

Oracle Forms Developer

Graphics Builder リファレンス

リリース 6*i*

2000 年 4 月

部品番号 : J01124-01

ORACLE®

部品番号: J01124-01

原本名: Oracle Forms Developer: Graphics Builder Reference, Release 6i

原本部品番号: A73075-01

Copyright © Oracle Corporation 1997, 1999. All rights reserved.

Printed in Japan.

制限付権利の説明

プログラム（ソフトウェアおよびドキュメントを含む）の使用、複製または開示は、オラクル社との契約に記された制約条件に従うものとします。著作権、特許権およびその他の知的財産権に関する法律により保護されています。

当プログラムのリバース・エンジニアリング等は禁止されています。

このドキュメントの情報は、予告なしに変更されることがあります。オラクル社は本ドキュメントの無謬性を保証しません。

* オラクル社とは、Oracle Corporation（米国オラクル）または日本オラクル株式会社（日本オラクル）を指します。

危険な用途への使用について

オラクル社製品は、原子力、航空産業、大量輸送、医療あるいはその他の危険が伴うアプリケーションを用途として開発されておりません。オラクル社製品を上述のようなアプリケーションに使用することについての安全確保は、顧客各位の責任と費用により行ってください。万一かかる用途での使用によりクレームや損害が発生いたしましても、日本オラクル株式会社と開発元である Oracle Corporation（米国オラクル）およびその関連会社は一切責任を負いかねます。当プログラムを米国国防総省の米国政府機関に提供する際には、『Restricted Rights』と共に提供してください。この場合次の Notice が適用されます。

Restricted Rights Notice

Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

このドキュメントに記載されているその他の会社名および製品名は、あくまでその製品および会社を識別する目的にのみ使用されており、それぞれの所有者の商標または登録商標です。

目次

はじめに	xvii
前提条件	xviii
表記規則	xviii
関連資料	xviii
 ビルトイン	1
PL/SQLビルトインについて	2
ビルトイン・サブプログラムの説明	2
チャート・ビルトイン	3
OG_Delete_Column	3
OG_Delete_Field	4
OG_Get_Chart_Element	5
OG_Get_Column	6
OG_Get_Field	7
OG_Get_Row	7
OG_Insert_Field	8
OG_Make_Chart	9
OG_Update_Chart	11
データベース・ビルトイン	13
OG_Connect	13
OG_Logged_On	14
OG_Logoff	14
OG_Logon	15
図表ビルトイン	17
OG_Close_Display	17
OG_Generate_Display	18
OG_Get_Display	19
OG_Isnull	20
OG_Open_Display	22
OG_Save_Display	23
グラフィック・オブジェクト・ビルトイン	25
OG_Clone (オブジェクト)	27
OG_Damage (オブジェクト)	27
OG_Delete_Child	28
OG_Delete_Cmptext	30
OG_Delete_Point	31
OG_Delete_Property	32
OG_Delete_Smptext	33
OG_Destroy (オブジェクト)	34
OG_Draw	35

OG_Export_Drawing (図表)	36
OG_Export_Drawing (オブジェクト/レイヤー)	38
OG_Export_Drawing (ウィンドウ)	40
OG_Export_Image.....	42
OG_Get_Char_Property	44
OG_Get_Child.....	45
OG_Get_CmpTEXT.....	46
OG_Get_Date_Property	47
OG_Get_Num_Property	48
OG_Get_Object.....	49
OG_Get_Point.....	50
OG_Get_Smptext	51
OG_Import_Drawing.....	53
OG_Import_Image.....	54
OG_Insert_Child	56
OG_Insert_CmpTEXT.....	57
OG_Insert_Point.....	59
OG_Insert_Smptext	60
OG_Make_Ellipse	61
OG_Make_Group	62
OG_Make_Image	63
OG_Make_Line.....	64
OG_Make_Poly.....	65
OG_Make_Rect.....	66
OG_Make_Rrect.....	67
OG_Make_Symbol.....	68
OG_Make_Text.....	69
OG_Move	70
OG_Point_In.....	71
OG_Point_Near	73
OG_Property_Exists	75
OG_Rotate.....	76
OG_Same	77
OG_Scale.....	79
OG_Set_Edgecolor.....	81
OG_Set_Fillcolor.....	82
OG_Set_Property.....	83
OG_Synchronize	84
OG_Update_Bbox	85
レイヤー・ビルトイン	87
OG_Activate_Layer.....	87
OG_Get_Layer	88
OG_Hide_Layer	89
OG_Show_Layer.....	90
その他のビルトイン	91
DO_SQL.....	92
Do_Sql Examples	93
OG_Append_Directory.....	93
OG_Append_File.....	94

OG_Center	96
OG_Damage (領域)	97
OG_Get_Attr (アプリケーション)	98
OG_Get_Attr (軸)	98
OG_Get_Attr (図表)	99
OG_Get_Attr (フィールド・テンプレート)	100
OG_Get_Attr (枠テンプレート)	101
OG_Get_Attr (オブジェクト)	102
OG_Get_Attr (プリンタ)	104
OG_Get_Attr (問合せ)	105
OG_Get_Attr (参照線)	106
OG_Get_Attr (サウンド)	106
OG_Get_Attr (タイマー)	107
OG_Get_Attr (ウィンドウ)	108
OG_Get_Buttonproc	109
OG_Help	110
OG_Host	110
OG_Pause	111
OG_Print	112
OG_Quit	112
OG_Root_Object	113
OG_Set_Attr (アプリケーション)	114
OG_Set_Attr (軸)	115
OG_Set_Attr (チャート要素)	116
OG_Set_Attr (図表)	117
OG_Set_Attr (フィールド・テンプレート)	118
OG_Set_Attr (枠テンプレート)	119
OG_Set_Attr (オブジェクト)	120
OG_Set_Attr (プリンタ)	122
OG_Set_Attr (問合せ)	123
OG_Set_Attr (参照線)	124
OG_Set_Attr (サウンド)	125
OG_Set_Attr (タイマー)	125
OG_Set_Attr (ウィンドウ)	126
OG_Translate_Envvar	127
OG_User_Exit	128
パラメータ・ビルトイン	130
OG_Delete_Param	130
OG_Get_Char_Param	131
OG_Get_Date_Param	132
OG_Get_Num_Param	133
OG_Get_Param_Type	134
OG_Param_Exists	135
OG_Set_Param	135
ビルトイン問合せ	137
OG_Append_Row	138

OG_Clear_Query	139
OG_Data_Changed	140
OG_Data_Queried	141
OG_Destroy (問合せ)	142
OG_Execute_Query	143
OG_Get_Charcell	144
OG_Get_Datecell	146
OG_Get_Newrows	147
OG_Get_Numcell	148
OG_Get_Query	149
OG_Get_Schema	150
OG_Insert_Column	151
OG_Make_Query	152
OG_Next_Row	154
OG_Numcols	155
OG_Numrows	156
OG_Set_Charcell	158
OG_Set_Datecell	159
OG_Set_Numcell	160
OG_Set_Schema	161
OG_Start_From	162
サウンド・ビルトイン	165
OG_Destroy (サウンド)	165
OG_Export_Sound	166
OG_Get_Sound	167
OG_Import_Sound	167
OG_Make_Sound	169
OG_Play_Sound	170
OG_Record_Sound	171
OG_Stop_Sound	171
テンプレート・ビルトイン	173
OG_Clone (テンプレート)	173
OG_Delete_Ftemp	174
OG_Delete_Refline	175
OG_Destroy (テンプレート)	175
OG_Export_Template	176
OG_Get_Axis	177
OG_Get_Ftemp	178
OG_Get_Refline	179
OG_Get_Template	180
OG_Import_Template	181
OG_Insert_Ftemp	182
OG_Insert_Refline	183
OG_Make_Template	184
タイマー・ビルトイン	188
OG_Activate_Timer	188
OG_Deactivate_Timer	189
OG_Destroy (タイマー)	190
OG_Get_Timer	190

OG_Make_Timer	191
TOOLS_INTビルトイン	193
TOOL_INT.ADD_PARAMETER	193
TOOL_INT.CREATE_PARAMETER_LIST	194
TOOL_INT.DELETE_PARAMETER	195
TOOL_INT.DESTROY_PARAMETER_LIST	196
TOOL_INT.GET_PARAMETER_ATTR	197
TOOL_INT.GET_PARAMETER_LIST	198
TOOL_INT.ISNULL	199
TOOL_INT.RUN_PRODUCT	200
TOOL_INT.SET_PARAMETER_ATTR	202
ウィンドウ・ビルトイン	204
OG_Destroy (ウィンドウ)	204
OG_Get_Window	205
OG_Hide_Window	206
OG_Make_Window	206
OG_Show_Window	207
プロパティ	209
アプリケーション・プロパティ	210
接続文字列プロパティ	210
カーソル・プロパティ	211
水平レイアウト解像度プロパティ	212
水平画面解像度プロパティ	213
パスワード・プロパティ	213
プラットフォーム・プロパティ	214
ユーザー名プロパティ	215
垂直レイアウト解像度プロパティ	216
垂直画面解像度プロパティ	217
円弧プロパティ	218
基本円弧プロパティ	218
閉じ形状プロパティ	219
塗り形状プロパティ	220
軸 (日付) プロパティ	222
自動最大値プロパティ	222
自動最小値プロパティ	223
自動主目盛間隔プロパティ	224
カスタム書式プロパティ	226
曜日書式プロパティ	227
年の最初の月プロパティ	228
ラベル・プロパティ	229
最大値プロパティ	231
最小値プロパティ	232
月書式プロパティ	233
四半期書式プロパティ	234

週末をスキップ・プロパティ	235
ステップ・プロパティ	236
年書式プロパティ	237
軸（一般）プロパティ	239
軸ラベル・プロパティ	239
軸タイプ・プロパティ	240
項目ラベル・プロパティ	241
軸の方向プロパティ	242
主目盛格子プロパティ	243
主目盛きざみプロパティ	244
補助目盛格子プロパティ	245
補助目盛きざみプロパティ	246
補助目盛りきざみ数プロパティ	247
位置プロパティ	248
目盛きざみラベル回転プロパティ	249
目盛きざみラベル・プロパティ	250
目盛きざみ位置プロパティ	251
軸（項目）プロパティ	253
自動最大値プロパティ	253
自動最小値プロパティ	254
日付書式プロパティ	255
最大項目数プロパティ	256
最小項目数プロパティ	257
数値書式プロパティ	258
軸（値）プロパティ	260
自動最大値プロパティ	260
自動最小値プロパティ	261
自動主目盛間隔プロパティ	262
最大値プロパティ	263
最小値プロパティ	264
数値書式プロパティ	265
パーセント元プロパティ	266
パーセント基準プロパティ	267
スケール・プロパティ	268
ステップ・プロパティ	269
チャート要素プロパティ	271
ボタン・プロシージャ・プロパティ	271
イベント・プロパティ	272
展開プロパティ	273
名前プロパティ	274
チャート・プロパティ	276
自動更新プロパティ	276
終了行プロパティ	277
フィルタ・プロパティ	278

問合せプロパティ	279
データ範囲プロパティ	280
サイズと位置プロパティ	281
開始行プロパティ	282
テンプレート・プロパティ	282
タイトル・プロパティ	283
複数テキスト・プロパティ	285
単一テキスト・カウント・プロパティ	285
図表プロパティ	286
クローズ・トリガー・プロパティ	286
日付書式プロパティ	287
高さプロパティ	287
オープン・トリガー・プロパティ	288
幅プロパティ	289
枠（軸チャート）プロパティ	291
基本線軸プロパティ	291
基本線の値プロパティ	292
項目幅プロパティ	293
カスタム日付書式プロパティ	294
カスタム数値書式プロパティ	295
参照線カウント・プロパティ	296
第2-Y軸プロパティ	296
枠（一般）プロパティ	298
深さサイズ・プロパティ	298
フィールド・テンプレート・カウント・プロパティ	299
枠タイプ・プロパティ	300
凡例プロパティ	301
凡例中の列数プロパティ	302
名前プロパティ	303
表示枠プロパティ	304
ルート・プロパティ	305
シャドウ方向プロパティ	305
シャドウ・サイズ・プロパティ	306
枠（円グラフ）プロパティ	308
項目ラベル・プロパティ	308
項目日付書式プロパティ	309
項目数値書式プロパティ	310
データ値プロパティ	311
オーバーラップなしプロパティ	312
その他プロパティ	313
パーセント書式プロパティ	314
パーセント値プロパティ	315
表示順序プロパティ	316
目盛きざみプロパティ	317

表示プロパティ	318
合計値プロパティ	319
値書式プロパティ	321
枠（表チャート）プロパティ	322
自動最大値プロパティ	322
自動最小値プロパティ	323
列名プロパティ	324
格子カウント・プロパティ	325
水平格子プロパティ	326
最大行数プロパティ	327
最小行数プロパティ	328
垂直格子プロパティ	329
フィールド・テンプレート（一般）プロパティ	331
カラー回転プロパティ	331
日付書式プロパティ	332
名前プロパティ	333
数値書式プロパティ	334
ルート・プロパティ	335
フィールド・テンプレート（軸チャート）プロパティ	336
軸プロパティ	336
カーブ調整プロパティ	337
ラベル回転プロパティ	338
線の形式プロパティ	339
オーバーラップ・プロパティ	341
表示位置プロパティ	342
表示形式プロパティ	343
一般プロパティ	345
ボタン・プロシージャ・プロパティ	345
列プロパティ	346
イベント・プロパティ	347
問合せ実行プロパティ	348
フォーマット・トリガー・プロパティ	349
オブジェクトを隠すプロパティ	350
内側枠ボックス・プロパティ	351
名前プロパティ	352
オブジェクト・タイプ・プロパティ	353
外側枠ボックス・プロパティ	354
親プロパティ	355
パラメータ設定プロパティ	356
図形プロパティ	357
バックグラウンド線カラー・プロパティ	357
バックグラウンド塗りカラー・プロパティ	358
凹凸スタイル・プロパティ	359
端形式プロパティ	360

線種スタイル・プロパティ	361
枠パターン・プロパティ	363
塗りパターン・プロパティ	365
フォアグラウンド線カラー・プロパティ	366
フォアグラウンド塗りカラー・プロパティ	367
結合形式プロパティ	368
回転角度プロパティ	369
転送モード・プロパティ	370
グループ・プロパティ	372
子の数プロパティ	372
クリップ・フラグ・プロパティ	372
イメージ・プロパティ	374
クリップする四角形プロパティ	374
粗さプロパティ	375
高さプロパティ	376
位置プロパティ	377
品質プロパティ	378
幅プロパティ	379
線プロパティ	380
矢印スタイル・プロパティ	380
終点プロパティ	381
始点プロパティ	382
多角形プロパティ	384
閉じ形状プロパティ	384
ポイント数プロパティ	385
印刷プロパティ	386
部数プロパティ	386
終了ページ・プロパティ	387
Landscapeプロパティ	387
名前プロパティ	388
ページ・サイズ・プロパティ	389
出力ファイル・プロパティ	390
開始ページ・プロパティ	390
問合せプロパティ	392
「キャッシュ・タイプ」プロパティ	392
「カスタム問合せプロシージャ」プロパティ	393
「日付書式」プロパティ	394
「オープン時実行」プロパティ	395
「タイマーで実行」プロパティ	396
「最大行」プロパティ	397
「最大行フラグ」プロパティ	398
名前プロパティ	399
「問合せ後トリガー・プロシージャ」プロパティ	399
「問合せソース」プロパティ	400

「問合せタイプ」プロパティ	401
四角形プロパティ	404
基本四角形プロパティ	404
参照線プロパティ	406
軸プロパティ	406
日付値プロパティ	407
ラベル・プロパティ	408
数値プロパティ	409
丸い四角形プロパティ	410
基本四角形プロパティ	410
丸め半径プロパティ	411
単一テキスト・プロパティ	413
カラー・プロパティ	413
フォント・プロパティ	414
テキスト文字列プロパティ	415
サウンド・プロパティ	417
名前プロパティ	417
記号プロパティ	418
中央プロパティ	418
索引プロパティ	419
記号サイズ・プロパティ	420
テキスト・プロパティ	422
バウンディング・ボックスの高さプロパティ	423
バウンディング・ボックスの幅プロパティ	424
キャラクタ・セット・プロパティ	425
カラー・プロパティ	427
複数テキスト数プロパティ	428
カスタム間隔プロパティ	429
固定バウンディング・ボックス・プロパティ	430
水平文字揃えプロパティ	431
水平原点プロパティ	432
非表示プロパティ	433
カーニング・プロパティ	434
近似プロパティ	435
原点プロパティ	436
ポイント・サイズ・プロパティ	437
スケーラブルな境界線プロパティ	437
スケーラブル・フォント・プロパティ	438
間隔プロパティ	439
スタイル・プロパティ	441
合成プロパティ	443
書体プロパティ	444
垂直文字揃えプロパティ	445
垂直原点プロパティ	446

