してください。

IWN.VAL.3999.e (10 の代替メッセージのうちの 9 つ目) XML 検証エラー - コメント内ではストリング "--" を使用できません。

説明: XML では、ハイフンで構成されるストリングをコメント内で使用することはできません。この問題は、一般には、VAGen 制御パーツがコメントを含んでいて、ハイフンで構成される行を使用してブロック・コメントのセクションを区切っていた場合に発生します。

ユーザーの処置: テキスト・エディターを使用してパーツを訂正します。"--" で構成されるストリングを "==" で構成されるストリングで置き換えます。

IWN.VAL.3999.e (10 の代替メッセージのうちの 10 番目) XML 検証エラー - 3 バイトの encodingSequence シーケンスのうちのバイト 2 が無効です。

説明: ファイル内のビルド・パーツのうちの 1 つ以上に、ファイルに関して指定されているエンコード・シーケンスでは無効な各国語文字が含まれています。デフォルトのエンコード・シーケンスは UTF-8 です。

ユーザーの処置: 単一ファイル・マイグレーション、またはステージ 2 のマイグレーションの前にデフォルトのエンコード・シーケンスを変更するには、EGL デベロッパー環境から「Windows」->「設定」->「EGL」をクリックします。ドロップダウン・リストを使用して、使用している各国語文字をサポートするエンコード設定の値を選択します。マイグレーション後にデフォルトのエンコード・シーケンスを変更するには、システム・エディターを使用してファイル内のエンコード情報を変更します。エンコード情報が訂正されるまでは、EGL ビルド・パーツ・エディターでファイルを開くことはできません。

IWN.VAL.4300.e パーツ partName は、次のタイプのいずれかに解決できなかったか、または解決されませんでした: partTypeList

説明: マイグレーション済み VAGen コードに関する意味は、ユーザー応答で記述されているコンテキストに基づいて変化します。

ユーザーの処置: 意味は、以下のコンテキストに基づいて変化します。

- 指定された制御パーツが存在しない。そのパーツを作成するか、そのパーツの参照を除去します。
- 指定された制御パーツが、エラーのある制御パーツとは異なるプロジェクトまたはパッケージ内にある。
VisualAge Generator では制御パーツが関連パーツを持っていないため、マイグレーション・ツールは制御パーツに関して **import** ステートメントを作成しません。指定された制御パーツが別のプロジェクト内にある場合は、現行プロジェクトのプロパティの EGL ビルド・パスを更新して、指定された制御パーツが存在するプロジェクトを組み込みます。指定された制御パーツが別のパッケージ内にある場合は、現行の .eglbld ファイルを編集して、指定された制御パーツが存在するパッケージに関する **import** ステートメントを組み込みます。
- XML パーサーが .eglbld ファイルを完全に処理できなかった。この場合、指定された制御パーツは、エラーのある制御パーツと同じファイル内に存在する可能性があります。メッセージ「**IWN.VAL.3999.e XML 検証エラー - 属性 "xxxxx" は、すでに要素 "yyyyy" に対して指定されていました**」が出されているかどうか調べてください。このメッセージは、属性 xxxxx が同じ制御パーツに対して複数回指定されていることを示しています。制御パーツ内で xxxxx がただ 1 つ指定されるように、現行 .eglbld ファイルを編集します。「自動的にビルド」を選択した場合、.eglbld ファイルを保管すると、「問題」ビューにメッセージが更新されるはずですが、XML パーサーは、.eglbld ファイル内で最初に重複する属性の箇所で処理を停止するので、ファイル全体の構文解析を行うには、その前にいくつかのエラーを解決する必要があることがあります。すべての IWN.VAL.3999.e メッセージが解決されると、指定された制御パーツが参照側の制御パーツと同じ .eglbld ファイル内にあれば、その制御パーツは使用可能になります。

JSP の Java メッセージ

文字定数が無効です。(Invalid character constant)

説明: このエラーは以下の状態のときに発生します。

- VGUI レコード用に生成された `xxxxxBean.java` に関して、`setFillCharacter` メソッドが値「`nullFill`」を使用しているとき。VisualAge Generator では、共用データ項目に 2 つのプロパティ・セットがあります。1 つはマップ用で、もう 1 つは UI レコード用です。EGL では、`DataItem` パーツに 1 つのプロパティ・セットがあります。マイグレーション・ツールは、2 つのプロパティ・セットをマージし、UI プロパティを優先します。ただし、マイグレーション・ツールは、データ項目パーツがマップ、UI レコード、あるいはその両方のいずれで使われるかを予測できません。データ項目で UI のデフォルト充てん文字が指

定されなかったため、マイグレーション・ツールはマップのデフォルト文字 (Null) を使用しました。UI 充てん文字が指定されていなかったため、UI レコードで共用データ項目が使用されたときに、VisualAge Generator はブランクをデフォルトの充てん文字として使用しました。データ項目パーツを VGUI レコードの型定義として使用する場合は、EGL は Null の充てん文字を使用して生成しますが、これは無効です。

ユーザーの処置: `setFillCharacter` メソッドに値「`nullFill`」を指定したために問題が発生した場合は、VGUI レコードを編集して項目の指定変更プロパティを追加し、`fillCharacter` プロパティを `""` に設定します。これにより、EGL 書式で使用するためのデフォルトの充てん文字として `nullFill` が保存されます。

メッセージの参照情報 - ネーム解決規則および名前の修飾規則

VisualAge Generator および EGL のネーム解決規則および名前の修飾規則は、EGL で拡張されているため異なっています。ほとんどの場合、VAGen と EGL の名前の修飾規則およびネーム解決規則は問題を起こしません。ただし、レコード、書式、または `DataTable` 内のフィールドが、別のレコード、書式、または `DataTable` と同じ名前を持っている場合は、VisualAge Generator および EGL は異なる方法で名前を解決することがあります。以下のセクションでは、これら 2 つの製品の規則について検討し、この状態が発生したときに EGL 検証が「問題」ビューに表示するメッセージについて説明します。

VisualAge Generator のネーム解決規則および名前の修飾規則

VisualAge Generator は、2 つのパーツが同じ名前を持つことを許可していません。ただし、以下の場所で定義されている非共用項目は、レコード、マップ、またはテーブルと同じ名前になることがあります。

- レコードまたはテーブル内の非共用項目
- マップ上のフィールド (これは常に非共用項目として扱われます)
- 関数のローカル・ストレージまたはパラメーター・リスト内の非共用フィールド

さらに、複数の項目が同じ名前を持つことがあります。例えば、以下の場所では同じ名前を使用することができます。

- 関数のローカル・ストレージまたは関数パラメーター・リスト
- プログラムの 1 次作業用ストレージ・レコード内のレベル 77 項目
- プログラムに対して呼び出されるパラメーター・リスト内の項目
- 関数の入出力オブジェクト内の項目
- プログラムの他のレコード、マップ、またはテーブル内の項目

プログラム内の複数の場所で同じ名前が定義されている状態では、VAGen のネーム解決はコンテキストに依存します。例えば、次のようになります。

- 以下の状態では、VAGen ネーム解決は項目を使用します。

- 以下のいずれかに該当する代入ステートメントまたは MOVE ステートメント:
 - ソースがリテラルまたは別の項目である、または代入ステートメントの場合は演算式である。
 - ターゲットが別の項目である
- 任意のステートメント内の添え字。
- MOVEA ステートメント内のソース、ターゲット、または for のカウント。
- 比較の右辺が項目 (例えば、BLANKS、NUMERIC、NULLS、TRUNC など) とともにしか使用できない、または比較の左辺がリテラルまたは別の項目である、IF、WHILE、TEST ステートメント。
- 設定される値が、項目 (NULL や CURSOR など) とともにしか使用できない SET ステートメント。
- FIND または RETR ステートメント。
- 項目名が修飾されておらず、同じ名前の項目が複数ある場合、VisualAge Generator はこのセクションの後述するカテゴリー VG1 から VG3 までに基づいて名前を解決します。
- 以下の状態では、VAGen のネーム解決はレコードを使用します。
 - 別のレコードまたはマップをソースまたはターゲットとする MOVE ステートメント。
 - 比較の右辺がレコード (例えば、DUP や ERR) とともにしか使用できない IF、WHILE、または TEST ステートメント。
 - 設定される値 (EMPTY や SCAN など) がレコードとともにしか使用できない SET ステートメント。
 - DXFR ステートメント。
 - XFER、または UI レコード・ステートメントが指定された XFER。
- 以下の状態では、VAGen のネーム解決はマップを使用します。
 - 別のレコードまたはマップをソースまたはターゲットとする MOVE ステートメント。
 - 設定される値 (EMPTY、CLEAR、または PAGE など) がマップとともにしか使用できない SET ステートメント。
 - マップ・ステートメントを含む XFER。
- 予期されているパーツ型がステートメント・コンテキストで判別されない場合は、そのステートメント、そのステートメントが入出力関数内にあるかどうか、および項目とレコード、マップ、またはテーブルが同じ名前になっているかどうかに基づいて、優先順位が変化します。例えば、次のようになります。
 - 引数名がすべて固有になっている CALL ステートメントまたはシステム関数呼び出しの場合、問題はありません。フィールドが別のレコード、マップ、またはテーブルと同じ名前になっている場合、実行される VAGen ネーム解決は、レベル 77 項目の使用、呼び出されるパラメーター、テーブルおよび追加レコード・リスト内のレコード、入出力オブジェクトとしてのレコードまたは書式の使用などによって変化します。

注: テーブルおよび追加のレコード・リスト内の VisualAge Generator 4.5 フィックスパック 5 を使用するテストに基づき、CALL ステートメントまたは関数呼び出しに対しては整合性のあるパターンが決定されませんでした。

CALL ステートメントまたはシステム関数呼び出しに関してネーム解決が競合する場合、COBOL プログラムを生成することが、VisualAge Generator が名前をどのように解決したのかを判別する最も速い方法であることがあります。

- IF *x* IS MODIFIED ステートメントの場合、*x* は、マップまたは別のマップのフィールドである場合があります。この場合はマップが使用されます。別のマップのフィールドを参照するには、名前を *mapName.x* として修飾する必要があります。同様に、*x* が UI レコードであるか、または別の UI レコードのフィールドである場合は、その UI レコードが使用されます。別の UI レコードのフィールドを参照するには、名前を *UIRecord.x* として修飾する必要があります。

VisualAge Generator が項目を見つけることを予期している場合は、以下の名前の修飾規則を使用してその項目の配置場所が判別されます。

- 項目名が修飾されている場合、修飾子にできるのはレコード、マップ、またはテーブルです。項目名はレコード、マップ、またはテーブル内で固有である必要があるため、複数レベルの修飾は不要です (サポートされていません)。
- 項目名が修飾されていない場合、修飾子を判別するためにデータ項目が以下の順序で検査されます。
 - カテゴリー VG1 - この関数のローカル・ストレージおよびパラメーター・リスト内の項目名。
 - カテゴリー VG2 - この関数に固有の以下のレコードおよびマップ:
 - 入出力オブジェクトとその項目。
 - 関数パラメーター・リスト内のレコード、およびその項目。
 - 関数のローカル・ストレージ・リスト内のレコード、およびその項目。
 - このカテゴリー内で名前が固有でない場合は、その名前を修飾する必要があります。
 - カテゴリー VG3 - プログラム内の以下のレコード、マップ、およびテーブル:
 - プログラムの作業用ストレージ・レコードとそれらの項目。
 - テーブルおよび追加レコード・リスト、およびそれらの項目
 - 呼び出されるパラメーター・リスト内のレコードとマップ、およびそれらの項目。
 - プログラム内の他の関数の入出力オブジェクト、およびそれらの項目。
 - このカテゴリー内で名前が固有でない場合は、その名前を修飾する必要があります。
 - プログラムで暗黙データ項目が許可されている場合、VisualAge Generator は使用方法に基づいて暗黙定義を作成します。

CALL ステートメントまたは関数呼び出しの場合、レコードはフィールドよりも優先される傾向があり (これは、レコード名を修飾できないからです)、レベル 77 項目は他のフィールドよりも優先される傾向があります (これは、かつて CALL ステートメントで使用できたのは、レベル 77 項目のみであったからです)。ただし、VisualAge Generator 4.5 フィックスバック 5 を使用してテストを行ったところ、CALL ステートメントまたは関数呼び出しに対しては一貫性のあるパターンは判別されませんでした。