フォントの太さプロパティ	447
フォントの幅プロパティ	449
丸めプロパティ	451
タイマー・プロパティ	452
アクティブ・プロパティ	452
間隔プロパティ	453
名前プロパティ	454
プロシージャ・プロパティ	455
ウィンドウ・プロパティ	456
高さプロパティ	456
ヘルプ・ターゲット・プロパティ	457
名前プロパティ	458
位置プロパティ	459
幅プロパティ	459
属性	461
属性レコードの使用方法	462
概要	462
属性クラス	463
結合属性レコード	464
マスク属性	465
マスク定数	465
作成可能、設定可能、取得可能な属性	467
ショートカット・ビルトイン	468
アプリケーション属性レコード	469
円弧結合属性レコード	470
円弧属性レコード	470
値軸結合属性レコード	471
値軸属性レコード	471
日付軸結合属性レコード	472
日付軸属性レコード	472
項目軸結合属性レコード	475
項目軸属性レコード	475
軸属性レコード	476
チャート結合属性レコード	478
チャート属性レコード	478
チャート要素結合属性レコード	479
チャート要素属性レコード	479
図表属性レコード	480
軸フィールド・テンプレート結合属性レコード	481
軸フィールド・テンプレート属性レコード	481
フィールド・テンプレート属性レコード	482
軸枠結合属性レコード	483

軸枠属性レコード	483
枠属性レコード	484
円グラフ枠結合属性レコード	485
円グラフ枠属性レコード	485
表枠結合属性レコード	487
表枠属性レコード	487
一般属性レコード	488
図形結合属性レコード	490
図形属性レコード	490
グループ結合属性レコード	493
グループ属性レコード	493
イメージ結合属性レコード	493
イメージ属性レコード	494
線結合属性レコード	495
線属性レコード	495
多角形結合属性レコード	496
多角形属性レコード	496
プリンタ属性レコード	497
問合せ属性レコード	497
四角形結合属性レコード	499
四角形属性レコード	499
角の丸い四角形結合属性レコード	500
角の丸い四角形属性レコード	500
サウンド属性レコード	500
記号結合属性レコード	501
記号属性レコード	501
テキスト属性の概要	502
テキスト結合属性レコード	502
テキスト属性レコード	502
フォント属性レコード	505
複数テキスト要素属性レコード	510
単一テキスト要素属性レコード	510
タイマー属性レコード	512
ウィンドウ属性レコード	513
 グローバル変数	 515
ビルトイン・グローバル変数	516
OG_App	516
OG_Inch	517
OG_Null_Axis	517
OG_Null_Buttonproc	517
OG_Null_Display	517
OG_Null_Ftemp	518
OG_Null_Layer	518

OG_Null_Object	518
OG_Null_Query	518
OG_Null_Refline	519
OG_Null_Sound.....	519
OG_Null_Template.....	519
OG_Null_Timer	519
OG_Null_Window	520
 索引	 521

はじめに

Oracle Forms Developer Graphics Builderリファレンス、リリース6iによろこそ。

このリファレンス・ガイドでは、Oracle Forms Developer Graphics Builderを効果的に利用できる
ようにするための情報と、次の項目に関する詳細な情報が説明されています。

- ビルトイン
- プロパティ
- 属性
- グローバル変数

ここでは、このガイドの構成を説明し、Graphics Builderを使用する際に参考になるその他の情
報源を紹介します。

前提条件

まず、ご使用のコンピュータおよびそのオペレーティング・システムについて精通している必要があります。たとえば、ファイルの削除およびコピーのコマンドの知識があり、検索パス、サブディレクトリおよびパス名を概念を理解していなければなりません。詳細は、各オペレーティング・システムの製品マニュアルを参照してください。

アプリケーション・ウィンドウの要素などのMicrosoft Windowsの基本要素も理解している必要があります。エクスプローラ、タスクバー、タスクマネージャ、またはレジストリなどのプログラムに精通している必要があります。

表記規則

このマニュアルでは、次のような表記上の規則を使用しています。

規則	意味
固定幅フォント	固定幅フォントのテキストは、表示されたとおりに入力するコマンドを示します。PCに入力するテキストでは、特に断りのない限り大文字と小文字を区別しません。 コマンドでは、大カッコと縦線以外の句読点は表示されているとおり正確に入力する必要があります。
小文字	コマンド文の小文字は変数を表します。適切な値に置き換えてください。
大文字	テキスト内の大文字は、コマンド名、SQL予約語、キーワードを表します。
ゴシック・テキスト	メニュー選択項目やボタンなど、ユーザー・インタフェース項目を示すには、ゴシック・テキストが使用されます。
C>	C>はDOSプロンプトを表します。実際とは異なる場合があります。

関連資料

次のOracleマニュアルを参照することもできます。

タイトル	部品番号
『Oracle Forms Developer and Oracle Reports Developer アプリケーション作成ガイド リリース6i』	J00449-01

ビルトイン

PL/SQLビルトインについて

このセクションでは、すべてのGraphics Builderビルトイン・サブプログラムについて説明します。これらのサブプログラムは、PL/SQL言語の拡張機能で、独自のサブプログラムにGraphics Builder機能と機能性を持たせることを可能にします。

ビルトイン・サブプログラムの説明

説明:

サブプログラムの用途と使用方法を指定します。

構文:

サブプログラムをコールするための構文です。サブプログラムをコールする方法が複数ある場合は、すべて表示されます。

パラメータ

サブプログラムのパラメータについて説明します。

戻り値:

サブプログラムの戻り値（ファンクションのみ）について説明します。

コメント:

使用方法のルールと警告について説明します。

例:

サブプログラム使用法の作業例です。

チャート・ビルトイン

OG_Delete_Column
OG_Delete_Field
OG_Get_Chart_Element
OG_Get_Column
OG_Get_Field
OG_Get_Row
OG_Insert_Field
OG_Make_Chart
OG_Update_Chart

OG_Delete_Column

説明

このプロシージャでは、カスタムの間合せから列を削除します。

構文

```
PROCEDURE OG_Delete_Column
  (query_hdl  OG_Query,
   indx       NUMBER,
   total      NUMBER);
```

パラメータ

<i>query_hdl</i>	列が削除される間合せのハンドル。
<i>indx</i>	間合せから削除する最初の列の索引。
<i>total</i>	削除する列の合計数。

OG_Delete_Columnの例

```
/* The following procedure deletes a column
** from the query 'query0':
*/
```

```
PROCEDURE example(col_num number) IS
```

```

    query    OG_Query;
BEGIN
    query:=OG_Get_Query('query0');
    OG_Delete_Column(query, col_num, 1);
END;
```

OG_Delete_Field

説明

このプロシージャでは、指定したチャート・オブジェクトから1つ以上のフィールドを削除します。

構文

```

PROCEDURE OG_Delete_Field
    (chart_hdl    OG_Object,
     indx         NUMBER,
     total        NUMBER);
```

パラメータ

<i>chart_hdl</i>	チャート・オブジェクトのハンドル。
<i>indx</i>	フィールド・リストから削除する最初のフィールドの索引。
<i>total</i>	削除するフィールドの合計数。

使用上の注意

フィールドの削除では、指定したチャートから該当するフィールドが削除されるのみです。フィールドが参照しているフィールド・テンプレートは削除されません(変更されることはありません)。また、チャートのフィールドを変更しても、そのチャートをOG_Update_Chartのコールによって更新するまで変更は適用されません。

OG_Delete_Fieldの例

```

/* Suppose one chart currently displays plots for both salary
** and commission data, and you want to remove the
** commission plot from that chart and plot it on another one.
*/

PROCEDURE transfer_comm(chart1 IN OG_Object, chart2 IN
OG_Object, field_index in number) IS
    the_field    OG_Field;
BEGIN
    the_field:=OG_Get_Field(Chart1, field_index);
    OG_Delete_Field(Chart1, field_index, 1);
```

```

OG_Insert_Field(Chart2, the_field, OG_Last);
OG_Update_Chart(Chart1, OG_All_Chupda);
OG_Update_Chart(Chart2, OG_All_Chupda);
END;
```

OG_Get_Chart_Element

説明

チャート要素（棒グラフの棒、円グラフの円弧など）のハンドルと行番号を割り当てると、このファンクションにより、行番号に対応した個々の要素が戻されます。

構文

```

FUNCTION OG_Get_Chart_Element
  (group_hdl  OG_Object,
   row_num    NUMBER)
RETURN OG_Object;
```

パラメータ

<i>group_hdl</i>	チャート要素で構成されるグループのハンドル。
<i>row_num</i>	取得するチャート要素に対応した行番号。

戻り値

指定した行番号に対応した個々のチャート要素。

使用上の注意

グループ・ハンドルは、OG_Get_Objectに該当する名前を指定して、チャート・オブジェクトから取り出せます。

OG_Get_Chart_Elementの例

```

/* The following procedure changes the color of the first:
** bar in a column chart, regardless of its value:
*/

PROCEDURE example(chart OG_Object) IS
  bars_group  OG_Object;
  elem        OG_Object;
BEGIN
  bars_group := OG_Get_Object('Sal_Bars', chart);
  elem := OG_Get_Chart_Element(Bars_Group, 0);
  OG_Set_Fillcolor(Elem, 'red');
```

END;

OG_Get_Column

説明

このファンクションを使用して、特定のチャート要素に対応付けられた問合せの列の名前を戻します。

構文

```
FUNCTION OG_Get_Column  
  (chelement_hdl OG_Object)  
RETURN VARCHAR2;
```

パラメータ

<i>chelement_hdl</i>	チャート要素のハンドル。
----------------------	--------------

戻り値

チャート要素に対応付けられた列の名前。

OG_Get_Columnの例

```
/* The following function returns the query column represented by  
** the first bar in a column chart:  
*/
```

```
FUNCTION example(chart OG_Object) RETURN CHAR IS  
  bars OG_Object;  
  elem OG_Object;  
  col VARCHAR2(15);  
BEGIN  
  bars := OG_Get_Object('Sal_Bars', chart);  
  elem := OG_Get_Chart_Element(Bars, 0);  
  col := OG_Get_Column(Elem);  
  RETURN(col);  
END;
```


OG_Get_Field

説明

このファンクションを使用して、フィールドの属性値で構成されたレコードを、指定したチャートに戻します。

構文

```
FUNCTION OG_Get_Field
  (chart_hdl OG_Object,
   indx      NUMBER)
RETURN OG_Field;
```

パラメータ

<i>chart_hdl</i>	チャート・オブジェクトのハンドル。
<i>indx</i>	チャートのフィールド内の、戻されるフィールドの索引。

戻り値

指定されたフィールドの属性。

OG_Get_Fieldの例

```
/* Suppose one chart currently displays plots for both salary
** and commission data, and you want to remove the
** commission plot from that chart and plot it on another one:
*/
```

```
PROCEDURE transfer_comm(chart1 IN OG_Object, chart2 IN
  OG_Object, field_index IN NUMBER) IS
  the_field OG_Field
BEGIN
  the_field:=OG_Get_Field(The_Chart, field_index);
  OG_Delete_Field(Char1, field_index, 1);
  OG_Insert_Field(Char2, the_field, OG_Last);
END;
```

OG_Get_Row

説明

このファンクションを使用して、特定のチャート要素で対応付けられた問合せの行番号を戻します。

構文

```
FUNCTION OG_Get_Row  
    (chelement_hdl OG_Object)  
RETURN NUMBER;
```

パラメータ

<i>chelement_hdl</i>	チャート要素のハンドル。
----------------------	--------------

戻り値

チャート要素に対応付けられた行番号。

OG_Get_Rowの例

```
/* The following format trigger explodes the pie slice  
** representing SAL for employee 'SMITH':  
*/
```

```
PROCEDURE OGFORMATTRIG0(elem IN OG_Object,  
                        query IN OG_Query) IS  
    ename    VARCHAR2(10);  
    chart    OG_Object;  
    row_num  NUMBER;  
BEGIN  
    ename := OG_Get_Charcell(Query, 'ENAME');  
    IF ename = 'SMITH' THEN  
        chart := OG_Get_Object('Chart0');  
        row_num := OG_Get_Row(Elem);  
        OG_Set_Explosion(Char, row_num, 'SAL', 25);  
    END IF;  
END;
```

OG_Insert_Field

説明

このプロシージャでは、指定したチャートに新しいフィールドを挿入します。

構文

```
PROCEDURE OG_Insert_Field  
    (chart_hdl OG_Object,  
     field_rec OG_Field,  
     indx      NUMBER);
```

パラメータ

<i>chart_hdl</i>	チャート・オブジェクトのハンドル。
<i>field_rec</i>	フィールドの属性で構成されるレコード。
<i>indx</i>	<p>チャートのフィールドに新しいフィールドを挿入するための索引。この引数は、必ず0から <i>n</i> まで(0と<i>n</i>を含む)の整数であることが必要です。この場合、<i>n</i> は、挿入を行う前にチャートの中にあったフィールドの数になります。次のビルトイン定数のいずれかを指定することもできます。</p> <p>OG_First チャートのフィールド・リストの最初に新しいフィールドを挿入する(索引は0になります)。</p> <p>OG_Last チャートのフィールド・リストの最後に新しいフィールドを挿入する(索引は挿入を行う前にチャートの中にあったフィールドの数になります)。</p>

使用上の注意

チャートのフィールドを変更しても、そのチャートをOG_Update_Chartのコールによって更新するまで変更は適用されません。

OG_Insert_Fieldの例

```
/* Suppose one chart currently displays plots for both
** salary and commission data, and you want to remove
** the commission plot from that chart and plot it on another one:
*/
```

```
PROCEDURE transfer_comm (chart1 IN OG_Object, chart2 IN
OG_Object, field_index IN NUMBER) IS
    the_field OG_Field;
BEGIN
    the_field:=OG_Get_Field(The_Chart, field_index);
    OG_Delete_Field(Char1, field_index, 1);
    OG_Insert_Field(Char2, the_field, OG_Last);
END;
```

OG_Make_Chart

説明

このファンクションを使用して、チャートを作成します。

構文

```
FUNCTION OG_Make_Chart
    (position OG_Point,
```

```
height    NUMBER,  
width     NUMBER,  
template  OG_Template,  
query     OG_Query)  
RETURN OG_Object;
```

パラメータ

<i>position</i>	チャート枠のx座標とy座標。
<i>height</i>	チャートの高さ。
<i>width</i>	チャートの幅。
<i>template</i>	チャートに使用するテンプレート。
<i>query</i>	チャートに使用する問合せ。

戻り値

新しく作成したチャートのハンドル。

使用上の注意

チャートを完成するには、OG_Insert_Fieldによってチャートにフィールドを追加し、OG_Update_Chartによってチャートを更新する必要があります。

OG_Make_Chartの例

```
/* The following function creates a chart using  
** the specified template and query:  
*/  
  
FUNCTION example(template OG_Template, query OG_Query) RETURN OG_Object  
IS  
  chart  OG_Object;  
  pos    OG_Point;  
  height NUMBER;  
  width  NUMBER;  
BEGIN  
  pos.x := OG_Inch;  
  pos.y := OG_Inch;  
  height := 4* OG_Inch;  
  width  := 4* OG_Inch;  
  
  chart := OG_Make_Chart(Pos, height, width, template, query);  
  RETURN(chart);  
END;
```

OG_Update_Chart

説明

このプロシージャでは、指定したチャートの指定した部分を更新して、新しい問合せ結果や、チャート要素にすでに適用されている新しい属性を反映させます。問合せに基づくチャートを更新するには、その問合せを少なくとも一度は実行した後にしてください。

構文

```
PROCEDURE OG_Update_Chart
  (chart_hdl      OG_Object,
   chart_mask     NUMBER,
   damage         BOOLEAN    := TRUE,
   update_bbox    BOOLEAN    := TRUE);
```

パラメータ

<i>chart_hdl</i>	更新するチャートのハンドル。
<i>chart_mask</i>	更新する必要があるチャート部分を指定する。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_All_Chupda チャート全体を更新する。 OG_Dep1axis_Chupda 最初の従属軸に対応付けられたチャート部分のみを更新する。 OG_Dep2axis_Chupda 2番目の従属軸に対応付けられたチャート部分のみを更新する。 OG_Frame_Chupda 枠に対応付けられたチャート部分のみを更新する。 OG_Indaxis_Chupda 独立軸に対応付けられたチャート部分のみを更新する。 OG_Inframe_Chupda 枠内に表示されたチャート部分のみを更新する。 OG_Legend_Chupda 凡例に対応付けられたチャート部分のみを更新する。 OG_None_Chupda どのチャート部分も更新しない。 OG_Title_Chupda チャートのタイトルのみを更新する。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

このプロシージャを起動すると、Graphics Builderにより現行のチャートが破棄され、更新した問合せの結果や属性の設定に従ってもう一度チャートが作成されます。このため、OG_Update_Chartを最後に起動してからチャート要素の属性の設定に対して加えた変更はすべて消失します。たとえば、OG_Set_Attrを使用してチャート内の特定の棒の属性を設定し、その棒を更新すると、希

望どおりの結果が得られます。ただし、OG_Update_Chartをもう一度コールすると、その変更はすべて消失し、棒の表示がデフォルト設定に戻ります。

チャートを更新するときには必ず、それぞれのチャート要素から変更してください。チャート要素を変更する基準はデータの変更に応じて変わることがあるので、この手順が必要となります。

OG_Update_Chartの例

```
/* Suppose you want to update a chart periodically.  
**You could write the following timer trigger:  
*/
```

```
PROCEDURE my_timer IS+  
    my_query    OG_Query;  
    my_chart    OG_Object;  
BEGIN  
    my_query:=OG_Get_Query('Emp_Query');  
    my_chart:=OG_Get_Object('Emp_Chart');  
    OG_Execute_Query(My_Query);  
    OG_Update_Chart(My_Chart, OG_All_Chupda);  
END;
```

データベース・ビルトイン

OG_Connect

OG_Logged_On

OG_Logoff

OG_Logon

OG_Connect

説明

このファンクションを使用して、「接続」ダイアログ・ボックスを表示します。

構文

```
FUNCTION OG_Connect  
RETURN BOOLEAN;
```

パラメータ

なし。

OG_Connectの例

```
/* Suppose your application requires the  
** user to be connected to a database.The  
** following procedure checks if a connection  
** exists and, if not, prompts the user to  
** connect by showing the Connect dialog box:  
*/
```

```
PROCEDURE ensure_connection IS  
status BOOLEAN;  
BEGIN  
    IF NOT OG_Logged_On THEN  
        status:=OG_Connect;  
    END IF;  
END;
```

OG_Logged_On

説明

このファンクションを使用して、ユーザーが現在データベースに接続している場合にはTRUEを、接続していない場合にはFALSEを戻します。

構文

```
FUNCTION OG_Logged_On
RETURN BOOLEAN;
```

パラメータ

なし。

戻り値

TRUE	ユーザーがデータベースに接続している場合。
FALSE	ユーザーがデータベースに接続していない場合。

OG_Logged_Onの例

```
/* Suppose your application requires the user to be
** connected to a database.The following procedure
** checks if a connection exists and, if not, prompts the
** user to connect by showing the Connect dialog box:
*/

PROCEDURE ensure_connection IS
status BOOLEAN;
BEGIN
    IF NOT OG_Logged_On THEN
        status := OG_Connect;
    END IF;
END;
```

OG_Logoff

説明

このプロシージャで、既存のデータベース接続をクローズします。

構文

```
PROCEDURE OG_Logoff;
```


パラメータ

なし。

OG_Logoffの例

```
/* Suppose you want to disconnect from a database when the display is closed.
** You could write the following Close Display trigger:
*/

PROCEDURE close_trig IS
BEGIN
    IF OG_Logged_On THEN
        OG_Logoff;
    END IF;
END;
```

OG_Logon

説明

このプロシージャで、指定したデータベースへの接続を確立します。

構文

```
PROCEDURE OG_Logon
(
    username      VARCHAR2 := NULL,
    password      VARCHAR2 := NULL,
    connect_string VARCHAR2 := NULL);
```

パラメータ

username	使用するユーザー名。
password	使用するパスワード。
connect_string	使用するデータベース接続文字列。リモート・データベースに接続する場合は、適切なSQL*Netデータベース接続文字列を指定する必要があります。詳細は、『Oracle Network Manager管理者ガイド』を参照してください。

使用上の注意

1つの接続がすでに確立されているときに、このプロシージャで次の接続を試行すると、その接続が成功しても失敗しても、最初にあった接続が切断されます。

OG_Logonの例

```
/* Suppose your application requires the user to be connected to a database.  
** The following procedure checks if a connection exists and, if not,  
** automatically establishes a connection:  
*/
```

```
PROCEDURE ensure_connection IS  
BEGIN  
    IF NOT OG_Logged_On THEN  
        OG_Logon('Scott', 'tiger', 't:london:MY_DB');  
    END IF;  
END;
```

図表ビルトイン

- OG_Close_Display
- OG_Generate_Display
- OG_Get_Display
- OG_Isnull
- OG_Open_Display
- OG_Save_Display

OG_Close_Display

説明

このプロシージャで、指定された図表をクローズし、その図表で使用されているすべてのウィンドウを破棄します。また、指定された図表のクローズ・トリガーを実行します。

構文

```
PROCEDURE OG_Close_Display
  (display_hdl  OG_Display);
```

パラメータ

<i>display_hdl</i>	クローズする図表のハンドル。
--------------------	----------------

使用上の注意

現行の図表をクローズするプロシージャをコールする場合、OG_Close_Displayはそのプロシージャの最後の行に記述してください。一度図表をクローズした後で、PL/SQLを実行することはできません。

OG_Close_Displayの例

```
/* Suppose the user is through with one display,
** and you want to close it and open another one.
*/

PROCEDURE continue(old_disp_name, new_disp_name) IS
  old_disp  OG_Display;
```

```
new_disp  OG_Display;
BEGIN
  old_dispb:=OG_Get_Display(Old_Disp_Name, OG_FileSystem);
  OG_Close_Display(Old_Disp);
  new_dispb:=OG_Open_Display(New_Disp_Name, OG_FileSystem);
END;
```

OG_Generate_Display

説明

このファンクションで、現行の図表を生成します。生成された図表は、Graphics Builderランタイムやバッチ・プログラムで実行されます。

構文

```
PROCEDURE OG_Generate_Display;

PROCEDURE OG_Generate_Display
  (name          VARCHAR2,
   repository    OG_Number);
```

パラメータ

<i>name</i>	生成される図表に割り当てる名前。図表をデータベースに格納する場合は、この引数に図表の名前のみを指定します。図表をファイル・システムに格納する場合は、この引数に図表ファイルの絶対パス名または相対パス名を指定します。
<i>repository</i>	図表をファイル・システムまたはデータベースのどちらに格納するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db 図表をデータベースに格納します。 OG_FileSystem 図表をファイル・システムに格納します。

使用上の注意

*name*と*repository*を省略すると、図表が最後にオープンされたときの名前とリポジトリを使用して図表が生成されます。

OG_Generate_Displayの例

```
/* Suppose your display allows the user to interactively specify
**which queries to view, and what chart types to use.
**When the user selects a 'generate' button, you may want to
**generate a runtime version of the display
** that the user can use in the future.
*/
```

```
PROCEDURE gen(buttonobj IN OG_Object, hitobj IN OG_Object,
  win IN OG_Window, eventinfo IN OG_Event) IS
BEGIN
  og_generate_display('my_disp', OG_FILESYSTEM);
END;
```

OG_Get_Display

説明

現行のGraphics Builderセッションでオープンしている図表のハンドルを戻します。現在実行中の図表以外の図表をオープンするには、OG_Open_Displayを使用します。

構文

```
FUNCTION OG_Get_Display
RETURN OG_Display;

FUNCTION OG_Get_Display
  (display_name VARCHAR2,
   repository NUMBER)
RETURN OG_Display;
```

パラメータ

<i>display_name</i>	図表の名前。図表がデータベースに格納されている場合は、この引数に図表の名前のみを指定します。図表がファイル・システムに格納されている場合は、この引数に図表ファイルの絶対パス名または相対パス名を指定します。
<i>repository</i>	図表がファイル・システムまたはデータベースのどちらに格納されているかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db 図表をデータベースに格納します。 OG_FileSystem 図表をファイル・システムに格納します。

戻り値

指定された図表のハンドル。図表が存在しないか、またはオープンされていない場合は、NULLハンドルが戻されます。

使用上の注意

*display_name*と*repository*を省略すると、このファンクションによって現行の図表のハンドルが戻ります。

OG_Get_Displayの例

```
/* Suppose the user is through with one display,
** and you would like to close it and open another one.
```

```
*/  
  
PROCEDURE continue(old_disp_name, new_disp_name) IS  
    old_disp    OG_Display;  
    new_disp    OG_Display;  
BEGIN  
    old_disp:=OG_Get_Display(Old_Disp_Name, OG_Filesystem);  
    OG_Close_Display(Old_Disp);  
    new_disp:=OG_Open_Display(New_Disp_Name, OG_Filesystem);  
END;
```

OG_Isnull

説明

このファンクションで、指定されたハンドルがNULLハンドルであるかどうかを判別します。

構文

FUNCTION OG_Isnull (<i>handle</i> OG_Query) RETURN BOOLEAN;	問合せ
FUNCTION OG_Isnull (<i>handle</i> OG_Object) RETURN BOOLEAN;	オブジェクト
FUNCTION OG_Isnull (<i>handle</i> OG_Template) RETURN BOOLEAN;	チャート・テンプレート
FUNCTION OG_Isnull (<i>handle</i> OG_Buttonproc) RETURN BOOLEAN;	ボタン・プロシージャ
FUNCTION OG_Isnull (<i>handle</i> OG_Sound) RETURN BOOLEAN;	サウンド
FUNCTION OG_Isnull (<i>handle</i> OG_Window) RETURN BOOLEAN;	ウィンドウ
FUNCTION OG_Isnull (<i>handle</i> OG_Layer) RETURN BOOLEAN;	レイヤー
FUNCTION OG_Isnull	タイマー

<code>(handle OG_Timer)</code>	
<code>RETURN BOOLEAN;</code>	
<code>FUNCTION OG_Isnull</code>	図表
<code>(handle OG_Display)</code>	
<code>RETURN BOOLEAN;</code>	
<code>FUNCTION OG_Isnull</code>	軸
<code>(handle OG_Axis)</code>	
<code>RETURN BOOLEAN;</code>	
<code>FUNCTION OG_Isnull</code>	フィールド・テンプレート
<code>(handle OG_Ftemp)</code>	
<code>RETURN BOOLEAN;</code>	
<code>FUNCTION OG_Isnull</code>	参照線
<code>(handle OG_Refline)</code>	
<code>RETURN BOOLEAN;</code>	

パラメータ

ハンドル	評価対象のハンドル。
------	------------

戻り値

TRUE	ハンドルがNULLハンドルである場合。
FALSE	ハンドルがNULLハンドルでない場合。

OG_Isnullの例

```
/* Suppose your display occasionally creates a text object that contains
a warning message. At times you
** may want to remove the warning message before continuing with the execution
of the display. Rather
** than keeping track of whether a warning has been generated, you can
check for the existence of the
** text object before deleting it.
*/

PROCEDURE remove_warning IS
    warning_obj  OG_Object;
BEGIN
    warning_obj:=OG_Get_Object('warning');
    IF NOT (OG_Isnull(warning_obj)) THEN;
        OG_Destroy(warning_obj);
    END IF;
END;
```

OG_Open_Display

説明

このファンクションで、指定された図表をオープンし、そのオープン・トリガーを実行します。また、後でOG_Close_Displayの引数として使用できる図表のハンドルを戻します。図表が存在しない場合は、このファンクションはNULLハンドルを戻します。

構文

```
FUNCTION OG_Open_Display
  (display_name  VARCHAR2,
   repository    NUMBER)
RETURN OG_Display;
```

パラメータ

<i>display_name</i>	図表の名前。図表がデータベースに格納されている場合は、この引数に図表の名前のみを指定します。図表がファイル・システムに格納されている場合は、この引数に図表ファイルの絶対パス名または相対パス名を指定します。
<i>repository</i>	図表がファイル・システムまたはデータベースのどちらに格納されているかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db 図表はデータベースに格納されています。 OG_Filessystem 図表はファイル・システムに格納されています。

戻り値

新たにオープンされた図表のハンドル。

使用上の注意

このファンクションでは、引数として図表のハンドルを受け入れません。これは、図表をオープンする前に、まずファイル・システムまたはデータベース内で図表の存在を検証する必要があるからです。OG_Get_Displayを使用して図表のハンドルを取得した後で、ファイル・システムまたはデータベースからその図表を削除したとします。図表のハンドルをOG_Open_Displayへ渡そうとしても、ハンドルで参照される図表の検出はできません。したがって、図表の名前をもう一度指定する必要があります。

OG_Open_Displayの例

```
/* Suppose the user is through with one display,
** and you would like to close it and open another one.
*/

PROCEDURE continue(old_display_name IN CHAR,
new_display_name IN CHAR) IS
```



```
old_display  OG_Display;
new_display  OG_Display;
BEGIN
  old_display:=OG_Get_Display(old_display_name, OG_FileSystem);
  new_display:=OG_Open_Display(new_display_name, OG_FileSystem);
  OG_Close_Display(old_display);
END;
```

OG_Save_Display

説明

このファンクションで、図表の現行の状態を保存します。保存された図表は、Graphics Builderでオープンしたり編集したりできます。

構文

```
PROCEDURE OG_Save_Display;

PROCEDURE OG_Save_Display
  (name          VARCHAR2,
   repository    OG_Number);
```

パラメータ

<i>name</i>	保存する図表に割り当てる名前。図表をデータベースに格納する場合は、この引数に図表の名前のみを指定します。図表をファイル・システムに格納する場合は、この引数に図表ファイルの絶対パス名または相対パス名を指定します。
<i>repository</i>	図表をファイル・システムまたはデータベースのどちらに格納するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db 図表をデータベースに格納します。 OG_FileSystem 図表をファイル・システムに格納します。

使用上の注意

*name*と*repository*を省略すると、図表が最後にオープンされたときの名前とリポジトリを使用して図表が保存されます。

OG_Save_Displayの例

```
/* Suppose you want to import 100 TIFF images.Doing this
**manually is tedious and would take a long time.
**The solution is to write the following procedure,
**which imports images from the files named 'image00'
**through 'image99'.When finished, it saves the display
**so that you can open it again in the Builder.
```

```
*/
```

```
PROCEDURE import_100 IS
    the_image  OG_Image;
    file_name  VARCHAR2(7);
BEGIN
    FOR i IN 0..99 LOOP
        file_name:='image'||SUBSTR(TO_CHAR(i, '09'), 2);
        the_image:=OG_Import_Image(File_Name, OG_FileSystem,
OG_Tiff_Iformat);
    END LOOP;
    OG_Save_Display;
END;
```

グラフィック・オブジェクト・ビルトイン

OG_Clone (オブジェクト)
OG_Damage (オブジェクト)
OG_Delete_Child
OG_Delete_Cmptext
OG_Delete_Point
OG_Delete_Property
OG_Delete_Smptext
OG_Destroy (オブジェクト)
OG_Draw
OG_Export_Drawing (図表)
OG_Export_Drawing (オブジェクト/レイヤー)
OG_Export_Drawing (ウィンドウ)
OG_Export_Image
OG_Get_Char_Property
OG_Get_Child
OG_Get_Cmptext
OG_Get_Date_Property
OG_Get_Num_Property
OG_Get_Object
OG_Get_Point
OG_Get_Smptext
OG_Import_Drawing
OG_Import_Image
OG_Insert_Child

OG_Insert_Cmptext
OG_Insert_Point
OG_Insert_Smptext
OG_Make_Ellipse
OG_Make_Group
OG_Make_Image
OG_Make_Line
OG_Make_Poly
OG_Make_Rect
OG_Make_Rrect
OG_Make_Symbol
OG_Make_Text
OG_Move
OG_Point_In
OG_Point_Near
OG_Property_Exists
OG_Rotate
OG_Same
OG_Scale
OG_Set_Edgecolor
OG_Set_Fillcolor
OG_Set_Property
OG_Synchronize
OG_Update_Bbox

OG_Clone (オブジェクト)

説明

このファンクションを使用して、指定したオブジェクトと同じオブジェクトを新たに作成します。

構文

```
FUNCTION OG_Clone
  (object_hdl  OG_Object,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE)
RETURN OG_Object;
```