EGL のネーム解決規則および名前の修飾規則

EGL でのネーム解決規則は、EGL が以下のように機能拡張されているため、VisualAge Generator の規則とは異なっています。

- 異なるサブストラクチャー下にあるレコードには、同じフィールド名を複数回含めることができます。複数レベルの修飾がサポートされており、場合によってはそれが必須になっています。修飾子にすることができるのは、レコード内のフィールドです (例えば、MYRECORD.SHIPPING.ADDRESS または単に SHIPPING.ADDRESS)。
- 関数は複数の入出力ステートメントを持つことができるため、複数の入出力オブジェクトを持つことができます。
- show** ステートメント (マップまたは UI レコードを指定した VAGen XFER) は、入出力ステートメントと見なされます。
- フィールド (項目変数) をプログラム・レベルで宣言できます。
- EGL にはレベル 77 項目はありません。マイグレーション・ツールは、レベル 77 項目を、プログラム内で宣言された別個の基本レコードに含めます。
- ライブラリーなどの新規の EGL パーツ。
- EGL のキーワード **this**。このキーワードを使用すると、プログラム・レベルで宣言されたレコード変数を使用すること、またはこのセクションで後述するカテゴリー EGL1 および EGL2 の名前ではなく、プログラムの **use** 書式ステートメントで指定された書式を使用することを指示できます。

EGL が名前を見つけると、以下のネーム解決規則および名前の修飾規則を使用して、フィールドが配置されている場所が判別されます。

- プログラムが **allowUnqualifiedItemReferences** プロパティを yes に設定していない場合、すべてのフィールド参照を修飾する必要があります。フィールドがサブストラクチャー内にある場合は、そのフィールドを完全修飾する必要があります。例えば、**allowUnqualifiedItemReferences** が yes に設定されておらず、CUSTOMER_RECORD に LASTNAME の親である NAME フィールドが含まれている場合、CUSTOMER_RECORD.NAME.LASTNAME は有効ですが、CUSTOMER_RECORD.LASTNAME は無効です。
- 修飾子にできるのは、レコード、書式、DataTable、または構造化レコードあるいは DataTable 内の別のフィールドです。複数のレベルの修飾が許可されており、場合によってはそれが必須になっています。
- フィールドが修飾されていない場合、または部分的に修飾されている場合は、EGL はそのフィールドをチェックして、以下の順序でその修飾子を判別します。
 - カテゴリー EGL1 - 関数レベルで宣言されている以下の変数:
 - この関数のローカル・ストレージおよびパラメーター・リストで宣言されている項目変数名。
 - この関数のローカル・ストレージおよびパラメーター・リストで宣言されているレコード変数名。
 - このカテゴリー内では名前は固有である必要があります。
 - カテゴリー EGL2 - この関数に固有の、以下の入出力オブジェクトおよびその他のフィールド:

- 入出力オブジェクトおよびそのフィールド。EGL では、関数に複数の入出力ステートメントを含めることができます。さらに EGL では、**show** ステートメント (マップが指定された VAGen XFER、または UI レコードが指定された XFER) は入出力ステートメントと見なされます。
- 関数パラメーター・リストのレコード変数内のフィールド。
- 関数のローカル・ストレージ・リストのレコード変数内のフィールド。
- このカテゴリ内で名前が固有でない場合は、その名前を修飾する必要があります。
- カテゴリ EGL3 - プログラム・レベルで宣言される以下の変数:
 - プログラムのレコード宣言からのレコード変数名。レコード宣言リストには、**inputRecord** プロパティによって指定されたレコード、またはプログラムの任意の関数内の入出力ステートメントで使用されるレコードが含まれます。
 - プログラム・レベルで宣言された項目変数名。
 - プログラムのパラメーター・リストにリストされる項目変数名およびレコード変数名。
 - このカテゴリ内で名前が固有でない場合は、その名前を修飾する必要があります。
- カテゴリ EGL4 - プログラムの、**use** 書式ステートメントから得られる書式名。このリストには、**inputForm** プロパティまたはプログラムのパラメーター・リストによって指定されている書式、またはプログラムの関数内の入出力ステートメントで使用される書式が含まれます。**use** 書式ステートメントが **FormGroup** のみを指定している場合は、その **FormGroup** 内のすべての書式が考慮されます。
- カテゴリ EGL5 - プログラムの **use** 宣言で指定されている **DataTable** 名。
- カテゴリ EGL6 - プログラム内の任意の場所で使用される以下のフィールド:
 - カテゴリ EGL3 から EGL5 までのレコード変数、書式、および **DataTable** のフィールド。
 - **use** 書式ステートメントが **FormGroup** のみを指定している場合は、その **FormGroup** 内のすべての書式のすべてのフィールドが考慮されます。
 - このカテゴリ内の名前が固有でない場合は、その名前を修飾する必要があります。
- カテゴリ EGL7 - プログラムの **use** 宣言で指定されたユーザー・ライブラリー内のフィールド。(EGL ライブラリーには何もマイグレーションされないため、このカテゴリはマイグレーションされた VAGen プログラムには影響しません。)
- カテゴリ EGL8 - システム・ライブラリー内のフィールド。(マイグレーション・ツールは常に、VAGen EZE ワードからマイグレーションされたフィールドを EGL システム・ライブラリー名で修飾します。)
- 暗黙定義は EGL では許可されていません。

EGL のネーム解決規則および名前の修飾規則は整合しています。つまり、それらの規則は、呼び出されるステートメントまたは関数のタイプには影響されません。

ネーム解決規則および名前の修飾規則の違いによる検証メッセージ

ほとんどの場合、VisualAge Generator と EGL のネーム解決規則および名前の修飾規則は問題を起こしません。ただし、レコード、書式、または DataTable 内のフィールドが、別のレコード、書式、または DataTable と同じ名前になっている場合は、VisualAge Generator と EGL は異なる方法で名前を解決することがあります。このため、以下の状態が発生する可能性があります。

- 通常の EGL 検証によって検出される無効な EGL ステートメント (例えば、書式のフィールドではなく、レコードに色を設定しようとした場合)。これらの場合、EGL 検証は「問題」ビューにエラー・メッセージを表示します。
- VisualAge Generator 互換性モードを使用している場合に検証によって作成される特殊な EGL 警告メッセージ。これらの警告メッセージは、VisualAge Generator と EGL の間でネーム解決が異なっている可能性がある、ということを示しています。
- VisualAge Generator では CALL ステートメントまたは関数呼び出しについて優先されていた、レベル 77 項目または呼び出されたパラメーターが存在していて、EGL ではその項目またはパラメーターの解決結果が異なる場合、EGL メッセージは発行されません。

以下に、同じ名前を持つものが存在している場合に、VAGen と EGL の間でネーム解決が変化する可能性のある状態の例を示します。

例 1

解決は、あるレコードのフィールドから別のレコードへと変化します。

```
ProgramA:
Tables and Additional Records List
  RECORDA - a record that contains field RECORDZ defined as CHAR
  RECORDZ - a record that contains other fields
FunctionA:
  MOVE "abc" to RECORDZ;
/* VAGen-resolves to field RECORDA.RECORDZ,      */
/* based on statement context;                    */
/* record is invalid if source is a literal */
/* EGL -resolves to record RECORDZ,              */
/* based on Category EGL3; a record is valid */
/* -message IWN.VAL.6621.w is issued              */
```

例 2

解決は、**show** ステートメントで使用される書式の名前によって、オリジナルの入出力オブジェクト内のフィールドから未確定状態へと変化します。

```
ProgramB:
FORMF is a form in the program FormGroup for the program
FunctionB:
  INQUIRY RECORDB /* RECORDB contains field FORMF defined as BIN */
  XFER PGMBX , FORMF; /* converts to EGL show statement */
  FORMF = 123; /* VAGen-resolves to field RECORDB.FORMF */
/* based on statement context; */
/* form is invalid if source is a literal */
/* EGL -cannot resolve between field or form FORMF; */
/* based on Category EGL2; form is not valid */
/* -FORMF is now an I/O object for show */
/* -a form is invalid; */
/* -message IWN.VAL.6620.e is issued */
```


例 3

解決は、レコードから、**show** ステートメントで使用される書式のフィールドへと変化します。

ProgramC:

```
FORMF is a form in the FormGroup for the program and contains
field RECORDZ Tables and Additional Records List
RECORDZ - a record that contains some fields
FunctionC:
  XFER PGMCY , FORMF; /* converts to EGL show statement          */
  CALL PGMCX RECORDZ; /* VAGen-resolves to record RECORDZ        */
                        /* based on statement context;            */
                        /* record receives precedence on a CALL    */
                        /* EGL -resolves to field FORMF.RECORDZ;    */
                        /* based on Category EGL2                  */
                        /* -FORMF is now an I/O object for show     */
                        /* -message IWN.VAL.6571.w is issued       */
```

例 4

解決は、**show** ステートメントで使用される書式のフィールドによって、オリジナルの入出力オブジェクト内のフィールドから未確定状態へと変化します。

ProgramD:

```
FORMF is a form in the FormGroup for the program and
contains a field named ITEMDD defined as BIN
FunctionD:
  INQUIRY RECORDD /* RECORDD contains field ITEMDD defined as BIN */
  XFER PGMDX , FORMF; /* converts to EGL show statement          */
  ITEMDD = 123; /* VAGen-resolves to field RECORDD.ITEMDD          */
                /* based on Category VG2                        */
                /* - RECORDD is I/O object                      */
                /* - FORMF is not an I/O object                  */
                /* EGL -cannot resolve between 2 fields ITEMDD   */
                /* based on Category EGL2                        */
                /* -FORMF is now an I/O object for show         */
                /* -message IWN.VAL.6620.e is issued            */
```

付録 F. VisualAge Generator に必要な APAR

現在では、VisualAge Generator Developer V4.5 フィックスパック 5 に含まれている以外には、必要な既知の APAR はありません。

付録 G. マイグレーション・データベース

DB2 マイグレーション・データベースの作成

注記のない限り、次の説明は、Java または Smalltalk のどちらからマイグレーションする場合にも関係なく適用されます。

マイグレーション・データベースは、DB2 バージョン 8.1 (フィックスパック 15) または DB2 バージョン 8.2 (フィックスパック 8) を必要とします。上記の 2 つのフィックスパックのレベルは同等です。DB2 バージョン 9.x は、Smalltalk でのステージ 1、およびステージ 2 および 3 でサポートされています。DB2 バージョン 9.x は、Java でのステージ 1 ではサポートされていません。

Windows XP 上での DB2 の使用

マイグレーション・ツールには以下の要件があります。

- マイグレーション・データベースへのアクセスに使用されるユーザー ID は、ブランクを含んでいてはなりません。
- ステージ 2 および 3 のマイグレーション・ツール・ウィザード内でマイグレーション・セットが表示されるようにするには、Windows のユーザー ID が限定権限でなく管理者権限を持っている必要があります。

DB2 権限要件

マイグレーション・ツールを実行するときに使用する DB2 ユーザー ID には、マイグレーションのすべての表、ビュー、および索引に対する、選択、挿入、更新、削除、および制御権限を付与する必要があります。制御権限は、残りのマイグレーション・ステップのパフォーマンスを向上するためにステージ 1 で発行される RUNSTATS コマンドによって必要とされます。

マイグレーション・データベースの作成

DB2 の該当するレベルをインストールするかそのレベルにアップグレードした後で、以下のステップに従ってマイグレーション・データベースを作成します。

1. DB2、および DB2 を使用するアプリケーションすべてがシャットダウンされていることを確認する。例えば、VisualAge Generator および EGL 開発環境をシャットダウンします。
2. DB2 コマンド・ウィンドウを開く。
 - Java からマイグレーションを行う場合は、*VisualAgeJava-installation-directory¥ide¥vgmigration* ディレクトリーにナビゲートする。
 - Smalltalk からマイグレーションを行う場合は、*VisualAge-Smalltalk-installation-directory* にナビゲートする。
3. ファイル SetupDatabase.bat を実行する。このファイルは、同じディレクトリーにあるファイル createdatabase.sql を実行し、同じディレクトリー内のファイル createdatabase.out に出力を保管します。このファイルは、VGMIG という名前の DB2 データベースを作成し、データベースに接続して、データベース・パラメ