パラメータ

<i>object_hdl</i>	複製するオブジェクトのハンドル。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

戻り値

新たに作成されたオブジェクトのハンドル。

OG_Clone (オブジェクト) の例

```
/* Suppose you have created an object, and you want to
** create another identical object without having to
** again specify the same properties.
*/

PROCEDURE dup_object (old_object IN OG_Object) IS
  new_object  OG_Object;
BEGIN
  new_object := OG_Clone (Old_Object);
END;
```

OG_Damage (オブジェクト)

説明

このプロシージャで、レイアウトされたオブジェクトを再描画します。

構文

```
PROCEDURE OG_Damage
```

```
(object_hdl OG_Object);
```

パラメータ

<code>object_hdl</code>	再描画するオブジェクトのハンドル。
-------------------------	-------------------

OG_Damage (オブジェクト) の例

```
/*Suppose you want to move an object. The default behavior of the built-in
**procedure OG_Move is to update the bounding boxes of all of the modified
**object's antecedants, including the layer on which the object resides.
**To update a layer's bounding boxes, Graphics Builder must examine every
object
**on that layer.If the layer contains a large number of objects,
**this operation can be very time-consuming.
*/

/*To make your application more efficient, you can move the object
**while inhibiting this automatic bounding box update, then explicitly
**update only that object's bounding boxes.(Note that since the
**automatic bounding box update does not occur, the bounding boxes
**of the object's antecedants may be inaccurate.)
*/

/*When you modify an object with a FALSE bounding box update flag,
**you may also want to use a FALSE damage flag.In this case,
**when you are through modifying the object, you would invoke
**OG_Damage to explicitly damage the object.
*/

PROCEDURE move_efficiently (the_object OG_Object) IS
    offset    OG_Point;
BEGIN
    offset.x:=OG_Inch;
    offset.y:=OG_Inch;
    OG_Move(The_Object, offset, FALSE, FALSE)
    OG_Update_Bbox(The_Object, OG_Bothbbox);
    OG_Damage(The_Object);
END;
```

OG_Delete_Child

説明

このプロシージャで、指定したグループ・オブジェクトから1つ以上の子オブジェクトを削除します。

構文

```
PROCEDURE OG_Delete_Child
  (group_hdl   OG_Object,
   indx        NUMBER,
   total        NUMBER,
   damage       BOOLEAN    := TRUE,
   update_bbox  BOOLEAN    := TRUE);
```

パラメータ

<i>group_hdl</i>	グループ・オブジェクトのハンドル。
<i>indx</i>	グループの子リストから削除する最初のオブジェクトの索引。
<i>total</i>	削除する子オブジェクトの総数。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

子を削除するというのは、オブジェクトとその親グループとの関連付けをなくすという意味です。子オブジェクトを破棄するという意味ではありません。つまり、削除された子の一般属性レコード内の親の属性がNULLハンドルに設定され、そのグループのオブジェクト・リストからその子が削除されます。なお、親の属性がNULLハンドルのオブジェクトはレイアウト上に表示されません。

レイヤーを削除するには、画面のルート・オブジェクトを1つのグループとして扱い、そのグループとレイヤー索引をこのプロシージャに渡します。

OG_Delete_Childの例

```
/*Suppose you have a several objects representing products
**in a warehouse, and you want to move one of the products
**from one warehouse to another.Your display may use a group
**comprised of the products to represent the inventory for each
**warehouse.To move a product from one warehouse to another,
**you would want to get the handle to the product object,
**delete it from one warehouse group, and add it to another
**warehouse group.
*/

/*Note that this procedure changes only the internal composition
**of the group objects.To move or change the appearance of the
**product object, you must use other Graphics Builder built-in procedures.
*/

PROCEDURE move_prod(warehouse1 IN OG_Object, warehouse2 IN
OG_Object, prod_index IN number) IS
  the_prod  OG_Object;
```

```
BEGIN
  the_prod:=OG_Get_Child(Warehouse1, prod_index);
  OG_Delete_Child(Warehouse1, prod_index, 1);
  OG_Insert_Child(Warehouse2, the_prod, OG_Last);
END;
```

OG_Delete_Cmptext

説明

このプロシージャで、指定したテキスト・オブジェクトから1つ以上の複数テキスト要素を削除します。「テキスト属性」に説明されているように、テキスト・オブジェクトのテキスト1行を複数テキスト要素といいます。

構文

```
PROCEDURE OG_Delete_Cmptext
  (text_hdl      OG_Object,
   indx          NUMBER,
   total          NUMBER,
   damage         BOOLEAN      := TRUE,
   update_bbox    BOOLEAN      := TRUE);
```

パラメータ

<i>text_hdl</i>	テキスト・オブジェクトのハンドル。
<i>indx</i>	複数テキスト要素リストから削除する最初の複数テキスト要素のテキスト・オブジェクト内の索引。
<i>total</i>	削除する複数テキスト要素の総数。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	パウンディング・ボックス更新フラグ。

使用上の注意

複数テキスト要素を削除すると、その要素を構成する単一テキスト要素がすべて削除されます。

OG_Delete_Cmptextの例

```
/*Suppose you use a text object to display messages to the user.
**A previous part of your application produced two-line messages,
**but the part of the display that is currently being used produces
**only one-line messages.You may want to delete the extraneous
**compound text element.
*/

PROCEDURE delete_msg_line(msg_object IN OG_Object,
```



```

line_index IN number) IS
BEGIN
    OG_Delete_Cmptext(Msg_Object, line_index, 1);
END;
```

OG_Delete_Point

説明

このプロシージャで、指定した多角形オブジェクトまたは折れ線オブジェクトから1つまたは複数の点を削除します。

構文

```

PROCEDURE OG_Delete_Point
(
    poly_hdl      OG_Object,
    indx          NUMBER,
    total         NUMBER,
    damage        BOOLEAN    := TRUE,
    update_bbox   BOOLEAN    := TRUE);
```

パラメータ

<i>poly_hdl</i>	多角形オブジェクトまたは折れ線オブジェクトのハンドル。
<i>indx</i>	点リストから削除する最初の点の索引。
<i>total</i>	削除する点の総数。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

Graphics Builderでオブジェクトを作成すると、そのオブジェクトを描画するときにマウスで選択した順序で点に初期索引値が付けられます（最初の点の初期索引値は0になります）。また、プログラムを使用して点を挿入すると、挿入時に指定した索引に従って索引値が付けられます。

OG_Delete_Pointの例

```

/*Suppose you have several polygons on a map, each of which
**connects the cities along a specific distribution route.
**If a city is transferred from one distribution route to another,
**you would want to get the point representing that city,
**delete it from one polygon, and add it to another polygon.
*/
```

```

PROCEDURE move_city(route1 IN OG_Object, route2 IN
```

```
OG_Object, city_index IN number) IS
    the_city  OG_Point;
BEGIN
    the_city:=OG_Get_Point(Route1, city_index);
    OG_Delete_Point(Route1, city_index, 1);
    OG_Insert_Point(Route2, OG_Last, the_city);
END;
```

OG_Delete_Property

説明

このプロシージャで、オブジェクトのユーザー定義プロパティを削除します。

構文

```
PROCEDURE OG_Delete_Property
    (object_hdl  OG_Object,
     prop_name   VARCHAR2);
```

パラメータ

<i>object_hdl</i>	オブジェクトのハンドル。
<i>prop_name</i>	削除するプロパティ名。

OG_Delete_Propertyの例

```
/* The following procedure deletes the property 'priority'
** from every child object in a named group:
*/

PROCEDURE example(group_name VARCHAR2) IS
    group_obj  OG_Object;
    child_count NUMBER;
    child_obj  OG_Object;
BEGIN
    group_obj := OG_Get_Object(Group_Name);
    child_count := OG_Get_Childcount(Group_Obj);

    FOR i IN 0..child_count LOOP
        child_obj := OG_Get_Child(Group_Obj, i);
        OG_Delete_Property(Child_Obj, 'priority');
    END LOOP;
END;
```

OG_Delete_Smptext

説明

このプロシージャで、指定したテキスト・オブジェクト内の指定した複数テキスト要素から1つ以上の単一テキスト要素を削除します。「テキスト属性」に説明されているように、複数テキスト要素の各テキスト文字列を単一テキスト要素といいます。

構文

```
PROCEDURE OG_Delete_Smptext
  (text_hdl      OG_Object,
   cmpindex      NUMBER,
   smpindex      NUMBER,
   total         NUMBER,
   damage        BOOLEAN    := TRUE,
   update_bbox   BOOLEAN    := TRUE);
```

パラメータ

<i>text_hdl</i>	テキスト・オブジェクトのハンドル。
<i>Cmpindex</i>	削除する単一テキスト要素を含む複数テキスト要素の索引。
<i>smpindex</i>	削除する最初の単一テキスト要素の索引。
<i>total</i>	削除する単一テキスト要素の総数。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

単一テキストを削除しても、そのテキストを含む複数テキスト要素の索引には影響しません。また、複数テキストが削除されることもありません。したがって、複数テキスト要素内の単一テキスト要素をすべて削除し、その複数テキスト要素を空のまま残せます。

OG_Delete_Smptextの例

```
/* Suppose you have created a message text object.To change
**the message it contains, you would delete the simple text element
**containing the current message and insert a new simple text element
**containing the new message.To maintain the font and other attributes,
**however, you first would want to get the simple text element into an
**attribute record.That way, you could modify only the text string,
**and leave the other attribute settings (such as font) unchanged.
*/
```

```
PROCEDURE put_msg(mess IN VARCHAR2) IS
  msgobj    OG_Object;
```

```
msgrec    OG_Smptext_Attr;
BEGIN
  msgobj := OG_Get_Object('msg');
  OG_Get_Smptext(msgobj, 0, 0, msgrec);
  OG_Delete_Smptext(msgobj, 0, 0, 1);
  msgrec.mask:= OG_STR_SMPTEXTA;
  msgrec.str:= mess;
  OG_Insert_Smptext(msgobj, msgrec, 0, OG_LAST);
END;
```

OG_Destroy (オブジェクト)

説明

このプロシージャで、指定したオブジェクトを破棄します。グループ・オブジェクトを破棄すると、そのグループの子もすべて破棄されます。

構文

```
PROCEDURE OG_Destroy
  (object_hdl  OG_Object,
   recurse     BOOLEAN    := FALSE,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);
```

パラメータ

<i>object_hdl</i>	破棄するオブジェクトのハンドル。
<i>Recurse</i>	再帰破棄フラグ。この引数はオプションです。指定しなければ、デフォルトでFALSEに設定されます。また、破棄するオブジェクトがグループの唯一の子でない場合は、この引数は無視されます。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

破棄するオブジェクトがグループの唯一の子の場合は、*recurse*の値をTRUEに設定すると、親グループも破棄されます。親グループの破棄は、そのオブジェクトのグループ・ツリーをさかのぼって実行されます（たとえば、そのオブジェクトの親がその親グループの唯一の子であれば、その親グループも破棄されます）。レイヤー上の最後のオブジェクトが破棄されると、そのレイヤーがアクティブになっているウィンドウがなければ、最終的には、そのレイヤーも破棄されます。

*recurse*がFALSEの場合は、*object_hdl*で指定したオブジェクトのみが破棄されます。

OG_Destroy (オブジェクト) の例

```
/* The following procedure destroys the specified object:
*/

PROCEDURE destroy_obj (obj_name VARCHAR2) IS
    object OG_Object;
BEGIN
    object := OG_Get_Object (Obj_Name);
    OG_Destroy (Object);
END;
```

OG_Draw

説明

このプロシージャで、指定したオブジェクトをレイアウト上に描画します。

構文

```
PROCEDURE OG_Draw
    (object_hdl OG_Object);
```

パラメータ

<i>object_hdl</i>	描画するオブジェクトへのハンドル。
-------------------	-------------------

使用上の注意

このプロシージャでは、オブジェクトを変更する他のプロシージャと異なり、レイアウト上の長方形の領域を描画しません。このプロシージャでは、指定したオブジェクトを描画するのみで、レイアウトには一切影響ありません。

このプロシージャを使用すると、Graphics Builderが確保した長方形の領域を再描画（必要以上に大きな領域になる場合があります）させずに、レイアウト上にオブジェクトを表示できます。

OG_Drawの例

```
/*Suppose you want to clone an object and have it appear on the
**layout smoothly, without causing a damage region to be redrawn.
**First, you would create the object by calling OG_Clone with a FALSE
**damage flag .Then, you can make the object appear on the layout
**by calling OG_Draw.
*/

PROCEDURE clone_object IS
```

```
the_object  OG_Object;  
new_object  OG_Object;  
BEGIN  
  the_object:=OG_Get_Object('My_Object');  
  new_object:=OG_Clone(The_Object, FALSE);  
  OG_Draw(new_object);  
END;
```

OG_Export_Drawing (図表)

説明

このプロシージャで、（隠れたレイヤーも含めて）レイアウト全体を1つの描画オブジェクトとしてエクスポートします。

構文

```
PROCEDURE OG_Export_Drawing  
(name          VARCHAR2,  
 repository     NUMBER,  
 format         NUMBER,  
 compression    NUMBER      :=  OG_No_Icompression);
```

パラメータ

<i>name</i>	描画オブジェクトのエクスポート先の名前。描画オブジェクトをデータベースに保存する場合は、この引数に描画オブジェクト名のみを指定します。描画オブジェクトをファイル・システムに保存する場合は、この引数に描画ファイルの絶対パス名または相対パス名を指定します。
<i>repository</i>	描画オブジェクトをファイル・システムとデータベースのどちらに保存するのかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db 描画オブジェクトをデータベースに保存します。 OG_Filessystem 描画オブジェクトをファイル・システムに保存します。
<i>format</i>	描画オブジェクトをエクスポートするときの形式を指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Cgm16_Dformat 描画オブジェクトをCGM 2バイト形式で保存します。 OG_Cgm32_Dformat 描画オブジェクトをCGM 4バイト形式で保存します。 OG_Oracle_Dformat 描画オブジェクトが他のOracle製品で使用されるOracleフォーマットで保存されていることを表します。

<i>compression</i>	<p>描画オブジェクトのイメージを圧縮するための圧縮タイプ。この引数の値には、次のビルトイン定数のいずれかを指定してください。</p> <p>OG_No_Icompression イメージを圧縮しません。</p> <p>OG_H3g_Icompression CCITT Group 3を使用してイメージを圧縮します。</p> <p>OG_G3fax_Icompression Group 3 FAX圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロ・イメージの場合のみです。</p> <p>OG_G4fax_Icompression Group 4 FAX圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロ・イメージの場合のみです。</p> <p>OG_Pack_Icompression PackBits圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロのTIFFイメージの場合のみです。</p> <p>OG_Lzwhdiff_Icompression 水平方向の差分を考慮したLZW圧縮を使用してイメージを圧縮します。</p> <p>OG_Lzwnohdiff_Icompression 水平方向の差分を考慮しないLZW圧縮を使用してイメージを圧縮します。</p> <p>OG_Jpeg_Lowest_Icompression JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が最も小さく品質が最も高くなります。</p> <p>OG_Jpeg_Low_Icompression JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が小さく品質が高くなります。</p> <p>OG_Jpeg_Medium_Icompression JPEG圧縮を使用してイメージを圧縮します。このタイプでは、中程度の圧縮率と品質になります。</p> <p>OG_Jpeg_High_Icompression JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が大きく品質が低くなります。</p> <p>OG_Jpeg_Highest_Icompression JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が最も大きく品質が最も低くなります。</p>
--------------------	--

OG_Export_Drawing (図表) の例

```

/*Suppose you want to export the display contents to the CGM file
**'my_draw' so that you can later import it into some other application.
** The following procedure does this:
*/
PROCEDURE export_the_drawing IS
BEGIN
    OG_Export_Drawing('My_Draw', OG_FileSystem, OG_Cgml6_Dformat);
END;
```

OG_Export_Drawing (オブジェクト/レイヤー)

説明

このプロシージャで、指定したオブジェクトまたはレイヤーを1つの描画オブジェクトとしてエクスポートします。

構文

```
PROCEDURE OG_Export_Drawing
  (name          VARCHAR2,
   repository    NUMBER,
   format        NUMBER,
   object_hdl    OG_Object,
   compression   NUMBER      := OG_No_Icompression);
```

パラメータ

<i>name</i>	描画オブジェクトのエクスポート先の名前。描画オブジェクトをデータベースに保存する場合は、この引数に描画オブジェクト名のみを指定します。描画オブジェクトをファイル・システムに保存する場合は、この引数に描画ファイルの絶対パス名または相対パス名を指定します。
<i>Repository</i>	描画オブジェクトをファイル・システムとデータベースのどちらに保存するのかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db 描画オブジェクトをデータベースに保存します。 OG_Fileystem 描画オブジェクトをファイル・システムに保存します。
<i>format</i>	描画オブジェクトをエクスポートするときの形式を指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Cgm16_Dformat 描画オブジェクトをCGM 2バイト形式で保存します。 OG_Cgm32_Dformat 描画オブジェクトをCGM 4バイト形式で保存します。 OG_Oracle_Dformat 描画オブジェクトが他のOracle製品で使用されるOracleフォーマットで保存されていることを表します。
<i>object_hdl</i>	エクスポートするオブジェクトのハンドル。エクスポートするオブジェクトは、グループでもレイヤーのオブジェクト・ハンドルでも構いません。オブジェクトとそのオブジェクトの子孫がすべてエクスポートされます。エクスポートするレイヤーを指定するには、OG_Get_Objectを使用してそのレイヤーのオブジェクト・ハンドルを指定します。

<i>Compression</i>	<p>描画オブジェクトのイメージを圧縮するための圧縮タイプ。この引数の値には、次のビルトイン定数のいずれかを指定してください。</p>	
	OG_No_Icompression	イメージを圧縮しません。
	OG_H3g_Icompression	CCITT Group 3を使用してイメージを圧縮します。
	OG_G3fax_Icompression	Group 3 FAX圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロ・イメージの場合のみです。
	OG_G4fax_Icompression	Group 4 FAX圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロ・イメージの場合のみです。
	OG_Pack_Icompression	PackBits圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロのTIFFイメージの場合のみです。
	OG_Lzwdiff_Icompression	水平方向の差分を考慮したLZW圧縮を使用してイメージを圧縮します。
	OG_Lzwnohdiff_Icompression	水平方向の差分を考慮しないLZW圧縮を使用してイメージを圧縮します。
	OG_Jpeg_Lowest_Icompression	JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が最も小さく品質が最も高くなります。
	OG_Jpeg_Low_Icompression	JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が小さく品質が高くなります。
	OG_Jpeg_Medium_Icompression	JPEG圧縮を使用してイメージを圧縮します。このタイプでは、中程度の圧縮率と品質になります。
	OG_Jpeg_High_Icompression	JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が大きく品質が低くなります。
	OG_Jpeg_Highest_Icompression	JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が最も大きく品質が最も低くなります。

OG_Export_Drawing (オブジェクト/レイヤー) の使用例

```

/* Suppose you want to export the contents of 'layer0' to the CGM file
** 'my_draw' so that you can later import it into some other application.
** The following procedure does this:
**/

*/

PROCEDURE export_the_drawing IS
    the_layer  OG_Object;
BEGIN

```

```
the_layer:=OG_Get_Object('Layer0');
OG_Export_Drawing('My_Draw', OG_FileSystem,
OG_Cgm16_Dformat, the_layer);
END;
```

OG_Export_Drawing (ウィンドウ)

説明

このプロシージャで、指定したウィンドウに表示されている内容を1つの描画オブジェクトとしてエクスポートします。

構文

```
PROCEDURE OG_Export_Drawing
(name          VARCHAR2,
 repository    NUMBER,
 format        NUMBER,
 window_hdl    OG_Window,
 compression   NUMBER      :=  OG_No_Icompression);
```

パラメータ

<i>name</i>	描画オブジェクトのエクスポート先の名前。描画オブジェクトをデータベースに保存する場合は、この引数に描画オブジェクト名のみを指定します。描画オブジェクトをファイル・システムに保存する場合は、この引数に描画ファイルの絶対パス名または相対パス名を指定します。
<i>Repository</i>	描画オブジェクトをファイル・システムとデータベースのどちらに保存するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db 描画オブジェクトをデータベースに保存します。 OG_FileSystem 描画オブジェクトをファイル・システムに保存します。
<i>format</i>	描画オブジェクトをエクスポートするときの形式を指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Cgm16_Dformat 描画オブジェクトをCGM 2バイト形式で保存します。 OG_Cgm32_Dformat 描画オブジェクトをCGM 4バイト形式で保存します。 OG_Oracle_Dformat 描画オブジェクトが他のOracle製品で使用するOracleフォーマットで保存されていることを表します。
<i>window_hdl</i>	エクスポートする描画オブジェクトを含むウィンドウのハンドル。そのウィンドウに表示されているレイヤーがすべてエクスポートされます。

<i>Compression</i>	<p>描画オブジェクトのイメージを圧縮するための圧縮タイプ。この引数の値には、次のビルトイン定数のいずれかを指定してください。</p> <p>OG_G3fax_lcompression Group 3 FAX圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロ・イメージの場合のみです。</p> <p>OG_G4fax_lcompression Group 4 FAX圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロ・イメージの場合のみです。</p> <p>OG_H3g_lcompression CCITT Group 3を使用してイメージを圧縮します。</p> <p>OG_Jpeg_High_lcompression JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が大きく品質が低くなります。</p> <p>OG_Jpeg_Highest_lcompression JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が最も大きく品質が最も低くなります。</p> <p>OG_Jpeg_Low_lcompression JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が小さく品質が高くなります。</p> <p>OG_Jpeg_Lowest_lcompression JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が最も小さく品質が最も高くなります。</p> <p>OG_Jpeg_Medium_lcompression JPEG圧縮を使用してイメージを圧縮します。このタイプでは、中程度の圧縮率と品質になります。</p> <p>OG_Lzwldiff_lcompression 水平方向の差分を考慮したLZW圧縮を使用してイメージを圧縮します。</p> <p>OG_Lzwnohdiff_lcompression 水平方向の差分を考慮しないLZW圧縮を使用してイメージを圧縮します。</p> <p>OG_No_lcompression イメージを圧縮しません。</p> <p>OG_Pack_lcompression PackBits圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロのTIFFイメージの場合のみです。</p>
--------------------	--

OG_Export_Drawing (ウィンドウ) の例

```

/* Suppose you want to export the contents of the 'Main Layout'
** window to the CGM file 'my_draw' so that you can later import it
** into some other application. The following procedure does this:
*/
PROCEDURE export_the_drawing IS
    the_window    OG_Window;
BEGIN
    the_window:=OG_Get_Window('Main Layout');
    OG_Export_Drawing('My_Draw', OG_FileSystem,
OG_Cgm16_Dformat, the_window);

```

END;

OG_Export_Image

説明

このプロシージャで、Graphics Builderオブジェクトを1つのイメージとしてエクスポートします。

構文

```
PROCEDURE OG_Export_Image
  (name          VARCHAR2,
   repository    NUMBER,
   format        NUMBER,
   image_hdl     OG_Object,
   compression   NUMBER      := OG_No_Icompression);
```

パラメータ

<i>name</i>	イメージのエクスポート先の名前。イメージをデータベースに保存する場合は、この引数にイメージの名前のみを指定します。イメージをファイル・システムに保存する場合は、この引数にイメージ・ファイルの絶対パス名または相対パス名を指定します。
<i>Repository</i>	イメージをファイル・システムとデータベースのどちらに保存するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db イメージをデータベースに保存します。 OG_FileSystem イメージをファイル・システムに保存します。
<i>format</i>	イメージをエクスポートするときの形式を指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Bmp_lformat イメージをWindows/OS2のビットマップ形式で保存します。 G_Cals_lformat イメージをCALS Type 1のラスター形式で保存します。 OG_Gif_lformat イメージをCompuServeのGIF形式で保存します。GIFファイルはOG_Lzwhdiff_lcompressionを使用して圧縮する必要があります。 OG_Jfif_lformat イメージをJPEGファイル・イメージ・フォーマットで保存します。 OG_Pict_lformat イメージをMacintoshのPICT形式で保存します。 OG_Ras_lformat イメージをSUNのラスター形式で保存します。 OG_Tiff_lformat イメージをTagのイメージ・ファイル・フォーマットで保存します。
<i>image_hdl</i>	エクスポートするイメージ・オブジェクトのハンドル。すべてのGraphics Builderオブジェクトがエクスポートできます。

<i>Compression</i>	使用する圧縮タイプ。この引数の値には、次のビルトイン定数のいずれかを指定してください。	
OG_G3fax_lcompression		Group 3 FAX圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロ・イメージの場合のみです。
OG_G4fax_lcompression		Group 4 FAX圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロ・イメージの場合のみです。
OG_H3g_lcompression		CCITT Group 3を使用してイメージを圧縮します。
OG_Jpeg_High_lcompression		JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が大きく品質が低くなります。
OG_Jpeg_Highest_lcompression		JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が最も大きく品質が最も低くなります。
OG_Jpeg_Low_lcompression		JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が小さく品質が高くなります。
OG_Jpeg_Lowest_lcompression		JPEG圧縮を使用してイメージを圧縮します。このタイプは、圧縮率が最も小さく品質が最も高くなります。
OG_Jpeg_Medium_lcompression		JPEG圧縮を使用してイメージを圧縮します。このタイプでは、中程度の圧縮率と品質になります。
OG_Lzw_hdiff_lcompression		水平方向の差分を考慮したLZW圧縮を使用してイメージを圧縮します。この圧縮タイプはGIFファイルに使用します。
OG_Lzw_nohdiff_lcompression		水平方向の差分を考慮しないLZW圧縮を使用してイメージを圧縮します。
OG_No_lcompression		イメージを圧縮しません。
OG_Pack_lcompression		PackBits 圧縮を使用してイメージを圧縮します。この圧縮タイプを指定できるのは、モノクロのTIFFイメージの場合のみです。

OG_Export_Imageの例

```

/* Suppose you want to export the image named 'image0' to the TIFF file
**'my_image' so that you can later import it into some other application
**The following procedure does this:
*/
PROCEDURE export_the_image IS
    the_image  OG_Object;
BEGIN
    the_image:=OG_Get_Object('image0');
    OG_Export_Image('my_image', OG_FileSystem,
        OG_Tiff_Iformat, the_image);

```

END;

OG_Get_Char_Property

説明

このファンクションを使用して、オブジェクトのユーザー定義CHARプロパティの値を取得します。

構文

```
FUNCTION OG_Get_Char_Property
  (object_hdl  OG_Object,
   prop_name   VARCHAR2)
RETURN VARCHAR2;
```

パラメータ

<i>object_hdl</i>	取得したいプロパティを含むオブジェクトのハンドル。
<i>prop_name</i>	値を取得するプロパティの名前。

戻り値

指定したプロパティの値。

OG_Get_Char_Propertyの例

```
/*The following procedure gets the 'status' property
**in each child object in a group, and then changes
**the object's color if the status is 'obsolete':
*/
PROCEDURE example(group_name VARCHAR2) IS
  group_obj  OG_Object;
  child_count NUMBER;
  child_obj  OG_Object;
  stat       VARCHAR2(10);
BEGIN
  group_obj := OG_Get_Object(Group_Name);
  child_count := OG_Get_Childcount(Group_Obj);

  FOR i IN 0..child_count LOOP
    child_obj := OG_Get_Child(Group_Obj, i);
    stat := OG_Get_Char_Property(Child_Obj, 'status');
    IF stat = 'obsolete' THEN
      OG_Set_Fillcolor(Child_Obj, 'red');
    END IF;
  END LOOP;
```

END;

OG_Get_Child

説明

このファンクションを使用すると、グループ・オブジェクト内の子オブジェクトのハンドルが戻ります。

構文

```
FUNCTION OG_Get_Child
  (group_hdl  OG_Object,
   indx       NUMBER)
RETURN OG_Object;
```

パラメータ

<i>group_hdl</i>	その子を含むグループ・オブジェクトのハンドル。
<i>indx</i>	グループの子リスト内のオブジェクトの索引。このオブジェクトのハンドルが戻ります。

戻り値

グループ内の指定した子オブジェクトのハンドル。

OG_Get_Childの例

```
/*Suppose you have a several objects representing products
**in a warehouse, and you want to move one of the products
**from one warehouse to another.Your display may use a group
**comprised of the products to represent the inventory for each
**warehouse.To move a product from one warehouse to another,
**you would want to get the handle to the product object, delete it
**from one warehouse group, and add it to another warehouse group.
*/

PROCEDURE move_prod(warehouse1 IN OG_Object, warehouse2 IN
OG_Object, prod_index in number) IS
  the_prod  OG_Object;
BEGIN
  the_prod:=OG_Get_Child(Warehouse1, prod_index);
  OG_Delete_Child(Warehouse1, prod_index, 1);
  OG_Insert_Child(Warehouse2, the_prod, OG_Last);
END;
```

```
/*Note that this procedure changes only the internal composition
**of the group objects.To move or change the appearance of the
**product object, you must use other Graphics Builder built-in procedures.
*/
```

OG_Get_Cmptext

説明

このプロシージャで、指定した複数テキスト要素の属性値を取得して、指定した複数テキスト属性レコードの対応するフィールドに割り当てます。「テキスト属性」に説明されているように、テキスト・オブジェクトのテキスト1行を複数テキスト要素といいます。

構文

```
PROCEDURE OG_Get_Cmptext
(
  text_hdl      OG_Object,
  indx          NUMBER,
  attr          IN OUT OG_Cmptext_Attr);
```

パラメータ

<i>text_hdl</i>	テキスト・オブジェクトのハンドル。
<i>indx</i>	取り出す属性のある複数テキスト要素の、複数テキスト要素内の索引。
<i>attr</i>	複数テキスト要素の属性を受け取る複数テキスト属性レコード。

使用上の注意

属性レコードのマスク属性の値で指定された属性値のみが取り出されます。したがって、OG_Get_Cmptextをコールしても、マスクが設定されていない属性レコードのフィールドには影響しません。

OG_Get_Cmptextの例

```
/* Suppose you want to determine how many simple text elements
**compose the first compound text element within a text object.
**Knowing this, you can loop through and examine each simple text element.
*/

FUNCTION how_many(my_text IN OG_Object) RETURN NUMBER IS
  ctext_rec  OG_Cmptext_Attr;
BEGIN
  ctext_rec.mask:=OG_Stcount_Cmptexta;
  OG_Get_Cmptxt(My_Text, 0, ctext_rec);
  RETURN(ctext_rec.stcount);
END;
```


OG_Get_Date_Property

説明

このファンクションを使用して、オブジェクトのユーザー定義DATEプロパティの値を取得します。

構文

```
FUNCTION OG_Get_Date_Property
  (object_hdl  OG_Object,
   prop_name   VARCHAR2,
   date_fmt    VARCHAR2  := 'DD-MON-YY')
RETURN DATE;
```

パラメータ

<i>object_hdl</i>	取得するプロパティを含むオブジェクトのハンドル。
<i>prop_name</i>	値を取得するプロパティの名前。
<i>date_fmt</i>	OG_Set_Propertyで日付プロパティを設定するのに使用した日付書式マスク。

戻り値

指定したプロパティの値。

OG_Get_Date_Propertyの例

```
/*The following procedure gets the 'due_date' property in each child object
**in a group, and then changes the object's color if the due date has past:
*/

PROCEDURE example(group_name VARCHAR2) IS
  group_obj  OG_Object;
  child_count NUMBER;
  child_obj  OG_Object;
  due        DATE;
BEGIN
  group_obj := OG_Get_Object(Group_Name);
  child_count := OG_Get_Childcount(Group_Obj);

  FOR i IN 0..child_count-1 LOOP
    child_obj := OG_Get_Child(Group_Obj, i);
    due := OG_Get_Date_Property(Child_Obj, 'due_date');
    IF due < sysdate THEN
      OG_Set_Fillcolor(Child_Obj, 'red');
    END IF;
  END LOOP;
```

END;

OG_Get_Num_Property

説明

このファンクションを使用して、オブジェクトのユーザー定義数値プロパティの値を取得します。

構文

```
FUNCTION OG_Get_Num_Property
  (object_hdl  OG_Object,
   prop_name   VARCHAR2)
RETURN NUMBER;
```