ーターを構成します。データベースの作成には長くて 1 分かかることがあります。必ず、すべてのコマンドの実行が終了するまで待ってください。

注:

- コンソールに最初に表示されるコマンドが実行された結果、エラー・メッセージが出る場合があります。このメッセージは無視できます。このメッセージは、単に VGMIG データベースがまだ存在しないことを示しています。
 - VGMIG 以外の名前を指定してデータベースを作成する場合は、createdatabase.sql 内で VGMIG が出現する箇所すべてを、目的のデータベース名に変更する必要があります。また、ステージ 1 から 3 のマイグレーション・ツール設定の中でも、VGMIG を忘れずに変更する必要があります。
 - デフォルトでは、VGMIG データベースはパスワードによって保護されていません。パスワード保護が必要な場合は、データベースがパスワードによって保護されるように変更する必要があります。
4. ファイル SetupTables.bat を実行する。このファイルは、同じディレクトリーにあるファイル createtables.sql を実行し、同じディレクトリー内のファイル createtables.out に出力を保管します。このファイルは、マイグレーション・ツールが必要とするすべての表とビューをマイグレーション・データベース内で作成します。表は、MIGSCHEMA という名前の高位修飾子 (スキーマ) を使用して作成されます。データベースの作成には長くて 1 分かかることがあります。必ず、すべてのコマンドの実行が終了するまで待ってください。

注:

- コンソールに最初に表示されるコマンドが実行された結果、エラー・メッセージが出る場合があります。これらのメッセージは無視できます。これらのメッセージは、単に表とビューがまだ存在しないことを示しています。
 - MIGSCHEMA 以外の名前を指定してスキーマを作成する場合は、createtables.sql 内で MIGSCHEMA が出現する箇所すべてを、目的のスキーマ名に変更する必要があります。ステージ 1 から 3 までのマイグレーション設定で、MIGSCHEMA を忘れずに変更する必要があります。
 - マイグレーション・データベースを完全に消去する必要がある場合は、SetupTables.bat ファイルを DB2 コマンド・ウィンドウから再実行できます。
5. DB2 コマンド・ウィンドウを閉じる。

この時点で、マイグレーション・データベース、スキーマ、表、およびビューが作成されました。ここで、ステージ 1 マイグレーション・ツールが使用する設定ファイルを作成できます。Java からマイグレーションを行う場合は、151 ページの『ステージ 1 設定の実行』を参照してください。Smalltalk からマイグレーションを行う場合は、179 ページの『ステージ 1 設定の実行』を参照してください。

ステージ 1 のマイグレーション・データベースのリセット

マイグレーション・データベースをリセットする必要がある場合は (例えば、名前変更規則を変更したために)、次のいずれかの手法を使用します。

- マイグレーション・データベース内の表をすべて削除して再作成するツールを使用する。

このツールは、次の状況で使用します。

- マイグレーション・プランをすべて削除する必要がある場合。
- Java プロジェクトの複数のバージョンをマイグレーションした場合。
- Smalltalk 構成マップの複数のバージョンをマイグレーションした場合。

表をすべて削除して再作成するツールを実行するには、以下のステップに従います。

1. DB2 コマンド・ウィンドウから、SetupTables.bat があるディレクトリーにナビゲートする。
 - Java の場合、このディレクトリーはご使用の *VisualAge-for-Java-install-directory¥ide¥vmigration* です。
 - Smalltalk の場合、このディレクトリーはご使用の *VisualAge-Smalltalk-install-directory* です。
 2. SetupTables.bat を実行する。
- 指定したマイグレーション・セットを削除するツールを使用する。少数のマイグレーション・セットのみを削除する必要がある場合は、このツールを使用します。指定したマイグレーション・セットを削除するツールを実行するには、以下のステップに従います。
 1. 次のステップに従って、マイグレーション・データベースから削除する必要があるマイグレーション・セット ID を特定する。
 - a. DB2 コントロール・センターまたは SQL 照会を使用して、CONFIGPLAN 表を検索する。
 - b. 削除する CONFIGPLANNAME を見つける。
 - c. 指定する必要があるマイグレーション・セット ID は、対応する CONFIGPLANID 列にあります。
 2. DB2 コマンド・ウィンドウから、deletemigsets.bat があるディレクトリーにナビゲートする。
 - Java の場合、このディレクトリーはご使用の *VisualAge-for-Java-install-directory¥ide¥vmigration* です。
 - Smalltalk の場合、このディレクトリーはご使用の *VisualAge-Smalltalk-install-directory* です。
 3. 次のいずれかの形式を使用して、ファイル deletemigsets.bat を実行する。
 - ただ 1 つのマイグレーション・セットを削除する場合は、次の形式を使用する。
`deletemigsets n`

ここで、*n* は削除するマイグレーション・セット ID に対応する CONFIGPLANID 列内の値です。

- 複数のマイグレーション・セットを削除する場合は、次の形式を使用する。

```
deletemigsets "n1,n2"
```

ここで、*n1* と *n2* は削除するマイグレーション・セット ID です。

DB2 を使用したリモート・データベースのカタログ

マイグレーション・ツールでは、ローカルの DB2 データベースを使用するとパフォーマンスが向上します。しかし、リモート・データベースを使用する場合は、このセクションの情報が役立ちます。DB2 上でリモート・データベースをカタログするには、次の情報が必要です。

- データベースがあるリモート・マシンのホスト名または IP アドレス
- クライアントのポート番号とプロトコル (例: 60000/tcp)
- ノード名 (リモート・マシンを記述する別名。例: db2node)
- データベース名
- データベース別名 (オプション)

DB2 を使用してリモート・データベースへの TCP/IP 接続を確立するには、以下のステップに従います。

1. Windows で、DB2 のコマンド・プロンプト・ウィンドウを起動する。
2. ノードをカタログするには、次のコマンドをすべて 1 行に入力する。

```
db2 catalog tcpip node nodeName remote [ hostName | ipAddress ]  
server [ svcname | portNumber ]
```

hostName または *ipAddress* のいずれを入力することもできます。例えば、ノード db2node 上にある、IP アドレス 9.10.11.123 をもち、ポート番号 60000 を使用するリモート・サーバーをカタログするには、次のコマンドを入力します。

```
db2 catalog tcpip node db2node remote 9.10.11.123 server 60000
```

3. データベースをカタログするには、次のコマンドをすべて 1 行に入力する。

```
db2 catalog database databaseName  
[ as databaseAlias ] at node nodeName
```

「as *databaseAlias*」はオプションです。 *databaseAlias* を指定しない場合、別名はデータベース名と同じになります。 *nodeName* は、ステップ 2 で使用した *nodeName* と同じであることが必要です。

例えば、ノード db2node 上での別名が sam1 になるように SAMPLE という名前のリモート・データベースをカタログするには、次のコマンドを入力します。

```
db2 catalog database sample as sam1 at node db2node
```

4. データベースへの接続をテストするには、次のコマンドをすべて 1 行に入力する。

```
db2 connect to databaseAlias use userName using password
```

ステップ 3 で別名 (*databaseAlias*) を指定しなかった場合は、データベース名を使用します。例えば、パスワード db2password を使用するユーザー db2user の別名 sam1 を使用して、データベース SAMPLE に接続するには、次のコマンドを入力します。

```
db2 connect to sam1 user db2user using db2password
```

ステップ 2 でデータベース別名として `sam1` を指定しなかった場合は、次のコマンドを入力します。

```
db2 connect to SAMPLE user db2user using db2password
```

5. データベース接続情報が表示されます。

その他の支援情報については、次の Web サイトをご覧ください。

<https://aurora.vcu.edu/db2help/db2i4/frame3.htm#idx>

DB2 を使用したリモート・データベースのアンカタログ

DB2 上でリモート・データベースをアンカタログするには、次の情報が必要です。

- データベース別名、またはデータベース名 (データベースをカタログしたときに別名を指定しなかった場合)。

DB2 を使用してリモート・データベースをアンカタログするには、以下のステップに従います。

1. Windows で、DB2 のコマンド・プロンプト・ウィンドウを起動する。
2. データベースをアンカタログするには、次のコマンドを入力する (ここで、*databaseAlias* はデータベース別名)。

```
db2 uncatalog database databaseAlias
```

例えば、データベース `SAMPLE` (別名 `sam1` が指定されている) をアンカタログするには、次のコマンドを入力します。

```
db2 uncatalog database sam1
```

データベースをカタログしたときにデータベース別名を指定しなかった場合は、データベースの名前を使用します。例えば、`SAMPLE` データベースのデータベース別名を指定しなかった場合は、以下のコマンドを入力します。

```
db2 uncatalog database SAMPLE
```

その他の支援情報については、次の Web サイトをご覧ください。

<https://aurora.vcu.edu/db2help/db2i4/frame3.htm#idx>

便利な照会

サンプルのステージ 1 マイグレーション・ツールを変更した場合、またはユーザー独自のステージ 1 マイグレーション・ツールを開発した場合は、次の SQL 照会が変更内容の検証に役立つことがあります。

注:

- これらの例は、DB2 コマンド・ウィンドウから実行できます。
- これらの例は、デフォルトのマイグレーション・データベース名 (VGMIG) とデフォルト・スキーマ (MIGSCHEMA) の使用を前提としています。
- 特に注記のない限り、DB2 コマンド全体を 1 行に入力する必要があります。本書で後述するコマンドは、スペースの制約によって複数の行に示されている場合があります。

- これらの例を使用するには、まずデータベースに接続する必要があります。
データベースに接続するには、次のコマンドを実行します。

```
db2 connect to VGMIG
```

ステージ 1 マイグレーション・ツールが正しく実行されたかどうか判別するための支援情報を得るには、Java の場合は *VAGen-installation¥ide¥vgmigration* ディレクトリー、Smalltalk の場合はご使用の *VAGen-installation* ディレクトリーにある、次に示す .bat ファイルを実行します。

```
checkStage1.bat
```

この .bat ファイルは、次のようにいくつかの照会を実行します。

- データベース内のマイグレーション・プラン名のリスト
- データベース内のパーツの総数。この数には、ステージ 1 のマイグレーション・ツールによって作成されたすべてのダミー・マップ・グループ・パーツが含まれています。
- 外部ソース形式の妥当性、および EGL ファイルへのパーツの配置を検査するその他の照会

最初の 2 つの照会結果は 0 より多く、その他の照会結果は 0 であることが必要です。結果が異なる場合は、ステージ 1 マイグレーション時に問題が発生しており、IBM サポートに連絡する必要があります。

VAGen パーツがマイグレーションされたかどうか判別するには、次のコマンドを実行します。

```
db2 select configplanname, configplanversion, vgpartname, vgparttime, is_migrated
from migschema.vgpart where vgpartname = 'yourPartName'
```

マイグレーション・データベース内にあるパーツすべての外部ソース形式の先頭数文字を検査するには、次のコマンドを実行します。

```
db2 select vgpartname, substr(vgesfsorce,1,n) from migschema.vgpart
```

n は、表示する文字の数を指定します。

マイグレーション・データベース内のパーツが、有効な外部ソース形式タグで始まっていないことを判別するには、次のコマンドを実行します。

```
db2 select vgpartname, substr(vgesfsorce,1,n) from migschema.vgpart
where vgesfsorce not like ':%'
```

n は、表示する文字の数を指定します。

マイグレーション・データベース内のパーツ数の判別

マイグレーション・データベース内のパーツの総数を判別するには、以下のようにします。

```
db2 select count(*) from migschema.vgpart
```

マイグレーション・データベース内の特定パーツ型のパーツ数を判別するには、以下のようにします。

```
db2 select count(*) from migschema.vgpart where vgparttype = nnnnnnn
```