パラメータ

<i>object_hdl</i>	取得するプロパティを含むオブジェクトのハンドル。
<i>prop_name</i>	値を取得するプロパティの名前。

戻り値

指定したプロパティの値。

OG_Get_Num_Propertyの例

```
/* The following procedure gets the 'priority' property in each child object
**in a group, and then sets the priority to one greater than its current
value:
*/
PROCEDURE example(group_name VARCHAR2) IS
  group_obj  OG_Object;
  child_count NUMBER;
  child_obj  OG_Object;
  current_p  NUMBER;
BEGIN
  group_obj := OG_Get_Object(Group_Name);
  child_count := OG_Get_Childcount(Group_Obj);

  FOR i IN 0..child_count-1 LOOP
    child_obj := OG_Get_Child(Group_Obj, i);
    current_p := OG_Get_Num_Property(Child_Obj, 'priority');
    OG_Set_Property(Child_Obj, 'priority', current_p + 1);
  END LOOP;

END;
```

OG_Get_Object

説明

このファンクションを使用して、円弧、チャート、グループ、イメージ、線、多角形、四角形、丸い四角形、記号、テキストなどのオブジェクトを取り出します。オブジェクトを取り出す前に、Builder内またはプログラムを使用して、そのオブジェクトを作成し、名前を付けておく必要があります。指定したオブジェクトが存在しなければ、このファンクションを使用するとNULLハンドルが戻ります。

構文

```
FUNCTION OG_Get_Object  
    (object_name VARCHAR2)  
RETURN OG_Object;  
  
FUNCTION OG_Get_Object  
    (object_name VARCHAR2,  
     root_hdl     OG_Object)  
RETURN OG_Object;
```

パラメータ

<i>object_name</i>	オブジェクトの名前。このオブジェクトのハンドルが戻ります。 注意: OBJECT_NAME 引数では大文字と小文字が区別されます。
<i>root_hdl</i>	グループ・ツリーを検索したいオブジェクトのハンドル。

戻り値

指定したオブジェクトのハンドル。

使用上の注意

*root_hdl*を指定しなければ、Graphics Builderによって画面上の実際のルート・オブジェクトから検索が開始されます。つまり、画面上から名前を持つすべてのオブジェクトが検索されます。一方、*root_hdl*を指定すると、Graphics Builderによってグループ・ツリーのそのオブジェクトの下から検索されます。なお、検索パスに同じ名前のオブジェクトが複数存在すると、正しい結果は戻りません。

*object_name*にレイヤー名を指定すると、このファンクションはレイヤーをグループ・オブジェクトとして処理し、そのグループのハンドルが戻ります。このハンドルを使用すれば、グループ関連のサブプログラム(OG_Insert_ChildやOG_Delete_Childなど)を使用してレイヤー上のオブジェクトが操作できます。

OG_Get_Objectの例

```
/* Suppose you have a map of the world and you want to change
```

```

**the color of one of the countries.First, you would get the handle
**to the country object, then you would change its color.
*/

PROCEDURE color_country(country_name) IS
    my_object    OG_Object;
    obj_record   OG_Graphic_Ca;
BEGIN
    my_object:=OG_Get_Object(Country_Name);
    obj_record.graphic_caob.mask:=OG_None_Generica;
    obj_record.graphic_caoh.mask:=OG_Ffcolor_Graphica;
    obj_record.graphic_caoh.ffcolor:='red';
    OG_Set_Attr(My_Object, obj_record);
END;
```

OG_Get_Point

説明

このファンクションを使用すると、指定したオブジェクト上の点のx座標とy座標を含むレコードが戻ります。

構文

```

FUNCTION OG_Get_Point
    (object_hdl  OG_Object,
     indx        NUMBER
     rotated     BOOLEAN      := FALSE)
RETURN OG_Point;
```

パラメータ

<i>object_hdl</i>	オブジェクトのハンドル。
<i>indx</i>	戻される点の索引。
<i>rotated</i>	オブジェクトに適用されている回転角を、戻された点に反映させるかどうかを指定します。

戻り値

指定した点の位置。

使用上の注意

多角形: オブジェクトの*indx*番目の点が戻ります。

円弧、チャート、四角形、丸い四角形: 索引0では左上角、索引1では右上角、索引2では右下角、索引3では左下角の点が戻ります。

テキスト: 索引0では左上角、索引1では右上角、索引2では右下角、索引3では左下角の点が戻ります。

行: 索引0では始点、索引1では終点が戻ります。

イメージ、グループ、記号: 適用されません。

Graphics Builderでオブジェクトを作成すると、そのオブジェクトを描画するときにマウスで選択した順序で点に初期索引値が付けられます（最初の点の初期索引値は0になります）。また、プログラムを使用して点を挿入すると、挿入時に指定した索引に従って索引値が付けられます。

OG_Get_Pointの例

```
/* Suppose you have several polygons on a map, each of which connects
**the cities within a specific distribution area.If a city is transferred
from one
**distribution area to another, you would want to get a handle to the point
**representing that city, delete it from one polygon, and add it to another
polygon.
*/

PROCEDURE move_city(area1 IN OG_Object, area2 IN OG_Object,
    city_index NUMBER) IS
    the_city OG_Point;
BEGIN
    the_city:=OG_Get_Point(Area1, city_index);
    OG_Delete_Point(Area1, city_index, 1);
    OG_Insert_Point(Area2, OG_Last, the_city);
END;
```

OG_Get_Smptext

説明

このプロシージャで、指定した複数テキスト要素内および指定したテキスト・オブジェクト内の指定した単一テキスト要素の属性値を取得します。取得した属性は、さらに、指定した単一テキスト属性レコードの対応するフィールドに割り当てられます。「テキスト属性」に説明されているように、複数テキスト要素の各テキスト文字列を単一テキスト要素といいます。

構文

```
PROCEDURE OG_Get_Smptext
    (text_hdl          OG_Object,
    cmpindex          NUMBER,
```

```

smpindex      NUMBER,
attr          IN OUT  OG_Smptext_Attr);

```

パラメータ

<i>text_hdl</i>	テキスト・オブジェクトのハンドル。
<i>Cmpindx</i>	属性を取り出す単一テキスト要素を含む複数テキスト要素のテキスト・オブジェクト内の索引。
<i>Smpindex</i>	属性を取り出す単一テキスト要素の複数テキスト要素内の索引。
<i>attr</i>	単一テキスト要素の属性を受け取る単一テキスト属性レコード。

使用上の注意

属性レコードのマスク属性の値で指定された属性値のみが取り出されます。したがって、OG_Get_Smptextをコールしても、マスクが設定されていない属性レコードのフィールドには影響しません。

OG_Get_Smptextの例

```

/* Suppose you have created a message text object.To change the
**message it contains, you would delete the simple text element containing
**the current message and insert a new simple text element containing the
**new message.To maintain the font and other attributes, however,
**you first would want to get the simple text element into an attribute
record.
**That way you could modify only the text string, and leave the other attribute
**settings (such as font) unchanged.
*/

PROCEDURE put_msg (mess IN VARCHAR2) IS
    msgobj  OG_Object;
    msgrec  OG_Smptext_Attr;
BEGIN
    msgobj:=OG_Get_Object('Msg');
    msgrec.mask:=OG_Font_Smptexta+
                OG_Color_Smptexta;
    msgrec.font.mask:=OG_All_Fonta;
    OG_Get_Smptext(Msgobj, 0, 0, msgrec);
    OG_Delete_Smptext(Msgobj, 0, 0, 1, FALSE);
    msgrec.mask:=msgrec.mask + OG_Str_Smptexta;
    msgrec.str:=mess;
    OG_Insert_Smptext(Msgobj, msgrec, 0, OG_Last);
END;
```

OG_Import_Drawing

説明

このプロシージャを使用して、描画オブジェクトをインポートします。描画オブジェクトのハンドルが戻ります。

構文

```
FUNCTION OG_Import_Drawing
  (name          VARCHAR2,
   repository    NUMBER,
   format        NUMBER,
   use_colors    BOOLEAN,
   damage        BOOLEAN := TRUE,
   update_bbox   BOOLEAN := TRUE)
RETURN OG_Object;

FUNCTION OG_Import_Drawing
  (name          VARCHAR2,
   repository    NUMBER,
   format        NUMBER,
   use_colors    BOOLEAN,
   parent_hdl    OG_Object,
   damage        BOOLEAN := TRUE,
   update_bbox   BOOLEAN := TRUE)
RETURN OG_Object;
```

パラメータ

<i>name</i>	描画オブジェクトの名前。描画オブジェクトがデータベースに保存されている場合は、この引数に描画オブジェクト名のみを指定します。描画オブジェクトがファイル・システムに保存されている場合は、この引数に描画ファイルの絶対パス名または相対パス名を指定します。
<i>Repository</i>	<p>描画オブジェクトがファイル・システムとデータベースのどちらに保存されているかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。</p> <p>OG_Db 描画オブジェクトがデータベースに保存されていることを表します。</p> <p>OG_FileSystem 描画オブジェクトがファイル・システムに保存されていることを表します。</p>
<i>format</i>	<p>描画オブジェクトが保存されている形式を指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。</p> <p>OG_Cgm_Dformat 描画オブジェクトがCGM形式(2バイトまたは4バイト)で保存されていることを表します。</p> <p>OG_Oracle_Dformat 描画オブジェクトが他のOracle製品で使用するOracleフォーマットで保存されていることを表します。</p>

<i>use_colors</i>	描画オブジェクトのカラー・パレットを使用するかどうかを指定します。この引数をTRUEにすると、描画オブジェクトのパレットが使用されます。FALSEにすると、Graphics Builderのデフォルトのカラー・パレットが使用されます。
<i>parent_hdl</i>	インポートした描画オブジェクトを子として挿入するグループ・オブジェクトのハンドル。この引数を指定しなければ、描画オブジェクトはルート・オブジェクトの子として挿入されます（ただし、描画オブジェクトをレイヤーとして認識させるには、そのオブジェクトをアクティブにまたは表示する必要があります）。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

戻り値

インポートされた描画オブジェクトのハンドル。

OG_Import_Drawingの例

```
/* Suppose you want to import the contents of the CGM file 'my_draw'
**onto the layer 'layer0'.You can get the object handle to the layer,
**then use that for parent_hdl.The following procedure does this:
*/
PROCEDURE import_the_drawing IS
    the_layer  OG_Object;
    dummy  OG_Object;
BEGIN
    the_layer:=OG_Get_Object('Layer0');
    dummy:=OG_Import_Drawing('My_Draw', OG_Filessystem,
OG_Cgml6_Dformat, FALSE, the_layer);
END;
```

OG_Import_Image

説明

このプロシージャを使用して、アクティブ・ウィンドウのアクティブ・レイヤー上にイメージをインポートします。

構文

```
FUNCTION OG_Import_Image
    (name          VARCHAR2,
    repository     NUMBER,
    format         NUMBER,
    damage         BOOLEAN   :=  TRUE,
    update_bbox    BOOLEAN   :=  TRUE)
RETURN OG_Object;
```


パラメータ

<i>name</i>	イメージの名前。イメージがデータベースに保存されている場合は、この引数にイメージ名のみを指定します。イメージがファイル・システムに保存されている場合は、この引数にイメージ・ファイルの絶対パス名または相対パス名を指定します。
<i>Repository</i>	<p>イメージがファイル・システムとデータベースのどちらに保存されているかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。</p> <p>OG_Db イメージがデータベースに保存されていることを表します。</p> <p>OG_FileSystem イメージがファイルシステムに保存されていることを表します。</p>
<i>format</i>	<p>イメージが保存されている形式を指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。</p> <p>OG_Any_lformat Graphics Builderでイメージ形式が自動的に判別されます。 注意: イメージが現在使用されていないOracleフォーマットでエクスポートされている場合に、この形式を指定します。</p> <p>OG_Bmp_lformat イメージがBMP形式で保存されていることを表します。</p> <p>OG_Cals_lformat イメージがCALS形式で保存されていることを表します。</p> <p>OG_Gif_lformat イメージがGIF形式で保存されていることを表します。</p> <p>OG_Jfif_lformat イメージがJFIF形式で保存されていることを表します。</p> <p>OG_Oracle_Sformat イメージが他のOracle製品で使用するOracleフォーマットで保存されていることを表します。</p> <p>OG_Pcd_lformat イメージがPCD形式で保存されていることを表します。</p> <p>OG_Pcx_lformat イメージがPCX形式で保存されていることを表します。</p> <p>OG_Pict_lformat イメージがPICT形式で保存されていることを表します。</p> <p>OG_Ras_lformat イメージがSUNのラスター形式で保存されていることを表します。</p> <p>OG_Tiff_lformat イメージがTIFF形式で保存されていることを表します。</p>
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

戻り値

インポートされたイメージのハンドル。

OG_Import_Imageの例

```

/* Suppose you want to import the contents of the TIFF file 'my_image'
**onto the layer 'layer0'.The following procedure does this:
*/
PROCEDURE import_the_image IS
    the_image  OG_Object;
BEGIN

```

```
OG_Activate_Layer(OG_Get_Layer('Layer0'));
the_image:=OG_Import_Image('My_Image', OG_Filesystem,
    OG_Tiff_Iformat);
END;
```

OG_Insert_Child

説明

このプロシージャで、指定したグループ・オブジェクトに子オブジェクトを挿入します。挿入されるオブジェクトがすでに別のグループ・オブジェクトの子になっている場合は、Graphics Builderによってその子が現在の親から自動的に削除されます。

構文

```
PROCEDURE OG_Insert_Child
  (group_hdl      OG_Object,
   child_hdl      OG_Object,
   indx           NUMBER,
   damage         BOOLEAN    :=  TRUE,
   update_bbox    BOOLEAN    :=  TRUE);
```

パラメータ

<i>group_hdl</i>	子を挿入するグループ・オブジェクトのハンドル。
<i>child_hdl</i>	子として挿入するオブジェクトのハンドル。
<i>indx</i>	新しい子を挿入するグループの子リスト内の位置の索引。この引数は、0から <i>n</i> までの整数であることが必要です。ここで、 <i>n</i> は新しい子を挿入する前のグループの子の数です。次のビルトイン定数のいずれかを指定することもできます。 OG_First グループの最初の子（索引 = 0）としてオブジェクトを挿入します。 OG_Last グループの最後の子（索引 = 挿入前のグループの子の数）としてオブジェクトを挿入します。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

レイヤーを1つのグループ・オブジェクトとして扱い、このプロシージャにそのオブジェクトのハンドルを渡せば、レイヤーにオブジェクトを挿入できます。また、画面のルート・オブジェクトにグループ・オブジェクトを挿入すれば、新しいレイヤーを作成できます。ただし、Graphics Builderにそのグループをレイヤーとして認識させるには、OG_Activate_LayerまたはOG_Show_Layerをコールして明示的に表示する必要があります。

なお、Graphics Builderではグループ・オブジェクトで発生するループはチェックしません。このようなグループは、グループをそのグループの子孫の子として挿入したときに発生します。このような場合、挿入の影響を受けるオブジェクトのバウンディング・ボックスを更新しようとしたときに、Graphics Builderが無限ループに入ります。

また、このプロシージャで変更されるのは、グループ・オブジェクトの内部構成のみです。したがって、製品オブジェクトの外観を移動または変更するには、Graphics Builderの他のビルトイン・プロシージャを使用する必要があります。

OG_Insert_Childの例

```
/* Suppose you have a several objects representing products in a warehouse,
**and you want to move one of the products from one warehouse to another.
**Your display may use a group comprised of the products to represent the
**inventory for each warehouse.To move a product from one warehouse to
**another, you would want to get the handle to the product object, delete
it
**from one warehouse group, and add it to another warehouse group.
*/

PROCEDURE move_prod (warehouse1 IN OG_Object, warehouse2 IN
  OG_Object, prod_index IN number) IS
  the_prod  OG_Object;
BEGIN
  the_prod:=OG_Get_Child(Warehouse1, prod_index);
  OG_Delete_Child(Warehouse1, prod_index, 1);
  OG_Insert_Child(Warehouse2, the_prod, OG_Last);
END;
```

OG_Insert_Cmptext

説明

このプロシージャで、指定したテキスト・オブジェクトに新しい複数テキスト要素を挿入します。「テキスト属性」に説明されているように、テキスト・オブジェクトのテキスト1行を複数テキスト要素といいます。

構文

```
PROCEDURE OG_Insert_Cmptext
(text_hdl  OG_Object,
indx       NUMBER,
damage     BOOLEAN    := TRUE,
update_bbox BOOLEAN    := TRUE);
```