ここで、*nnnnnnn* は、以下の表に示されるようにパーツ型に基づいて変化します。

表 166. パーツ型の値

パーツ型	vgparttype の値
Function	65536
リンク・エディット	131072
バインド制御	262144
リソース関連	524288
リンケージ・テーブル	1048576
生成オプション	2097152
テーブル	8388608
レコード	33554432
PSB	67108864
マップ	268435456
マップ・グループ	536870912
データ項目	1073741824
Program	2147483648

EGL ファイル名の検討

ステージ 1 からのレポートを検討する場合の代替方法または補足として、ステージ 1 のマイグレーション・ツールによって生成された EGL ディレクトリーおよびファイル名をすべてリストできます。これを簡単に実行するには、以下の行に示すような DB2 コマンドが含まれた .bat ファイルを作成します。

```
db2 connect to VGMIG
db2 set schema migschema
db2 select distinct('ListOfEGLFileNames ') from configplan
>>%1\DBResults%2.txt
db2 select distinct(substr(eglfilename, 1, 130)) from eglfile order by 1
>>%1\DBResults%2.txt
db2 select distinct('ListOfVAGenPartsAndEGLFileNames') from configplan
>>%1\DBResults%2.txt
db2 select substr(vgpartname, 1, 32), substr(eglfilename,1,130)
from vgpart_lineage order by 1, 2
>>%1\DBResults%2.txt
db2 disconnect VGMIG
```

注:

1. 最初および最後のコマンドで、マイグレーション・データベースに接続し、切断します。
2. **db2 set** コマンドではスキーマ名を設定し、.bat ファイルのその他の部分で表名を限定しないで済むようにします。
3. **db2 select distinct('ListOfEGLFileNames ')** および **db2 select distinct('ListOfVAGenPartsAndEGLFileNames')** の各コマンドでは、単に出カレポートの区切り文字を指定しています。
4. **db2 select distinct(substr(eglfilename, 1, 130))** コマンドは、すべての EGL ファイル名をファイル名でソートしたリストを生成します。 *eglfilename* には EGL プロジェクト、パッケージ、およびファイル名が含まれるため、この照会で

EGL ワークスペースの編成の概要を調べることができます。必要であれば、完全なファイル名を格納するように値 **130** を変更できます。デフォルトの `eglfile` テーブルの最大長は 512 文字です。

5. **db2 select substr(vgpartname, 1, 32), substr(eglfilename,1,130)** コマンドは、すべての VAGen パーツおよびそれに対応する EGL ファイルをパーツ名およびファイル名でソートしたリストを生成します。この照会には、ステージ 1 のマイグレーション・ツールによって作成されるすべてのダミー・マップ・グループが含まれています。ご使用の命名規則によっては、将来の EGL 開発のためにパーツが規則正しく配置されるかどうかを判別するためにこの照会が役立つ場合があります。前の照会と同様に、必要であれば、完全なファイル名を格納するように値 130 を変更できます。また、パーツ型を含める (およびソートする) ようにこの照会を変更することもできます。VAGen パーツ型と、マイグレーション・データベースに格納される数値パーツ型との間の対応について詳しくは、553 ページの表 166を参照してください。
6. 照会の結果は、以下を使用してファイルにパイプ出力されます。

```
>>%1\DBResults%2.txt
```

ここで

- %1 は .bat ファイルの第 1 パラメーターであり、出力ファイルを配置するドライブおよびディレクトリーの名前を指定します。
- %2 は .bat ファイルの第 2 パラメーターであり、ファイル名に意味のある接尾部を指定して、ステージ 1 マイグレーション・ツールの異なる実行からの出力を区別できるようにします。

.bat ファイルを実行するには、以下のステップを実行します。

1. DB2 コマンド・プロンプト・ウィンドウを開く。
2. 2 つのパラメーターを指定して .bat ファイルを実行する。例えば、次のようになります。

```
reportStage1Results.bat c:\myTestMigration\stage1 _TrialA
```

結果は以下のファイルに入ります。

```
c:\myTestMigration\stage1\DBResults_TrialA.txt
```

特定のエラー・メッセージを支援する照会

以下のセクションの各クエリーは、特定のエラーや警告メッセージを解決する場合に役立ちます。

IWN.MIG.0302.w

このメッセージは、表の内容が単一行のみである場合に発生します。このような表は、MOVEA ステートメントのソースに使用される場合に問題を起こすことがあります。限定子として表名を使用せずに表のフィールドが MOVEA ステートメントのソースとして使用される MOVEA ステートメントを見つけるには、以下の照会を使用します。


```
db2 connect to VGMIG
db2 set schema migschema
db2 select vgpdbname, egldbname from vgpdbname
  where vgpdbname in (65536, 2147483648)
  and vgesfsname like '%MOVEA %fieldName%'
>>message302Results.txt
```

fieldName は、表のフィールドです。表の各フィールドごとにこの照会を繰り返す必要があります。また、必要に応じて MOVEA の後に続く空白の数を変化させることもできます。

ステージ 2 のマイグレーション・データベースのリセット

ステージ 1 を再実行しないでマイグレーションのステージ 2 と 3 を再実行する場合に、マイグレーション・データベース内のパーツをすべてリセットするには、次の DB2 コマンドを実行します。

```
db2 update migschema.vgpdbname set is_migrated = 'N', egldbname = NULL,
  egldbname = NULL
db2 delete from migschema.translation_msgs
```

マイグレーション・データベースのバックアップおよび復元

マイグレーション・データベースをバックアップするには、次のコマンドを実行します。

```
db2 backup database vgmig to x:\mybackups\backupName
```

x:\mybackups\backupName は、バックアップを配置するドライブとディレクトリーです。複数のサブディレクトリーが *x:\mybackups\backupName* の下に作成されます。

前にバックアップしたマイグレーション・データベースを復元するには、次のコマンドを実行します。

```
db2 restore database vgmig from x:\mybackups\backupName REPLACE EXISTING
```

x:\mybackups\backupName は、バックアップを配置するドライブとディレクトリーです。

付録 H. マイグレーション・ツールのパフォーマンス

以下の要因がマイグレーション・ツールのパフォーマンスに影響することがあります。

- ステージ 1、2、および 3 の全般的なパフォーマンス。
 - メモリー。
 - プロセッサ速度。
 - ローカルまたはリモートの DB2 データベース。リモート・データベースの場合は、ネットワーク接続の速度が重要です。
 - Java プロジェクトとパッケージの数、または Smalltalk 構成マップとアプリケーションの数。
 - パーツの数、およびパーツ型別の分布。
 - マイグレーション・セットの数。
 - 演算部の行数。
- ステージ 1 のパフォーマンス。
 - マイグレーションを開始する前に Java ワークスペースまたは Smalltalk イメージが新規のものであること。
 - ローカルまたはリモートの Java リポジトリ、または Smalltalk ライブラリー。リモート・リポジトリまたはリモート・ライブラリーの場合は、ネットワーク接続の速度が重要です。マイグレーションを実行するマシンにリポジトリをコピーすることにより、処理速度が向上したお客様の例があります。
 - 名前変更規則の複雑さ。
 - マイグレーション・データベースにマイグレーション・セットがすでに存在するかどうか。別の名前変更規則を使用してマイグレーション・セットを再作成する場合は、ステージ 1 マイグレーション・ツールを使用してオリジナルのマイグレーション・セットを消去するよりも、`setuptables.bat` を実行して SQL 表を再作成する方が効率的です。SQL 表の再作成は、マイグレーション・データベースに他のマイグレーション・セットがない場合に限り有効です。
- ステージ 2 および 3 のパフォーマンス。
 - マイグレーション・オプション。

数多くの要因が関係するため、マイグレーションの実行時間を予測できる具体的な公式は存在しませんが、次に示すセクションでは、マイグレーションのさまざまなステージの所要時間を予測するためのガイダンスを示します。

- プロジェクト、パッケージ、パーツ、およびプログラムの数
- マイグレーション・セットおよびその他のマイグレーション・オプションの数。
- プロセッサ速度
- 演算部の行数
- ステージ 1 の新規 Java ワークスペース

さらに、ディスク・スペース要件の計画に役立つ情報が記載されたセクションがあります。

プロジェクト、パッケージ、パーツ、およびプログラムの数

次の表に、マイグレーションのさまざまなステージにかかる時間についての情報を示します。これらのテストは、2.0 GB のメモリー、および 2.1 ギガヘルツのプロセッサ速度の環境で、Windows XP を使用して実行されました。測定は、EGL 7.1 を対象としたものであり、時間はすべて分単位で表記されています。

表 167. マイグレーション時間に対するマイグレーション・セットのサイズの影響

テスト・ケース	プロジェクトの数	パッケージの数	パーツの数	プログラムの数	ステージ 1 の時間	ステージ 2 の時間	ステージ 3 のファイル書き込みにかかった時間	ステージ 3 の更新またはビルドにかかった時間
1	4	93	10,614	25	14	1	1	2
2	478	516	11,350	163	21	3	2	16
3	3	44	12,083	249	24	3	1	6
4	6	118	15,281	71	40	5	2	3
5	7	8	16,800	107	224	5	1	5
6	11	226	19,453	225	87	9	4	8
7	1	99	20,486	1,246	21	3	3	10
8	52	1,592	48,323	2,191	155	9	9	37

表 167でのテスト・ケースに関する詳細を以下に示します。

- テスト・ケース 5 のステージ 1 では Smalltalk でテスト・ケース 1 を使用、その他のテスト・ケースでは Java でステージ 1 を使用。
- テスト・ケース 6 ではホワイト・ペーパーの手法を使用してパッケージを統合。また、組み込みのカスタマイズを使用してパーツ型およびパーツ名別に共通ファイルを分割。
- テスト・ケース 8 では組み込みのカスタマイズを使用してパーツ型別に共通ファイルを分割。

次に、表 167 に基づく全般的な観察結果を示します。

- テスト・ケース 6 および 7 のパーツ数は類似していますが、テスト・ケース 6 ではステージ 1 の実行よりも 4 倍、ステージ 2 の実行よりも 3 倍長い時間がかかっています。テスト・ケース 7 にはより多くのプログラムがありますが、それぞれのプログラムは極小であり、関連パーツも非常に少なくなっています。反対に、テスト・ケース 6 のプログラム数は比較的少ないですが、それぞれのプログラムは非常に複雑であり、多くの関連パーツがあります。プログラム数およびその関連パーツ数は、以下のようにパフォーマンスに影響します。
 - ステージ 1 では、すべての関連付けが分析されて、各パーツをプログラムに配置するか共通パーツ・ファイルに配置するかが判別されます。

- ステージ 2 では、パーツが EGL に変換されるのは 1 回のみですが、それでも関連として使用される各プログラムごとにパーツが検討される必要があります。例えば、関数の I/O オブジェクトは、その関数を使用される各プログラムのレコード宣言または **use** 書式ステートメントに追加する必要があります。
- テスト・ケース 8 には、テスト・ケース 5 の約 3 倍の数のパーツ、および約 20 倍の数のプログラムがあります。テスト・ケース 5 内の各プログラムは非常に複雑であるため、テスト・ケース 5 にはテスト・ケース 8 より約 30% 多い関連付けがあります。これは、テスト・ケース 5 のステージ 1 の実行時間が長くなる原因となっています。さらに、Smalltalk でのステージ 1 は Java でのステージ 1 よりも少し遅くなると思われますが、Smalltalk にあるのは 1 つの大規模なテスト・ケースのみであるため、Smalltalk と Java とのパフォーマンスの比較は非現実的です。
- テスト・ケース 8 にはテスト・ケース 7 の 2 倍以上の数のパーツがありますが、ワークスペースのビルドには 4 倍近く長い時間がかかります。テスト・ケース 7 にあるプロジェクトは 1 つのみであるため、反復依存性はありません。反対に、テスト・ケース 8 には 52 のプロジェクト間に多数の反復依存性があります。ステージ 3 のマイグレーション・ツールでは自動的に EGL プロジェクトのビルド順序を最適化するツールが起動されますが、ビルドではその反復依存性の解決に時間がかかります。

558 ページの表 167 の情報に基づき、数百あるいは 1000 のプログラムがある場合には、これらのプログラムが複数のサブシステムにわたっていても、これらを単一のマイグレーション・セットとしてマイグレーションすることをお勧めします。このためには、すべてのサブシステムが共通プロジェクトの同じバージョンを使用していて、サブシステムに重複するパーツ名がないことが前提になります。

マイグレーション・セットおよびその他のマイグレーション・オプションの数

560 ページの表 168 に、より少ない数のマイグレーション・セットに統合した場合の影響を示します。また、ステージ 2 の「残りの VAGen パーツをマイグレーションする」オプション、およびステージ 3 の「既存ファイルを上書き」オプションの影響も示しています。表には、3 つの異なる技法を使用してマイグレーションされた VAGen プロジェクトの同じセットが示されています。10 のプロジェクト、226 のパッケージ、19453 のパーツ、および 225 のプログラムがあります。2 つの共通プロジェクトには、7734 のパーツと 41 のプログラムが含まれています。これは、パーツの 40% に相当します。8 つのマイグレーション・セットは、それぞれ 1 つのサブシステムを表し、2 つの共通プロジェクトが含まれています。8 つのサブシステムには重複するパーツはありません。したがって、テスト・ケース 6C に示すように、すべてのサブシステムを単一のマイグレーション・セットとしてマイグレーションできます。測定は EGL 7.1 を対象としたものであり、時間はすべて分単位で表記されています。

表 168. マイグレーション・セットおよびマイグレーション・オプションの数の影響

テスト・ケース	マイグレーション・セットの数	残りの VAGen パーツをマイグレーションする	既存ファイルを上書き	ステージ 1 の時間	ステージ 2 の時間	ステージ 3 のファイル書き込みにかかった時間	ステージ 3 の更新またはビルドにかかった時間
6A	8	なし	なし	234	11	5	9
6B	8	あり	あり	234	10	5	7
6C	1	あり	あり	87	9	4	8

次に、表 168 に基づく全般的な観察結果を示します。

- テスト・ケース 6A および 6B は、同じステージ 1 データベースをステージ 2 とステージ 3 の開始点として使用しています。これらの 2 つのテスト・ケースで異なる点は、ステージ 2 とステージ 3 で選択するオプションのみです。
 - ステージ 2 では、オプション「残りの VAGen パーツをマイグレーションする」の選択による影響は軽微です。これは、恐らく、未使用パーツの数が比較的少ない (250) ためであると考えられます。
 - ステージ 3 では、新規にマイグレーションされたファイルを 6A での既存ファイルにマージする時間は、ファイルをすべて再書き込みする時間と違いがありません。オプション「既存ファイルを上書き」をクリアする利点は、さまざまなマイグレーション・セットからの `import` ステートメントがすべて最終の EGL ファイルに含まれることです。
- テスト・ケース 6C により、その他の 10 のプロジェクトを示すハイレベルの PLP パーツを含む、1 つのプロジェクトが追加されます。この手法により、8 つのサブシステムすべてを単一のマイグレーション・セットとしてマイグレーションできるようになります。
 - ステージ 1 では、各マイグレーション・セットに対して、共通パーツをロードして分析する必要がないため、時間が大幅に節約されます。
 - ステージ 2 では、複数のマイグレーション・セットにおけるパーツ間マイグレーションに対して、共通パーツを分析する必要がないため、ある程度節約されます。ただし、マイグレーション・ツールはすべてのパーツを変換するため、ステージ 1 やステージ 3 の場合ほど大幅には節約されません。
 - ステージ 3 では、以下のいくつかの要素の組み合わせにより、少し時間が節約されます。
 - 共通パーツは、**import** ステートメントを特定するために一度のみ分析されます。
 - EGL ファイルは、一度しか書き込まれません。
 - すべてのパーツが同時にマイグレーションされるため、ファイルに関するマージ・ロジックは必要ありません。

プロセッサ速度

表 169 には、マシンのメモリーおよびプロセッサ速度を変更した場合の影響が示されています。テスト・ケース 8A では、1.0 GB のメモリー、および 1.1 ギガヘルツのプロセッサ速度のマシンが使用されています。テスト・ケース 8B では、2.0 GB のメモリー、および 2.1 ギガヘルツのプロセッサ速度のマシンが使用されています。測定は、EGL 7.1 を対象としたものであり、時間はすべて分単位で表記されています。

表 169. プロセッサ速度がマイグレーション時間に及ぼす影響

テスト・ケース	プロジェクトの数	パッケージの数	パーツの数	プログラムの数	ステージ 1 の時間	ステージ 2 の時間	ステージ 3 のファイル書き込みにかかった時間	ステージ 3 の更新またはビルドにかかった時間
8A	52	1,592	48,323	2,191	290	使用不可	使用不可	使用不可
8B	52	1,592	48,323	2,191	155	9	9	37

表 169 に基づき、マイグレーション時にはプロセッサ速度の高いマシンを使用することをお勧めします。

演算部の行数

ある一時期、VisualAge Generator には一連のブランク行が関数に挿入されるという問題がありました。場合によっては、32,000 行ものブランク行が挿入されていました。これらの余分なブランク行は、パフォーマンスに重大な影響を及ぼします。表 170 に、関数の行数がマイグレーション時間に及ぼす影響を示します。このテスト・ケースのプロジェクト数は 2、パッケージ数は 17、パーツ数は 919、プログラム数は 87 です。497 の関数が存在し、そのうち 8 つに大量のブランク行が含まれていました (30,000 行以内と考えられる)。この測定は EGL 5.1.2 を対象としたものであり、1.0 GB のメモリー、および 1.1 ギガヘルツのプロセッサ速度のマシンが使用されています。時間はすべて分単位で表記されています。

表 170. 関数の行数がマイグレーション時間に及ぼす影響

テスト・ケース	VAGen 内でブランク行を除去する前				VAGen 内でブランク行を除去した後			
	ステージ 1 の時間	ステージ 2 の時間	ステージ 3 のファイル書き込みにかかった時間	ステージ 3 のファイルの更新またはビルドにかかった時間	ステージ 1 の時間	ステージ 2 の時間	ステージ 3 のファイル書き込みにかかった時間	ステージ 3 のファイルの更新またはビルドにかかった時間
9	40	25	1	3	12	11	1	3

マイグレーションの開始前に余分なブランク行を除去しただけで、ステージ 1 と 2 の処理時間には劇的な違いが生じます。多数のブランク行がある関数が分かっている場合は、マイグレーション前にブランク行を除去する必要があります。ただし、VisualAge Generator 内でこの問題が発生することはまれであるため、マイグレーションの前にこうした関数を探すことはコスト面で効率的でないことが考えられます。連続するブランク行が 3 行を超える場合、マイグレーション・ツールはステージ 2 マイグレーション時に余分なブランク行を自動的に除去します。このため、ステージ 3 では処理時間に変化はありません。これらのブランク行が除去されるので、EGL ではパフォーマンスが向上します。

ステージ 1 の新規 Java ワークスペース

次の表に、ステージ 1 マイグレーションの開始時に新規ワークスペースを使用した場合の影響を示します。テスト・ケース 13 には、3 つのプロジェクト、29 のパッケージ、2660 のパーツ、および 33 のプログラムが含まれています。テスト・ケース 14 には、マイグレーション・セットのバージョンが 7 つあります。最初のバージョンのプロジェクト数は 3、パッケージ数は 4、パーツ数は 30、プログラムはありません。最後のバージョンのプロジェクト数は 6、パッケージ数は 11、パーツ数は 66、プログラム数は 7 です。この測定は EGL 5.1.2 を対象としたものであり、1.0 GB のメモリー、および 1.1 ギガヘルツのプロセッサ速度のマシンが使用されています。時間はすべて分単位で表記されています。

表 171. 新規 Java ワークスペースがマイグレーションに及ぼす影響

テスト・ケース	新規ワークスペースを使用しない場合のステージ 1 の時間	新規ワークスペースを使用する場合のステージ 1 の時間
10	17	12
11	11	1

表 171 に基づき、VisualAge Java からマイグレーションを行う場合は、新規ワークスペースから始めることを検討してください。新規 Java ワークスペースからの開始方法について詳しくは、166 ページの『パフォーマンスの向上』を参照してください。

VisualAge Smalltalk の場合も、同様に時間が節約される可能性があります。このため、VisualAge Smalltalk からマイグレーションを行う場合も、新規イメージから始めることを検討してください。新規 Smalltalk イメージからの開始方法について詳しくは、194 ページの『パフォーマンスの向上』を参照してください。

ディスク・スペース要件

VisualAge Generator アプリケーションのディスク・スペース要件と、対応する EGL アプリケーションのディスク・スペース要件との間に、直接の関係はありません。

次のテーブルには、558 ページの表 167 で示したものと同一テスト・ケースのディスク・スペース要件が記載されています。EGL 測定が行われたのは、すべてマイグレーション直後であり、「問題」ビューのメッセージの収集や生成を実行する前です。測定は、EGL 7.1 を対象としたものであり、サイズはすべてメガバイト (MB) 単位で表記しています。

表 172. ディスク・スペース要件

テスト・ケース	VAGen .dat ファイル・サイズ	ステージ 1 の DB2 バックアップ・ファイル・サイズ	EGL プロジェクト交換ファイル	EGL ワークスペース・サイズ	EGL .metadata ディレクトリー・サイズ	EGL ワークスペース・サイズの合計
1	11.4	114.7	8.2	40.6	40.0	80.6
2	33.4	163.9	6.3	157.5	41.5	199.0
3	15.3	163.9	9.5	155.7	59.3	215.0
4	13.6	180.3	1.5	86.4	26.6	113.0
5	15.8	426.1	3.6	190.6	38.4	229.0
6	24.2	262.2	3.0	169.3	31.7	201.0
7	26.1	196.7	6.7	147.0	46.0	193.0
8	64.1	393.3	32.5	453.0	156.0	609.0

次に、表 172 に基づく全般的な観察結果を示します。

- VAGen .dat ファイルのサイズは、エクスポートされた VisualAge Java リポジトリ・ファイルのサイズです。このリポジトリ・ファイルには、マイグレーション・セットからの VAGen プロジェクトおよびパッケージのみが含まれています。
- DB2 バックアップ・ファイル・サイズは、DB2 runstats.bat コマンド・ファイルを実行した後の、ステージ 1 終了時のマイグレーション・データベース用バックアップ・ファイルのサイズです。ステージ 2 のサイズは、データベースに EGL ソース・コードを追加したため、さらに大きくなっています。
- テスト・ケース 1、3、および 8 にはすべて、VAGen Web トランザクション・プログラムおよび UI レコードがあります。マイグレーション・ツールは、VAGen Web トランザクションまたは UI レコードを含むテスト・ケース内の各プロジェクトに対して EGL Web プロジェクトを作成します。EGL Web プロジェクトには次のファイルが含まれています。
 - csogw.properties および gw.properties ファイルなどの標準 .properties ファイル
 - fda7.jar、hpt.jar、および hptGateway.jar ファイルなどの標準 .jar ファイル。
 - Vagen1LogonPage.jsp および CSOERRORUIR.jsp ファイルなどの標準 .jsp ファイル
 - プロジェクトの UI レコードに対して生成される .jsp ファイル
 これらのファイルは、VAGen Web トランザクション・プログラムまたは UI レコードを含まないプロジェクトに対してよりも大きいプロジェクト交換ファイル・サイズとなります。

付録 I. VisualAge Generator および EGL インターオペラビリティ

以下のタイプのプログラムをマイグレーションするとき、VAGen プログラムと EGL プログラムにインターオペラビリティを提供するために、EGL プログラムを生成するか、VAGen プログラムを再リンクする必要があります。