パラメータ

<i>text_hdl</i>	テキスト・オブジェクトのハンドル。
-----------------	-------------------

<i>indx</i>	新しい複数テキスト要素を挿入するテキスト・オブジェクト内の複数テキスト要素リスト内の位置の索引。この引数は、0から <i>n</i> までの整数であることが必要です。ここで、 <i>n</i> は挿入前のテキスト・オブジェクト内の複数テキスト要素の数です。次のビルトイン定数のいずれかを指定することもできます。 OG_First テキスト・オブジェクトの最初（索引 = 0）に新しい複数テキスト要素を挿入します。 OG_Last テキスト・オブジェクトの最後（索引 = 挿入前のテキスト・オブジェクト内の複数テキスト要素の数）に新しい複数テキスト要素を挿入します。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

複数テキスト要素の属性は、Graphics Builderによって自動的に設定されます。したがって、新しい複数テキストを作成する際は、複数テキスト属性レコードを指定する必要はありません。（逆に、要素の属性を取得する場合は、属性を受け取るための複数テキスト属性レコードをOG_Get_Cmptextに指定する必要があります。）

OG_Insert_Cmptextの例

```
/* Suppose you want to create a text object that contains a message for
**the user.The following function will create the object, insert a compound
text
**element, then insert a simple text element that contains the text of
the message.
*/

PROCEDURE make_text (the_message IN VARCHAR2) IS
    text_rec    OG_Text_Ca;
    text_obj    OG_Object;
    smp_rec     OG_Smptext_Attr;
BEGIN
    text_rec.text_caob.mask:=OG_None_Generica;
    text_rec.text_caoh.mask:=OG_None_Graphica;
    text_rec.text_caot.mask:=OG_None_Texta;
    text_obj:=OG_Make(Text_Rec);
    OG_Insert_Cmptext(Text_Obj, OG_Last);
    smp_rec.mask:=OG_Str_Smptexta;
    smp_rec.str:=the_message;
    OG_Insert_Smptext(Text_Obj, smp_rec, 0, OG_Last);
END;
```

OG_Insert_Point

説明

このプロシージャで、指定した多角形オブジェクトに新しい点を挿入します。

構文

```
PROCEDURE OG_Insert_Point
  (poly_hdl      OG_Object,
   indx          NUMBER,
   pt            OG_Point,
   damage        BOOLEAN    := TRUE,
   update_bbox   BOOLEAN    := TRUE);
```

パラメータ

<i>poly_hdl</i>	多角形オブジェクトまたは折れ線オブジェクトのハンドル。
<i>indx</i>	新しい点を挿入する点リスト内の位置の索引。この引数は、0から <i>n</i> までの整数である必要があります。ここで、 <i>n</i> は新しい点を挿入する前のオブジェクト内の点の数です。次のビルトイン定数のいずれかを指定することもできます。 OG_First オブジェクトの最初（索引 = 0）に新しい点を挿入します。 OG_Last オブジェクトの最後（索引 = 挿入前のテキスト・オブジェクト内の点の数）に新しい点を挿入します。
<i>pt</i>	挿入する点のx座標とy座標が入ったレコード。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

Graphics Builderでオブジェクトを作成すると、そのオブジェクトを描画するときにマウスで選択した順序で点に初期索引値が付けられます（最初の点の初期索引値は0になります）。また、プログラムを使用して点を挿入すると、挿入時に指定した索引に従って索引値が付けられます。

OG_Insert_Pointの例

```
/* Suppose you have several polygons on a map, each of which connects
**the cities within a specific distribution area.If a city is transferred
from one
**distribution area to another, you would want to get a handle to the point
**representing that city, delete it from one polygon, and add it to another
polygon.
*/
```

```
PROCEDURE move_city(area1 IN OG_Object, area2 IN OG_Object,
```

```
city_index IN NUMBER) IS
    the_city  OG_Point;
BEGIN
    the_city:=OG_Get_Point(Area1, city_index);
    OG_Delete_Point(Area1, city_index, 1);
    OG_Insert_Point(Area2, OG_Last, the_city);
END;
```

OG_Insert_Smptext

説明

このプロシージャで、指定したテキスト・オブジェクト内の指定した複数テキスト要素に新しい単一テキスト要素を挿入します。「テキスト属性」に説明されているように、複数テキスト要素の各テキスト文字列を単一テキスト要素といいます。

構文

```
PROCEDURE OG_Insert_Smptext
(
    textobj      OG_Object,
    smp_attr     OG_Smptext_Attr,
    cmpindex     NUMBER,
    smpindex     NUMBER,
    damage       BOOLEAN          := TRUE,
    update_bbox  BOOLEAN          := TRUE);
```

パラメータ

<i>text_hdl</i>	テキスト・オブジェクトのハンドル。
<i>smp_attr</i>	挿入する単一テキスト要素の属性レコード。
<i>Cmpindex</i>	単一テキストを挿入するテキスト・オブジェクト内の複数テキスト要素の索引。
<i>Smpindex</i>	新しい単一テキスト要素を挿入する複数テキスト要素内の位置の索引。この引数は、0から <i>n</i> までの整数であることが必要です。ここで、 <i>n</i> は挿入前の複数テキスト要素内の単一テキスト要素の数です。次のビルトイン定数のいずれかを指定することもできます。 OG_First 複数テキスト要素の最初（索引 = 0）に新しい単一テキスト要素を挿入します。 OG_Last 複数テキスト要素の最後（索引 = 挿入前の複数テキスト要素内の単一テキスト要素の数）に新しい単一テキスト要素を挿入します。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

指定した単一テキスト要素の属性レコード (*smp_attr*) には、テキスト文字列をはじめ、単一テキスト要素の属性が入っています。設定される属性値は、その属性レコードのマスク属性の値で指定された属性値のみです。

OG_Insert_Smptextの例

```
/* Suppose you want to create a text object that contains a message
**for the user.The following procedure will create the object, insert a
**compound text element, then insert a simple text element that contains
**the text of the message.
*/
```

```
PROCEDURE make_text (the_message IN VARCHAR2) IS
    text_rec    OG_Text_Ca;
    text_obj    OG_Object;
    smp_rec     OG_Smptext_Attr;
BEGIN
    text_rec.text_caob.mask:=OG_None_Generica;
    text_rec.text_caoh.mask:=OG_None_Graphica;
    text_rec.text_caot.mask:=OG_None_Texta;
    text_obj:=OG_Make(Text_Rec);
    OG_Insert_Cmptext(Text_Obj, OG_Last);
    smp_rec.mask:=OG_Str_Smptxta;
    smp_rec.str:=the_message;
    OG_Insert_Smptext(Text_Obj, smp_rec, 0, OG_Last);
END;
```

OG_Make_Ellipse

説明

このファンクションは楕円を作成するために使用します。

構文

```
FUNCTION OG_Make_Ellipse
    (position OG_Point,
     height  NUMBER,
     width   NUMBER)
RETURN OG_Object;
```

パラメータ

<i>position</i>	楕円のx座標とy座標。
<i>height</i>	楕円の高さ。
<i>width</i>	楕円の幅。

戻り値

新たに作成された楕円のハンドル。

OG_Make_Ellipseの例

```
/* The following function creates an ellipse:
*/

FUNCTION example RETURN OG_Object IS
  object  OG_Object;
  pos     OG_Point;
  height  NUMBER;
  width   NUMBER;
BEGIN
  pos.x := OG_Inch;
  pos.y := OG_Inch;
  height := 4* OG_Inch;
  width  := 4* OG_Inch;

  object := OG_Make_Ellipse(Pos, height, width);
  RETURN(object);
END;
```

OG_Make_Group

説明

このファンクションは、グループ・オブジェクトを作成するために使用します（なお、OG_Insert_Childを使用して子を追加するまで、作成されたグループは空のままです）。

構文

```
FUNCTION OG_Make_Group
RETURN OG_Object;
```

パラメータ

なし。

戻り値

新たに作成されたグループのハンドル。

OG_Make_Groupの例

```
/* The following function creates a group with the specified name:
*/

FUNCTION example(group_name VARCHAR2) RETURN OG_Object IS
  object OG_Object;
BEGIN
  object := OG_Make_Group;
  OG_Set_Name(object, group_name);
  RETURN(object);
END;
```

OG_Make_Image

説明

このファンクションを使用して、データベースの表に格納されたデータからイメージを作成します。

構文

```
FUNCTION OG_Make_Image
  (query          OG_Query,
   which_data     NUMBER,
   colname        VARCHAR2)
RETURN OG_Object;
```

パラメータ

<i>query</i>	データベース内の表からイメージを取り出す問合せのハンドル。なお、この表はユーザーの表であることが必要です。データベースにモジュールを保存またはエクスポートするときにGraphics Builderによって使用されるプライベート表は指定できません。
<i>which_data</i>	作成するイメージを問合せの新しいデータ・セットに入れるか古いデータ・セットに入れるかを指定します。値は、次のビルトイン定数のうちのいずれかを指定してください。 OG_Newdata 問合せの新しいデータ・セットにイメージを入れます。 OG_Olddata 問合せの古いデータ・セットにイメージを入れます。
<i>Colname</i>	イメージ・データを入れる問合せ列の名前。作成されたイメージは、この属性で指定した列と問合せのカーソルが指し示す行が交わる問合せセルに入れます。

戻り値

新たに作成されたイメージのハンドル。

OG_Make_Imageの例

```
/* The following function creates an image from data in the sixth
**row of the query 'image_query' in the column IMAGE_COLUMN:
*/

FUNCTION example(image_name VARCHAR2) RETURN OG_Object IS
  query   OG_Query;
  object  OG_Object;
BEGIN
  query := OG_Get_Query('Image_Query');
  OG_Execute_Query(Query);
  OG_Start_From(Query, OG_Newdata, 5);
  object := OG_Make_Image(Query, OG_Newdata, 'IMAGE_COLUMN');

  OG_Set_Name(Object, image_name);
  RETURN(object);
END;
```

OG_Make_Line

説明

このファンクションは線を作成するために使用します。

構文

```
FUNCTION OG_Make_Line
  (startpt  OG_Point,
   endpt    OG_Point)
RETURN OG_Object;
```

パラメータ

<i>startpt</i>	線の始点（レイアウト単位で指定します）。
<i>endpt</i>	線の終点（レイアウト単位で指定します）。

戻り値

新たに作成された線のハンドル。

OG_Make_Lineの例

```
/* The following function creates a 2" horizontal line:
*/

FUNCTION example RETURN OG_Object IS
  object  OG_Object;
  startpt OG_Point;
  endpt   OG_Point;
BEGIN
  startpt.x := OG_Inch;
  startpt.y := OG_Inch;
  endpt.x   := 2 * OG_Inch;
  endpt.y   := OG_Inch;

  object := OG_Make_Line(startpt, endpt);
  RETURN(object);
END;
```

OG_Make_Poly

説明

このファンクションを使用して、多角形オブジェクトまたは折れ線オブジェクトを作成します（なお、OG_Insert_Pointを使用して点を追加するまで、作成されたオブジェクトには点がありません）。

構文

```
FUNCTION OG_Make_Poly
RETURN OG_Object;
```

パラメータ

なし。

戻り値

新たに作成された多角形オブジェクトまたは折れ線オブジェクトのハンドル。

OG_Make_Polyの例

```
/* The following function creates a polygon with the specified name:
*/

FUNCTION example(poly_name VARCHAR2) RETURN OG_Object IS
  object OG_Object;
```

```
BEGIN
  object := OG_Make_Poly;
  OG_Set_Name(Object, poly_name);
  RETURN(object);
END;
```

OG_Make_Rect

説明

このファンクションは、四角形を作成するために使用します。

構文

```
FUNCTION OG_Make_Rect
  (position OG_Point,
   height   NUMBER,
   width    NUMBER)
RETURN OG_Object;
```

パラメータ

<i>position</i>	四角形のx座標とy座標。
<i>height</i>	四角形の高さ。
<i>width</i>	四角形の幅。

戻り値

新たに作成された四角形のハンドル。

OG_Make_Rectの例

```
/* The following function creates a rectangle:
*/

FUNCTION example RETURN OG_Object IS
  object OG_Object;
  pos    OG_Point;
  height NUMBER;
  width  NUMBER;
BEGIN
  pos.x := OG_Inch;
  pos.y := OG_Inch;
  height := 4* OG_Inch;
  width  := 4* OG_Inch;
```

```
object := OG_Make_Rect(Pos, height, width);  
RETURN(object);  
END;
```

OG_Make_Rrect

説明

このファンクションは、丸い四角形を作成するために使用します。

構文

```
FUNCTION OG_Make_Rrect  
  (position OG_Point,  
   height   NUMBER,  
   width    NUMBER)  
RETURN OG_Object;
```

パラメータ

<i>position</i>	丸い四角形のx座標とy座標。
<i>height</i>	丸い四角形の高さ。
<i>width</i>	丸い四角形の幅。

戻り値

新たに作成された丸い四角形のハンドル。

OG_Make_Rrectの例

```
/* The following function creates a rounded rectangle:  
*/  
  
FUNCTION example RETURN OG_Object IS  
  object OG_Object;  
  pos    OG_Point;  
  height NUMBER;  
  width  NUMBER;  
BEGIN  
  pos.x := OG_Inch;  
  pos.y := OG_Inch;  
  height := 4* OG_Inch;  
  width  := 4* OG_Inch;  
  
  object := OG_Make_Rrect(Pos, height, width);  
  RETURN(object);  
END;
```

END;

OG_Make_Symbol

説明

このファンクションは、記号を作成するために使用します。

構文

```
FUNCTION OG_Make_Symbol
  (position OG_Point,
   indx     NUMBER,
   symsize  NUMBER)
RETURN OG_Object;
```

パラメータ

<i>position</i>	記号の中央の位置。
<i>indx</i>	記号パレットに表示するときの記号の位置の索引（または数値）。
<i>Symsize</i>	記号のサイズ。このプロパティには、次のビルトイン定数を指定できます。 OG_Large_Symsize OG_Medium_Symsize OG_Small_Symsize

戻り値

新たに作成された記号のハンドル。

OG_Make_Symbolの例

```
/* The following function creates a symbol:
*/

FUNCTION example RETURN OG_Object IS
  symbol OG_Object;
  pos    OG_Point;
BEGIN
  pos.x := OG_Inch;
  pos.y := OG_Inch;

  symbol := OG_Make_Symbol(Pos, 5, OG_Large_Symsize);
  RETURN(symbol);

END;
```

OG_Make_Text

説明

このファンクションは、テキスト・オブジェクトを作成するために使用します。

構文

```
FUNCTION OG_Make_Text
  (position OG_Point,
RETURN OG_Object;

OG_Make_Text
  (position OG_Point,
   string VARCHAR2)
RETURN OG_Object;

OG_Make_Text
  (position OG_Point,
   string VARCHAR2,
   ptsize NUMBER)
RETURN OG_Object;
```

パラメータ

<i>position</i>	テキスト・オブジェクトのx座標とy座標。
<i>string</i>	テキスト文字列。
<i>ptsize</i>	文字サイズ。

戻り値

新たに作成されたテキスト・オブジェクトへのハンドル。

OG_Make_Textの例

```
/* The following function creates a text object:
*/

FUNCTION example RETURN OG_Object IS
  object OG_Object;
  pos    OG_Point;
BEGIN
  pos.x := OG_Inch;
  pos.y := OG_Inch;
  object := OG_Make_Text(Pos, 'Sales Rankings');
  RETURN(object);
END;
```

OG_Move

説明

このプロシージャで、指定したオブジェクトをレイアウトの別の位置に移動します。

構文

```
PROCEDURE OG_Move
  (object_hdl      OG_Object,
   offset          OG_Point,
   damage          BOOLEAN    := TRUE,
   update_bbox     BOOLEAN    := TRUE);
```

パラメータ

<i>object_hdl</i>	移動するオブジェクトのハンドル。
オフセット	オブジェクトを現在の位置から移動する相対的な距離。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

移動後のオブジェクトのxy座標が負になるようにオフセットを指定すれば、オブジェクトをレイアウトの外に移動できます。

また、オフセットのxy成分に正の値を指定すると、オブジェクトが右下に移動します。負の値を指定すると、オブジェクトが左上に移動します。

オブジェクトをレイアウト上の任意の絶対座標に移動するには、適切な属性レコードに新しい位置を設定します。

OG_Moveの例

```
/* Suppose you have an object that represents inventory in a warehouse.
**If the inventory is moved from one warehouse to another, you would
**want to move the object that represents it.
*/

PROCEDURE move_inventory(invent_obj IN OG_Object) IS
  distance  OG_Point;
BEGIN
  distance.x:= (3*OG_Inch);
  distance.y:= (3*OG_Inch);
  OG_Move(Invent_Obj, distance);
END;
```