- CICS プログラム
- iSeries プログラム
- VAGen Web トランザクション
- システム共通プロダクト

z/OS CICS における VisualAge Generator および EGL のインターオペラビリティ

同じ z/OS CICS 実行単位のすべてのプログラムは、同じランタイム・サーバー製品とリンクされる必要があります。z/OS CICS の実行単位は、同じ CICS トランザクションで実行するすべてのプログラムで構成されます (**call** または **transfer to program** (VAGen DXFR) ステートメントによって転送されるすべてのプログラムを含む)。実行単位の任意のプログラムを IBM Rational COBOL Runtime for zSeries にリンクした後、その他の VAGen プログラムを EGL にマイグレーションしたり、事前にマイグレーション済みの他のプログラムを EGL で再生成する必要は必ずしもありませんが、IBM Rational COBOL Runtime for zSeries を使用するために、実行単位内のすべてのプログラムを再リンクすることが最低限必要です。実行単位のすべてのプログラムを再リンクしない場合は、ASRA 異常終了が発生する可能性があります。

iSeries における VisualAge Generator および EGL のインターオペラビリティ

VisualAge Generator および EGL のランタイム・ライブラリーは、VisualAge Generator で生成されたプログラムおよび EGL で生成されたプログラムの共存をすぐに利用できません。このセクションでは、共存可能にするために行える変更について概論します。

このセクションでは、以下の用語を使用しています。

- *QVGEN*。VAGen 製品に付属のランタイム・ライブラリーのこと。
- *QEGL*。EGL に付属のランタイム・ライブラリーのこと。
- *PREP* ステップ。ジェネレーター・マシンによって対象のホスト・マシンに生成されたオブジェクトをデプロイするタスクのこと。このステップは、プログラムのコンパイルや作成を含みます。デプロイする必要があるその他のオブジェクトには、メッセージ・テーブル、DataTable、およびフォーマット・モジュールなどがあります。

VisualAge Generator では、VAGen Developer クライアント・ワークステーションに常駐するテンプレートを変更することによって、PREP ステップをカスタマイズすることができます。以下のテンプレートは、PREP ステップで使用されます。

- efk24pcl.tpl
- efk24pmn.tpl
- efk24psc.tpl
- efk24psm.tpl
- efk24wcl.tpl
- efk24wsc.tpl
- efk24ppm.tpl

EGL の場合、PREP ステップは、ビルド・サーバー (QEGL に付属) が FDAPREP という名前の REXX™ ビルド・スクリプトを使用して実行します。このビルド・スクリプトをカスタマイズして、コンパイラまたは CRTPGM オプションを変更することができます。また、ビルド記述子でユーザー・シンボリック・パラメーターをセットアップし、FDAPREP をこれらのパラメーターの値に従って動作させることにより、FDAPREP のロジックを制御することができます。

以下の問題は、混合環境でプログラムを呼び出すときに発生する可能性があります。

- **問題 1:** 呼び出し先プログラム (VisualAge Generator または EGL で生成した) を呼び出すときに、主な問題が発生します。メインプログラムが初期化されるときに、ライブラリー・リストに掲載されている製品に応じて、QVGEN または QEGL のサーバー・プログラムである QVGNMEM のサービス・ルーチンによって、ヒープ・メモリーが割り当てられます。例えば、QEGL がライブラリー・リストに (単独で、または QVGEN の前に) ある場合は、QEGL/QVGNMEM が使用されます。ただし、呼び出し先プログラムは、PREP ステップの間にその PREP ステップが VisualAge Generator または EGL ライブラリーのどちらを指定しているかに応じて QVGEN/QVGNMEM または QEGL/QVGNMEM にバインドされます。メインプログラムが呼び出し先プログラムを正しく呼び出すには、**両方** のプログラムが同じ QVGNMEM にバインドされる必要があります。そうでない場合は、ヒープ・メモリーが破損して、異常終了が起こります。

解決策: 以下のいずれかの技法を使用して、呼び出し側プログラムと呼び出し先プログラムの両方で同じ QVGNMEM が使用されることを確認してください。

– **技法 1:** 以下の手順に従って、常に QEGL を使用します。

1. BNDDIR(QVGEN/QVGN) の代わりに BNDDIR(QEGL/QEGLBND) を使用するように、VAGen 呼び出し先プログラムの PREP ステップを変更します。そのためには、VAGen テンプレートの efk24pcl.tpl を変更します。
2. 実行時には、QEGL ライブラリーが QVGEN ライブラリーの前にリストされるように LIBL を設定します。

– **技法 2:** 以下の手順に従って、常に QVGEN を使用します。

1. BNDDIR(QEGL/QEGLBND) の代わりに BNDDIR(QVGEN/QVGN) を使用するように、EGL 呼び出し先プログラムの PREP ステップを変更します。そのためには、EGL ビルド・スクリプト FDAPREP のプロシーチャー PCL を変更します。

2. 実行時には、QVGEN ライブラリーが QEGL ライブラリーの前にリストされるように LIBL を設定します。

QEGL を使用する技法 1 では、すべての VAGen 呼び出し先プログラムについて、PREP ステップを再実行する必要があります。これは、EGL で生成したプログラムがこの VAGen 呼び出し先プログラムと正しい (QEGL) ライブラリーをバインドするための呼び出しを、現在、または将来のいずれに行う場合も該当します。QVGEN を使用する技法 2 では、QVGEN がサービスを休止する場合に、すべての EGL プログラムについて PREP ステップを再実行する必要があります。どちらの場合も、すべての VAGen プログラムがマイグレーションされたら、QVGEN ライブラリーを削除することができます。使用する技法を判別する際には、PREP ステップを実行しなければならないプログラム数を考慮してください。例えば、EGL を使用して多数の新規システムを作成することを想定しており、これらの新規システムが VisualAge Generator で作成済みのプログラムを呼び出す可能性がある場合、技法 1 を使用して、新規システムが新規の QEGL ライブラリーとともに始動するようにしたいと考えるかもしれません。

- **問題 2:** VAGen で生成された Java クライアント・プログラムは、VAGen キャッチャー・プログラムを使用したりリモート呼び出ししか行うことができません。そして、VAGen キャッチャー・プログラムは、VAGen QVGEN モジュールにバインドされたプログラムしか呼び出すことができません。同様に、EGL で生成された Java クライアント・プログラムは、EGL キャッチャー・プログラムを使用したりリモート呼び出ししか行うことができません。そして、EGL キャッチャー・プログラムは、EGL QEGL モジュールにバインドされたプログラムしか呼び出すことができません。この制限は、Java からリモート側で呼び出されるプログラムに影響を与えます (Java ラッパーから呼び出されるプログラムも含む)。

解決策: 以下のいずれかの技法を使用して、呼び出されるプログラムの QVGNMEM モジュールがキャッチャー・プログラムの QVGNMEM モジュールと一致するようにしてください。

- **技法 1:** クライアント・プログラムが EGL を使用して生成される場合、呼び出されるプログラムおよびそれが呼び出すプログラムをすべて QEGL にバインドします。これは、それらのプログラムが VisualAge Generator または EGL のどちらで生成されるかには関係ありません。この技法をお勧めします。
- **技法 2:** クライアント・プログラムが VisualAge Generator を使用して生成される場合、呼び出されるプログラムおよびそれが呼び出すプログラムをすべて QVGEN にバインドします。これは、呼び出されるプログラムが VisualAge Generator または EGL のどちらで生成されるかには関係ありません。
- **問題 3:** VAGen が生成した Web トランザクション・プログラムは、EGL が生成した Web トランザクションと相互運用できません (**call**、**transfer**、または **show** ステートメントを使用して呼び出された、または転送されたプログラムすべてを含む)。

解決策: すべての VisualAge Generator Web トランザクションを、同時に EGL にマイグレーションする必要があります。これを行わないと、結果は予測不可能となり、呼び出されるプログラムまたは転送先のプログラムで異常終了が生じる可能性があります。

以下の表は、iSeries での VisualAge Generator および EGL プログラムのインターオペラビリティを要約しています。

表 173.

プログラムの種類	問題
メインプログラム	なし。VisualAge Generator のメインプログラムは、QEGL ライブラリーで実行でき、EGL メインプログラムは QVGEN ライブラリーで実行できます。
呼び出し先プログラム	問題 1 を参照 (前述)
リモート側から呼び出されたプログラム	問題 2 を参照 (前述)
Web トランザクション・プログラム	問題 3 を参照 (前述)
印刷サービス・プログラム	なし
フォーマット・モジュール	なし
メッセージ・テーブル	なし
データ・テーブル	なし
DDS ファイル	なし

Web トランザクションでの VisualAge Generator および EGL インターオペラビリティ

Web トランザクション・プログラムおよび UI レコード (EGL VGWebTransaction プログラムおよび VGUIRecord) に関しては、以下の規則が適用されます。

- VGWebTransaction プログラムをデバッグする前に、すべての VGWebTransaction programs および VGUIRecord を生成する必要があります。
- EGL 生成の Bean をデプロイまたは使用する場合は、以下のタスクを実行する必要があります。
 - 新規 WAR ファイルに組み込むことを計画しているすべての VGUIRecord を再生成します。
 - 対応するすべての VGWebTransaction プログラムを再生成します。
 - VGWebTransaction プログラムが **call**、**transfer**、または **show** の各ステートメント (VAGen CALL、DXFR、または XFER ステートメント) を使用して呼び出すかまたは転送する、すべての相手先プログラムをマイグレーションおよび生成します。

プログラムのリンクまたはハイパーリンクで参照されるプログラムをマイグレーションおよび生成する必要はありません。

システム共通プロダクトのインターオペラビリティ

以下のプログラムおよびそれらに関連する DataTable と FormGroup を生成する必要があります。

- システム共通プロダクト バージョン 3.3 以降で生成されたすべてのプログラムおよび関連する DataTable と FormGroup (システム共通プロダクト バージョン 3.3 の COBOL 生成機能で生成されたプログラムを含む)。
- システム共通プロダクト バージョン 4.1 で生成されたすべてのプログラムおよび関連する DataTable と FormGroup。これらのプログラムは、以下の規則に従って、EGL で生成する必要があります。
 - プログラムを EGL で生成する場合、以下のパーツも EGL で生成する必要があります。
 - プログラムが使用するすべての FormGroup。
 - FormGroup の任意の形式フィールドに **validatorDataTable** プロパティで指定されたすべての DataTable。
 - FormGroup を EGL で生成する場合、以下のパーツも EGL で生成する必要があります。
 - FormGroup を使用するすべてのプログラム。
 - FormGroup の任意の形式フィールドに **validatorDataTable** プロパティで指定されたすべての DataTable。
 - EGL で DataTable を生成する場合は、その DataTable を使用するプログラムや FormGroup を生成する必要はありません。
 - プログラム、DataTable または FormGroup を EGL で生成するとすぐに、ご使用のランタイム環境で EGL サーバー製品を使用する必要があります。

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。

〒106-8711
東京都港区六本木 3-2-12
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を提供し、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
3600 Steeles Avenue East
Markham, ON
Canada L3R 9Z7

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ひします。

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ます、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。

お客様は、IBM のアプリケーション・プログラミング・インターフェースに準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、配布することができます。

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

商標

以下は、International Business Machines Corporation の米国およびその他の国における商標です。

- IBM
- AIX
- CICS
- DB2
- IMS
- iSeries
- MVS
- OS/2
- OS/400
- Rational
- VisualAge
- WebSphere
- z/OS

Intel は、Intel Corporation または子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。

Java およびすべての Java 関連の商標およびロゴは、Sun Microsystems, Inc. の米国およびその他の国における商標です。

Linux は、Linus Torvalds の米国およびその他の国における商標です。

Microsoft[®]、Windows および Windows NT[®] 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。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。

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]

暗黙項目 63, 129, 501
 プログラム内 108
一般的な関数の EZE ワード 411
インポート 43, 220, 225, 229, 230, 235, 236, 483, 484
 外部ソース形式 32
 ステージ 3 ツール 24
ウィザード
 インポート 32, 239
エクスポート 32, 235, 483, 484
エラー・メッセージ
 HPT 471
 IWN.MIG 482
 IWN.VAL 519
 IWN.XML 534

[カ行]

外部ソース形式 24, 28, 30, 32, 33, 34, 41, 55, 56, 61, 96, 117, 159, 169, 197, 198, 207, 224, 235, 236, 237, 347, 483, 484, 486, 487, 493, 494, 501, 509, 545, 552
関数
 未確定状態の処理 113
 SQL 入出力 117
関連パーツ 4, 37, 38, 42, 47, 63, 79, 169, 199, 484
 使用しないマイグレーション 56
 マイグレーションに使用 55
関連プログラム・パーツ 108
機能 17, 35, 36, 37, 43, 47, 48, 51, 52, 53, 54, 55, 56, 61, 62, 79, 83, 84, 88, 94, 100, 101, 103, 113, 115, 120, 124, 125, 127, 128, 129, 131, 132, 133, 137, 139, 142, 143, 144, 213, 356, 357, 363, 364, 365, 367, 368, 386, 389, 483, 484, 486, 487, 492, 494, 495, 496, 498, 499, 500, 523
 共通 40, 55, 63
 名前変更 49, 215, 295, 299
 入出力 369
 DL/I I/O 380

機能 (続き)
 DL/I ステートメント 382
 SQL 62, 213
 SQL 入出力 370, 373, 377
共通コード 9, 26, 35, 38, 39, 40, 41, 42, 55, 131, 186
共通パーツ 156, 159, 183
結果
 検討 159
 ステージ 1 169
 マイグレーション・データベース 198
 中間 23
 パイロット・プロジェクト 9
 マイグレーション 486
構成マップ 13, 17, 23, 25, 26, 27, 28, 30, 34, 38, 51, 52, 54, 108, 180, 181, 182, 183, 184, 185, 186, 194, 197, 199, 200, 201, 230, 421, 549
構文 33, 115
 一般的な関数の例 365
 一般的な規則
 VisualAge Generator と EGL の違い 304
 一般的なテーブルの例 332
 一般的な表示マップの例 340
 一般的なプリンター・マップの例 343
 一般的なプログラムの例 357
 一般的なレコードの例 314
 サービス・ルーチンの一般的な例 415
 ステートメントの例
 function invocation 389
 データ項目の例 306
 テーブル 303
 プログラムの main 関数の例 363
 マップ・グループの例 334
 割り当て、MOVE、MOVEA の例 389
EGL 5, 24, 35, 36, 42, 54, 215, 229, 246, 303, 471, 509
 エラー 64
 厳密さ 36
 変換 (ステージ 2) 207
 無効 120
SET の例 391
VAGen 37, 63, 215, 483, 484
XFER の例 402

[サ行]

サービス・ルーチン 112, 303, 415
 一般構文 415
 VisualAge Generator ルーチンおよび同等な EGL ルーチン 416
サブシステム 23, 38, 39, 40, 41, 42, 46, 48, 49, 56, 62, 63, 84, 88, 90, 91, 101, 117, 120, 124, 125, 132, 171, 201, 245
システム・ライブラリー関数 36, 55, 84, 111, 142, 389, 396
実行時の違い
 COBOL
 マップ 284
 CALL 284
 DXFR 284
 XFER 284
 Java
 CALL 284
 DXFR 284
 XFER 284
充てん文字 85, 308, 327, 352
出力ファイル 34, 35
上位 PLP プロジェクト 25, 26, 27, 155, 169, 170, 171
 作成 170
シンボリック・パラメーター 421, 452, 457, 458
 パーツ関連 458
 ファイル関連 460
 ユーザー定義 460
ステージ 1 24
 Java 149
 実行 168
 設定 51, 52, 151
 Smalltalk 177
 実行 195
 設定 51, 52, 179
ステージ 2 24, 207
 実行 224
 バッチ・モード 225
 ユーザー・インターフェース 224
 設定
 設定 218
ステージ 3 24, 229
 実行 229
 設定 229
ステートメント 61, 62, 142, 303, 357, 368, 386

ステートメント (続き)

一般規則

データ項目修飾と数値リテラル
387

印刷 113

入出力 79, 213, 408, 501

入出力のあいまいさ 113

表示 113

フロー 356, 363, 404

未確定状態で作成される 79

リンク・エディット 457

レベル 77 項目 129

割り当て、MOVE、MOVEA 389

CALL 401, 408

CALL、DXFR、XFER 472

call、transfer、show 107, 501

DXFR 402

FIND 131

function invocation 389

IF、WHILE、TEST 396

RETRIEVE、FIND 394

SET 391

SETUPD、UPDATE 494

SQL 123, 125, 126, 364

use 宣言 95, 359, 491

XFER 402

use 宣言 94

制御パーツ 28, 33, 51, 220, 419, 420,
421, 472, 505

生成オプション 31, 47, 50, 51, 63,
419, 421

バインド制御 31, 50, 51, 458, 506,
507

リソース関連 31, 51, 419, 452

リンク・エディット 31, 50, 51, 457,
505, 506

リンケージ・オプション 31

リンケージ・テーブル 443

Calllink 444

Crtxlink 449

Dxfirlink 450

Filelink 448

リンケージ・テーブル・オプション
419

生成オプション 14, 247, 419, 420, 421,
457, 458, 503, 504

変換テーブルの値 441

VisualAge Generator 43

生成オプション・パーツ 14

設定 27, 33, 229, 230, 473, 480, 481

エディター 42

サンプル・ファイル 151

推奨 210

ステージ 1 25, 26, 28, 51, 201

設定 171, 174

Java 26

設定 (続き)

ステージ 1 (続き)

Java 上での設定 151

Smalltalk 上での設定 179

ステージ 2 29, 30, 224

設定 218

ステージ 3 31

設定 235

単一ファイル・モード 211

名前変更 215

必要な EGL 208

ビルド記述子 42

ファイル名の取得 191

リポジトリ・フィルター 199

ワークベンチ

設定 207

SQL 188, 213

VAGen 構文マイグレーションの設定
370, 373

VAGen マイグレーション構文の設定
215

VAGen マイグレーションの設定 49,
211, 239, 377

設定ファイル 481

マイグレーション 196

Java 27

ソース・コード 3, 9, 23, 24, 28, 29, 30,
31, 32, 35, 39, 61, 169, 199, 243, 246,
275, 415, 519

検討 246

パイロット・プロジェクト 5

Java からの抽出 149, 168

Smalltalk からの抽出 195

【タ行】

代替仕様レコード 490

単一ファイル・マイグレーション

バッチ・モード 237

ユーザー・インターフェース 236

単一ファイル・モード 32, 34, 35, 49,

50, 55, 96, 236, 237, 238, 334, 339, 484,
487, 492, 509

セットアップ 235

パーツの配置 235

マイグレーション 235

ツールの実行

ステージ 1 29, 195, 545

Java 168

Smalltalk 195

ステージ 2 29, 224

バッチ・モード 30, 225

ユーザー・インターフェース 224

ステージ 3 31, 229

バッチ・モード 32

データ項目 14, 39, 44, 46, 51, 63, 64,
79, 80, 86, 135, 142, 305, 386, 387, 472,
483, 486, 487, 488, 489, 491, 509

暗黙 107, 128, 501

設定 216

代入ステートメント 131

名前変更 49, 215, 295, 299

shared 37, 40, 41, 49, 53, 54, 79, 81,
83, 85, 91

データベース 11, 23, 24, 25, 28, 40,
115, 159, 170, 173, 218, 219, 223, 224,
229, 421, 472, 480, 481, 499, 504

データベースの更新 159, 197, 198

テーブル 79, 80, 90, 116, 331, 491

データベース 160, 187

名前変更 295, 299

FIND ステートメント 132

RETR ステートメント 132

テーブルおよび追加レコードのリスト

64, 109, 359, 421

デバッグ

実行時の違い 282

マップ 282

SQL 283

EZESQLCA 283

EZESQRRM 283

EZESQWN6 283

トレース 421

メッセージ 471

level 161, 188

【ナ行】

名前変更 27, 28, 50, 98, 152, 158, 159,
160, 180, 482, 485

ユーザー出口情報 211

「名前変更」ページ 158, 186

名前変更規則 159, 184, 185, 476, 480,
549

名前変更接頭部 49, 215, 408, 500

入出力オプション、デフォルト (未変更)
の DL/I ステートメントの 382

【ハ行】

パーツ 27

ステージ 1、2、3 24, 31

多数 23

配置 42, 50

ステージ 1 から 3 50

ステージ 1、2、3 33, 50

単一ファイル・モード 33, 235,
236

プロジェクト・リスト・パーツ
(PLP) 25

パーツ間マイグレーション 5, 25, 36, 37, 41, 55, 79
パーツ名 17, 42, 45, 49, 245
 解決 37, 48, 49
 競合 62, 112, 213, 216, 420
 重複 37, 49
 名前変更 215, 299
 無効 49, 93, 94, 107, 295, 299, 457, 458, 505, 506
 VisualAge Generator 482
パーツ名制限事項 42
バインド制御 249
バインド制御パーツ 250, 255
 テンプレートとして使用 254
 プログラム固有 256
パッケージ 13, 15, 17, 23, 26, 28, 31, 32, 34, 41, 43, 45, 46, 47, 48, 50, 51, 52, 62, 64, 91, 108, 109, 151, 156, 157, 158, 159, 168, 172, 183, 184, 197, 225, 227, 228, 233, 236, 239, 240, 245, 246, 301, 334, 421, 444, 449, 476, 480, 483
 名前変更 158, 186
 命名 184
バッチ・モード 23, 30, 32, 207, 219, 224, 226, 237, 240
パフォーマンス
 マイグレーション・ツール 555
比較値項目
 DL/I I/O 127
表示 36, 39, 40, 43, 55, 56, 64, 113, 133, 336, 338, 339, 340, 344, 368, 493, 495, 498, 509
ビルド記述子 247, 249
 デバッグ 251
 デフォルト 259
 デフォルト 249
 EGL 255
ビルド記述子オプション 247, 251, 255, 265, 266
 検討
 一般 247
 バインド 256
 リンク・エディット 256
 COBOL 生成
 検討 249
 genproject 250
 Java 生成
 検討 250
ビルド記述子パーツ 243
 検討 247
ビルド・パーツ 47, 63, 421
ビルド・パス 17, 37, 40, 41, 42, 45, 46, 48, 49, 233, 245
フィルター 26, 28, 218, 473
 構成マップ 181, 182
 バージョン 155

フィルター (続き)
 バージョン深さ 154, 155, 181, 183
 バージョン名 154, 155, 181, 182
 パッケージ 158
 プロジェクト 154, 155, 157
 リポジトリ 25, 152, 153, 154, 155, 159, 169, 171, 181, 199, 201
複数の更新を行う SQL 入出力 126
付録索引項目 295, 305, 313, 331, 334, 339, 364, 386, 404, 415, 419, 471, 509, 545
プログラム 36, 37, 38, 39, 42, 49, 50, 51, 52, 53, 55, 56, 64, 79, 86, 97, 109, 115, 136, 171, 245, 356, 357, 359, 363, 472, 483, 484, 501
 暗黙データ項目 107
 サンプル 23
 ステージ 1 ツール 26, 29
 単一ファイル・マイグレーション 32
 名前変更 295, 299
 振る舞い 23, 36
 プロパティ 42, 43
 マイグレーションに使用 54
プロジェクト名 26, 51, 154, 155, 156, 157, 158, 169, 170, 173, 184, 225, 226
プロジェクト・リスト・パーツ 17
プロジェクト・リスト・パーツ (PLP) 26, 170
ヘルプ・マップ 98, 340, 343, 357
ヘルプ・マップ名 97
ヘルプ・マップ・グループ 64, 97, 216, 472, 492, 494
変換 36, 39, 64, 137, 368
編集関数 36
編集テーブル 36, 56, 308, 312, 327, 329
編集ルーチン 36, 53, 55, 56, 83, 100, 312, 352, 356, 472, 486, 487, 492

[マ行]

マイグレーションの計画 3, 5
マイグレーション・セット 25, 27, 28, 30, 31, 37, 38, 41, 42, 51, 52, 54, 55, 56, 62, 63, 96, 109, 153, 155, 156, 157, 167, 169, 170, 172, 173, 180, 182, 183, 184, 185, 187, 195, 199, 220, 223, 224, 225, 227, 232, 334, 473, 483, 484, 549
 処理 42
マイグレーション・ツールのパフォーマンス 555
マイグレーション・データベース 26, 27, 28, 29, 30, 31, 41, 149, 152, 154, 160, 161, 169, 172, 177, 179, 180, 187, 197, 198, 227, 229, 233, 483, 484, 485, 509, 545, 551, 552
 作成 547

マイグレーション・データベース (続き)
 テーブル 548
 ビュー 548
 リセット
 テーブル 548
マイグレーション・フィーチャー 178
 追加 150
 ロード 178
マイグレーション・プラン 25, 26, 27, 28, 152, 153, 154, 159, 161, 169, 170, 173, 180, 187, 188, 189, 196, 197, 198, 201, 472, 549
 手動作成 172
 上位構成マップ 199
 複数 27
マップ 33, 37, 44, 53, 54, 63, 79, 81, 82, 98, 113, 131, 172, 245, 295, 338, 389, 495, 498, 499, 509
 一般情報 339
 印刷 501
 関数と入出力オプション 368
 数値ハードウェア属性 101
 スパン 184
 代入ステートメント 131
 定数フィールド 344, 347, 350
 名前なし変数フィールド 103
 名前変更 49, 215, 299
 表示 36, 40, 55
 一般構文、マップ・タイプ、およびプロパティ 340
 プリンター 36, 40, 55
 一般構文、マップ・タイプ、およびプロパティ 343
 変数フィールド 36, 55, 56, 100, 344, 347, 350, 352, 354
 エラー・メッセージ 356
 未確定状態 94
 無保護定数 104
 EZEMSG 408
 XFER への指定 402
マップ項目
 暗黙 107
 編集ルーチン 83
 NULL かどうかの検査 136
マップ編集 85
マップ名 97
マップ・グループ 33, 36, 39, 49, 50, 51, 79, 95, 172, 334, 472, 483, 484, 491, 492, 493, 494
 一般構文と浮動域 336
 一般情報 334
 スパン 156, 184
 装置の名前、タイプ、サイズ 338
 名前変更 295
 未確定状態 94
マップ・グループ・パーツ 14

マップ・パーツ 14
マップ・プロパティ
一般情報 308
一般編集 308
エラー・メッセージ 312
数値の編集 311
未確定状態 30, 33, 37, 41, 79, 246, 303
関数 113
その他のステートメント 128
データ項目 79
テーブル 93
プログラム 107
マップ・グループとマップ 94
レコード 86
EZE ワード 142
未使用パーツ 27, 156, 159, 183, 185
メッセージ 28, 81, 90, 97, 101, 106,
120, 124, 125, 168, 178, 197, 228, 236,
239, 308, 329, 340, 343, 344, 347
警告 161, 188
情報 161, 188
ステージ 1 共通 471
ステージ 2 224, 482
ステージ 3 219, 229
致命的 188
デバッグ 161, 188
マイグレーション・ツールからの 471
「問題」ビュー 31, 210, 218, 240,
509
HPT 471
IWN.MIG 482
IWN.VAL 519
IWN.XML 534
VisualAge for Java のステージ 1 475
VisualAge for Smalltalk のステージ
1 480
「問題」ビュー 15, 36, 48, 50, 56, 62,
63, 64, 84, 90, 95, 97, 106, 107, 115,
117, 132, 133, 134, 135, 137, 210, 237,
245, 246, 253, 421, 444, 448, 449, 484,
493, 494, 501, 506, 509, 519, 534

[ヤ行]

用語 3
予約語 28, 30, 33, 50, 295, 326, 332,
340, 343, 357, 472
テーブル名 93
プログラム名 107
EGL
リスト 295
FormGroup 名 94
Java
リスト 301
SQL 215
リスト 299

予約語 (続き)
UI レコード名 92
予約語リスト 49

[ラ行]

ライブラリー 17, 180, 182, 194, 444
管理 4, 5, 7, 9, 10, 17, 54
Smalltalk 24, 29, 179, 181, 195
リソース関連 249, 421, 452, 506, 534
リソース関連パーツ 243
EGL
検討 253
リポジトリ 17, 23, 152, 167, 173, 220
ソース・コード 4, 7, 9, 10, 24, 30,
31, 32, 421
Java 24, 27, 29, 168
リポジトリ管理 17
リポジトリ・エクスプローラー 15
リポジトリ・フィルター 182
リンク・エディット 249
リンケージ・オプション 249
リンケージ・オプション・パーツ 243
検討 251
レコード 37, 44, 63, 79, 85, 91, 92, 109,
131, 313, 314, 487, 488, 489, 499, 501
共通 63
再定義 86, 87
作業用ストレージ 29, 215
索引付き 369
シリアル 369
相対 369
代替仕様 89, 90, 316
代入ステートメント 131
名前変更 49, 295, 299
入出力 368
メッセージ・キュー 369
ユーザー・インターフェース (UI) 28,
50, 326, 327, 329, 330, 368, 472
レベル 77 項目 87, 88
DL/I 322
SQL 318, 509
レポート 26, 28, 39, 152, 169, 472
ステージ 1 マイグレーション 173,
187, 189, 191, 198, 220, 230
ログ・ファイル 26, 28, 30, 33, 168, 197,
228, 236, 239
ステージ 2 マイグレーション 224
名前の設定 191
name 161, 188, 219, 229

[ワ行]

ワークスペース 5, 7, 13, 15, 24, 30, 31,
39, 40, 41, 42, 45, 91, 150, 164, 171,
172, 173, 210, 218, 225, 227, 228, 230,
235, 238, 421, 474, 485
新規 166
重複パーツ 34, 37
復元 167, 195
保管 167, 192
ワークスペースにインポート 220, 223,
224, 226
ワークベンチ
設定
設定 207

A

array 42, 327, 347, 389, 396, 406
多次元 4
動的 4
マップ 64
AUDIT 286

C

CALL AUDIT 416
CALL COMMIT 416
CALL CREATX 416
CALL CSPTDLI 416
CALL EZCHART 416
CALL RESET 416
CICS 8, 10, 40, 49, 94, 95, 107, 275,
421, 444, 448, 449, 452, 460, 472, 491,
501
機能語
サポートされない、ネイティブ環境
286
コミットに関する違い 287
サービス・ルーチン
サポートされない、ネイティブ環境
286
サポートされない機能
ネイティブ環境 287
リソース関連
サポートされない、ネイティブ環境
287
ロールバックに関する違い 287
CALL CREATX に関する違い 287
EZE 特殊データ・ワードの違い
EZEAPP 288
EZEDEST 288
EZEDESTP 288
EZELTERM 288
EZERCODE 288
EZERT8 288

CICS (続き)

EZE 特殊データ・ワードの違い (続き)

EZESEGTR 288

EZEUSR 288

EZEUSRID 288

EZECONCT の違い 288

XFER、DXFR 287

COBOL 生成

生成とテスト 264

containerContextDependent プロパティ

17, 37, 40, 41, 42, 48, 63

D

DB2

パフォーマンス情報 207

deleteAfterUse 501

destPort 504

display 55

DL/I I/O

比較値項目 127

E

evensql 487, 490

EZE ワード 61, 79, 303, 404

一般的な関数

EZEBYTES 411

EZEC10 411

EZEC11 411

EZECOMIT 411

EZECONV 411

EZEG10 411

EZEG11 411

EZEPURGE 411

EZEROLLB 411

EZEWAIT 411

一般的な数学関数

EZEABS 413

EZECEIL 413

EZEEXP 413

EZEFLOOR 413

EZEFREXP 413

EZELDEXP 413

EZELOG 413

EZELOG10 413

EZEMAX 413

EZEMIN 413

EZEMODF 413

EZENCMPR 413

EZEPOW 413

EZEPRSCN 413

EZEROUND 413

EZESQRT 413

EZE ワード (続き)

オブジェクト・スクリプト

EZESCRPT 415

三角関数

EZEACOS 414

EZEASIN 414

EZEATAN 414

EZEATAN2 414

EZECOS 414

EZECOSH 414

EZESIN 414

EZESINH 414

EZETAN 414

EZETANH 414

その他のデータ

EZE Aid 408

EZEAPP 408

EZECNVCM 408

EZECONVT 408

EZEDEST 408

EZEDESTP 408

EZEFEC 408

EZELOC 408

EZELTERM 408

EZEMNO 408

EZEMSG 408

EZE OVER 408

EZE OVERS 408

EZERCODE 408

EZEREPLY 408

EZERT2 408

EZERT8 408

EZESEGM 408

EZESEGTR 408

EZESYS 408

EZETST 408

EZEUSR 408

EZEUSRID 408

日時

EZEDAY 408

EZEDAYL 408

EZEDAYLC 408

EZEDTE 408

EZEDTEL 408

EZEDTELC 408

EZETIM 408

浮動小数点数学関数

EZEFLADD 414

EZEFLDIV 414

EZEFLMOD 414

EZEFLMUL 414

EZEFLSET 414

EZEFLSUB 414

プログラム・フロー

EZECLOS 404

EZE FLO 404

EZE ワード (続き)

プログラム・フロー (続き)

EZERTN 404

ユーザー・インターフェース

EZEUIERR 415

EZEUILOC 415

DL/I

EZEDLCER 407

EZEDLCON 407

EZEDLDBD 407

EZEDLERR 407

EZEDLKEY 407

EZEDLKYL 407

EZEDLLEV 407

EZEDLPCB 407

EZELTERM

未確定状態 142

EZESYS

未確定状態 142

EZEWAIT

未確定状態 144

math 413

SQL

EZECONCT 405

EZESQCOD 405

EZESQISL 405

EZESQLCA 405

EZESQRD3 405

EZESQRRM 405

EZESQWN1 405

EZESQWN6 405

string

EZESBLKT 412

EZESCCWS 412

EZESCMPR 412

EZESCNCCT 412

EZESCOPY 412

EZESFIND 412

EZESNULT 412

EZESSET 412

EZESTLEN 412

EZESTOKN 412

EZEDLPCB 111

EZELOC 287

EZEPURGE 286

F

FormGroup 94, 95, 245

G

generate 37, 38, 40, 41, 48, 55, 57, 495

テーブル 32

プログラム 8, 25, 32, 37, 486, 504

generate (続き)
レポート 159, 162, 197
VisualAge Generator 172

I

import ステートメント 31, 34, 37, 40,
41, 42, 43, 45, 46, 48, 49, 62, 64, 109,
233, 246, 334, 339, 421
isDecimalDigit 102, 350
IWN.MIG 482
IWN.VAL 519
IWN.XML 534

J

Java 生成
生成とテスト 266
Java と C++ の違い
一般 290
マップ 290
EZE 特殊データ・ワード
EZCONVT 291
EZERCODE 291
SQL
EZESQLCA 291
EZESQRRM 291
EZESQWN6 291
JSP
文字定数が無効です 537

M

MigPreferences.xml 149, 151, 154, 159,
168, 177, 179
サンプル 162, 189

P

PLN の上書き 196, 202
PSB 359, 416

S

SET map PAGE 133
SQL 11, 35, 306, 314, 421, 488, 489,
495
項目が NULL かどうかの検査 136,
499
ステートメント 79
ハード・エラー 138
WHERE 文節 43, 79
SQL EZE ワード 405
SQL 行レコード 80

SQL 照会 549, 551
SQL ステートメント
未変更
Execution time statement build を使
用しない 370
modified
Execution time statement build を使
用しない 373
Execution time statement build を使
用する 377
SQL 入出力 495, 496
Execution time statement build 213
SQL 入出力オプション
ADD 370, 373, 377
CLOSE 370, 373, 377
DELETE 370, 373, 377
INQUIRY 370, 373, 377
REPLACE 370, 373, 377
SCAN 370, 373, 377
SETINQ 370, 373, 377
SETUPD 370, 373, 377
SQLEXEC 370, 373, 377
UPDATE 370, 373, 377
SQL 入出力ステートメント 116
SQL 入出力と SQL 文節の欠落 119
SQL 入出力と !itemColumnName 123,
125
SQL 表 29, 117, 488
SQL 文節
FOR UPDATE OF 373, 377
GROUP BY 373, 377
HAVING 373, 377
INTO 373, 377
ORDER BY 373, 377
SELECT 373, 377
WHERE 373, 377
SQL レコード 62, 489
代替仕様 89
SQL レコード定義 38, 39

T

table 37, 44, 49, 63, 171

U

UI レコード 53
名前変更 49
use 宣言 501

V

VAGen マイグレーションの設定 227

W

Windows XP
DB2 の使用 547



プログラム番号: 5724-J19

Printed in Japan

SD88-7536-05



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