OG_Point_In

説明

このファンクションを使用して、指定した参照点がオブジェクトの領域内にあるかどうかを判別します。

構文

```
FUNCTION OG_Point_In
  (object_hdl  OG_Object,
   ref_pt      OG_Point,
   aperture    OG_Point)
RETURN OG_Object;

FUNCTION OG_Point_In
  (window_hdl  OG_Window,
   ref_pt      OG_Point,
   aperture    OG_Point)
RETURN OG_Object;
```

パラメータ

<i>object_hdl</i>	チェックするオブジェクトのハンドル。
<i>window_hdl</i>	チェックするウィンドウのハンドル。
<i>ref_pt</i>	参照点。
<i>Aperture</i>	参照点からの最大許容距離(<i>object_hdl</i> に塗りなしがある場合にのみ指定します)。

戻り値

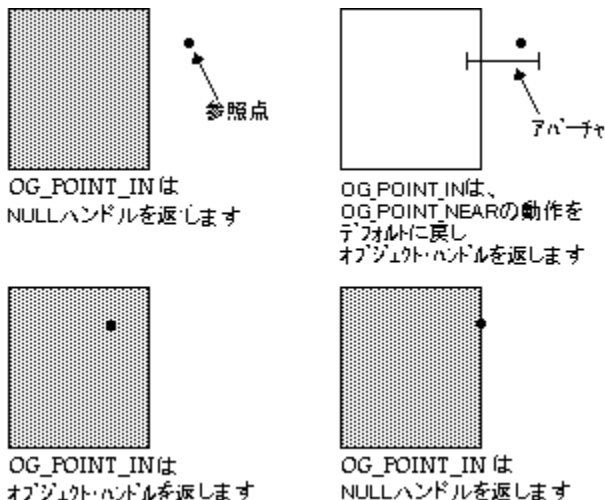
指定した参照点がオブジェクトの領域内にあれば、このファンクションによりそのオブジェクトのハンドルが戻ります。参照点がオブジェクトの領域内になければ、NULLハンドルが戻ります。

使用上の注意

このファンクションは、ユーザーのマウスのアクティビティを監視する場合に特に便利です。たとえば、グループ・オブジェクト用のボタン・プロシージャを作成し、そのプロシージャの最初の部分でイベント・レコードを使用して、マウスが選択または移動されたときのマウスの位置を特定できます。その後、OG_Point_InやOG_Point_Nearをコールすれば、グループ・オブジェクトとマウスの座標を引数として渡せます。このとき、このファンクションにより、ユーザーが選択したグループ内のオブジェクトが戻ります。

なお、このファンクションで判別できるのは、オブジェクトの領域内にある点のみです。参照点がオブジェクトの枠上にあれば、NULLハンドルが戻ります。（このファンクションでは、オブジェクトの枠の太さが可能な範囲で最も細いことが前提です。つまり太い枠線に変換された領域

は無視されます)。オブジェクトが塗りなしパターンの場合(つまり、塗り領域がまったくない場合)、このファンクションによりデフォルトでOG_Point_Nearの動作が実行されます。なお、引数`aperture`はOG_Point_Inでは使用されませんが、オブジェクトが塗りなしパターンの場合にOG_Point_Nearに渡されます。



引数にオブジェクトを1つのみ指定すると、このファンクションによりそのオブジェクト内に参照点が存在するかどうかチェックされます。一方、グループ・オブジェクトを指定すると、そのグループ(とサブグループ)のメンバーがすべてチェックされ、参照点が存在する一番上の単一オブジェクトが戻ります(グループのどのオブジェクトにも参照点が存在しなければ、NULLハンドルが返されます)。なお、レイヤーはグループ・オブジェクトとして渡すことができます。同様に、オブジェクトのかわりにウィンドウを指定すると、このファンクションにより、そのウィンドウ内のオブジェクトがすべてチェックされます。

参照点が複数のオブジェクトの枠上に存在する場合は、グループ・ツリーの一番上のオブジェクトが戻ります。

OG_Point_Inの例

```
/* Suppose your application allows the user to select an object and drag
it on
**top of other objects that are within a group.When the user releases the
mouse
**button, you want to determine which object the mouse is pointing to,
and destroy it.
**The following procedure could be used as a button procedure for the object
that was dragged.
*/
```

```
PROCEDURE destroy_target (hitobj IN OG_Object, buttonobj IN
```

```

OG_Object, win IN OG_Window, eventinfo IN OG_Event) IS
    the_group    OG_Object;
    aper         OG_Point;
    target_obj   OG_Object;
BEGIN
    IF eventinfo.event_type=OG_Mouse_Up THEN
        the_group:=OG_Get_Object('Big_Group');
        aper.x:=3*OG_App.HSCREEN_RES;    /* three pixels */
        aper.y:=3*OG_App.VSCREEN_RES;    /* three pixels */
        target_obj:=OG_Point_In(The_Group,
            eventinfo.mouse_position, aper);
        IF not (OG_Isnull(Target_Obj)) THEN
            OG_Destroy(Target_Obj);
        END IF;
    END IF;
END;

```

OG_Point_Near

説明

このファンクションを使用して、オブジェクトの枠上に、指定した参照点があるかどうかを判別します。

構文

```

FUNCTION OG_Point_Near
    (object_hdl  OG_Object,
     ref_pt      OG_Point,
     aperture    OG_Point)
RETURN OG_Object;

FUNCTION OG_Point_Near
    (window_hdl  OG_Window,
     ref_pt      OG_Point,
     aperture    OG_Point)
RETURN OG_Object;

```

パラメータ

<i>object_hdl</i>	チェックするオブジェクトのハンドル。
<i>window_hdl</i>	チェックするウィンドウのハンドル。
<i>ref_pt</i>	参照点。
<i>Aperture</i>	参照点からの最大許容距離。

戻り値

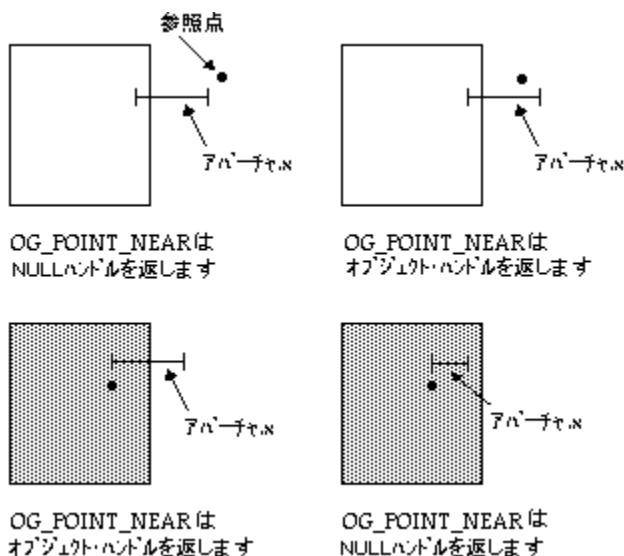
指定した参照点がオブジェクトの枠上にあれば、このファンクションによりそのオブジェクトのハンドルが戻ります。参照点がオブジェクトの枠上になければ、NULLハンドルが戻ります。

使用上の注意

このファンクションは、ユーザーのマウスのアクティビティを監視する場合に特に便利です。たとえば、グループ・オブジェクト用のボタン・プロシーダを作成し、そのプロシーダの最初の部分でイベント・レコードを使用して、マウスが選択または移動されたときのマウスの位置を特定できます。その後、OG_Point_InやOG_Point_Nearをコールすれば、グループ・オブジェクトとマウスの座標を引数として渡せます。このとき、このファンクションにより、ユーザーが選択した単一オブジェクトが戻ります。

なお、オブジェクトが透明な枠パターンの場合、このファンクションではNULLハンドルが戻ります。

引数に単一オブジェクトを指定すると、このファンクションにより、そのオブジェクトの枠上に参照点が存在するかどうかチェックされます。一方、グループ・オブジェクトを指定すると、そのグループ(とサブグループ)のメンバーがすべてチェックされ、枠上に参照点が存在する単一オブジェクトが戻ります(グループのどのオブジェクトの枠上にも参照点が存在しなければ、NULLハンドルが戻ります)。なお、レイヤーはグループ・オブジェクトとして渡すことができます。同様に、オブジェクトのかわりにウィンドウを指定すると、このファンクションにより、そのウィンドウ内のオブジェクトがすべてチェックされます。



引数`aperture`は、参照点とオブジェクトの枠の最大許容距離を表します。すなわち、枠からの距離がこの引数の値以下であれば、そのオブジェクトのハンドルが戻ります。また、`aperture`で指定した領域がオブジェクトの枠にかかっている場合も、`OG_Point_Near`はそのオブジェクトのハンドルを戻します。なお、`aperture`にはxとy両方の成分があります。

参照点が複数のオブジェクトの枠上に存在する場合は、グループ・ツリーの一番上のオブジェクトが戻ります。

通常、`aperture`は、ユーザーがマウスで位置を指定するときの、許容できるずれの範囲を決める目的で使われます。このような場合、`aperture`のx成分とy成分には同じ値、一般に3ピクセルに設定します。

OG_Point_Nearの例

```
/* Suppose your application allows the user to select an object and drag
it
**to the edge of other objects that are within a group. When the user releases
**the mouse button, you want to determine which object's edge the mouse
**is pointing to, and destroy it. The following procedure could be used
as a
**button procedure for the object that was dragged.
*/
```

```
PROCEDURE destroy_target (hitobj IN OG_Object, buttonobj IN
  OG_Object, win IN OG_Window, eventinfo IN OG_Event) IS
  the_group    OG_Object;
  aper         OG_Point;
  target_obj   OG_Object;
BEGIN
  IF eventinfo.event_type=OG_Mouse_Up THEN
    the_group:=OG_Get_Object('Big_Group');
    aper.x:=3*OG_App.HSCREEN_RES; /* three pixels */
    aper.y:=3*OG_App.VSCREEN_RES; /* three pixels */
    target_obj:=OG_Point_Near(The_Group,
      eventinfo.mouse_position, aper);
    IF not (OG_Isnull(Target_Obj)) THEN
      OG_Destroy(Target_Obj);
    END IF;
  END IF;
END;
```

OG_Property_Exists

説明

このファンクションを使用して、ユーザー定義プロパティがオブジェクトに設定されているかどうかをチェックします。

構文

```
FUNCTION OG_Property_Exists
  (object_hdl  OG_Object,
   prop_name   VARCHAR2)
RETURN BOOLEAN;
```

パラメータ

<i>object_hdl</i>	チェックするオブジェクトのハンドル。
<i>prop_name</i>	存在するかどうかをチェックしたいプロパティの名前。

戻り値

TRUE	指定したプロパティが存在する場合。
FALSE	指定したプロパティが存在しない場合。

OG_Property_Existsの例

```
/* The following procedure adds the property 'priority' to an object, if
it doesn't already exist:
*/

PROCEDURE example(object OG_Object) IS
BEGIN
  IF NOT OG_Property_Exists(Object, 'priority') THEN
    OG_Set_Property(Object, 'priority', 10);
  END IF;
END;
```

OG_Rotate

説明

このプロシージャで、指定したオブジェクトを指定した角度だけ反時計回りに回転させます。

構文

```
PROCEDURE OG_Rotate
  (object_hdl  OG_Object,
   center_pt   OG_Point,
   degrees     NUMBER,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);
```

パラメータ

<i>object_hdl</i>	回転させるオブジェクトのハンドル。
-------------------	-------------------

<i>center_pt</i>	回転軸となるレイアウト上の点。
<i>Degrees</i>	オブジェクトを反時計回りに回転させる角度。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

OG_Rotateの例

```
/* Suppose you have a display that contains a dial. The needle of the dial
**points at specific values along the face of the dial. When the data changes,
**you may want to rotate the needle to point to a new value.
*/
```

```
PROCEDURE rotate_needle (degrees IN NUMBER,
    center_pt IN OG_Point) IS
    the_needle OG_Object;
BEGIN
    the_needle:=OG_Get_Object('Needle 1');
    OG_Rotate(The_Needle, center_pt, degrees);
END;
```

OG_Same

説明

このファンクションを使用して、2つのハンドルを比較して同じものかどうかチェックします。たとえば、このファンクションに2つのオブジェクトのハンドルを渡すと、2つのハンドルが同じオブジェクトを指しているかどうかチェックされます。

構文

FUNCTION OG_Same (handle1 OG_Object, handle2 OG_Object) RETURN BOOLEAN;	object
FUNCTION OG_Same (handle1 OG_Query, handle2 OG_Query) RETURN BOOLEAN;	query
FUNCTION OG_Same (handle1 OG_Template, handle2 OG_Template) RETURN BOOLEAN;	chart template
FUNCTION OG_Same	button procedure

```

    (handle1 OG_Buttonproc,
     handle2 OG_Buttonproc)
RETURN BOOLEAN;

FUNCTION OG_Same                                sound
    (handle1 OG_Sound,
     handle2 OG_Sound)
RETURN BOOLEAN;

FUNCTION OG_Same                                window
    (handle1 OG_Window,
     handle2 OG_Window)
RETURN BOOLEAN;

FUNCTION OG_Same                                layer
    (handle1 OG_Layer,
     handle2 OG_Layer)
RETURN BOOLEAN;

FUNCTION OG_Same                                timer
    (handle1 OG_Timer,
     handle2 OG_Timer)
RETURN BOOLEAN;

FUNCTION OG_Same                                display
    (handle1 OG_Display,
     handle2 OG_Display)
RETURN BOOLEAN;

FUNCTION OG_Same                                axis
    (handle1 OG_Axis,
     handle2 OG_Axis)
RETURN BOOLEAN;

FUNCTION OG_Same                                field template
    (handle1 OG_Ftemp,
     handle2 OG_Ftemp)
RETURN BOOLEAN;

FUNCTION OG_Same                                reference line
    (handle1 OG_Refline,
     handle2 OG_Refline)
RETURN BOOLEAN;
```

パラメータ

<i>handle1</i>	比較する2つのハンドルのうち最初のハンドル。
<i>handle2</i>	比較する2つのハンドルのうち2番目のハンドル。

戻り値

TRUE	2つのハンドルが同じ場合。
FALSE	2つのハンドルが異なる場合。

使用上の注意

"="は使用できないため、ハンドルの値を比較するにはこのファンクションが必要です。たとえば、次のプロシージャは正しくありません。

```
PROCEDURE invalid (obj1 OG_Object, obj2 OG_Object) IS
BEGIN
    IF obj1 = obj2 THEN      --illegal comparison
        NULL;
    END IF;
END;
```

OG_Sameの例

```
/* Suppose you want to compare two objects to see if they are the same.
**The following function returns TRUE if they are the same and FALSE if
they are not:
*/
```

```
FUNCTION compare (obj1 OG_Object, obj2 OG_Object) RETURN BOOLEAN IS
BEGIN
    IF OG_Same(obj1, obj2) THEN
        RETURN(TRUE);
    ELSE
        RETURN(FALSE);
    END IF;
END;
```

OG_Scale

説明

このプロシージャで、指定したオブジェクトのサイズを変更します。

構文

```
PROCEDURE OG_Scale
(object_hdl OG_Object,
 anchor      OG_Point,
 oldpt       OG_Point,
 newpt       OG_Point,
 damage      BOOLEAN    := TRUE,
 update_bbox BOOLEAN    := TRUE);
```

パラメータ

<i>object_hdl</i>	サイズを変更するオブジェクトのハンドル。
-------------------	----------------------

<i>anchor</i>	オブジェクトのアンカー・ポイント。
<i>oldpt</i>	始点。
<i>newpt</i>	終点。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

新しい点とアンカー・ポイントとの距離、および古い点とアンカー・ポイントとの距離から、スケール・ファクタ値が計算されます。つまり、この2つの距離の比がスケール変更係数になります。スケール変更係数はx座標とy座標で別々に計算されます。

次に、オブジェクトの各制御ポイントのアンカー・ポイントに対する相対位置がこの係数を使用してスケールリングされます。なお、x座標とy座標のスケール変更係数が等しい場合は、同じ比率を保ったまま（つまり、正方形は正方形のまま）オブジェクトのサイズが変更されます。

たとえば、左上の制御ポイントが(OG_Inch, OG_Inch)のオブジェクトのサイズを2倍にする場合、各パラメータの値を次のように指定します。

anchor =(OG_Inch, OG_Inch), *oldpt*=(OG_Inch+1, OG_Inch+1), *newpt*=(OG_Inch+2, OG_Inch+2)。

したがって、x座標のスケール変更係数は次のようになります。

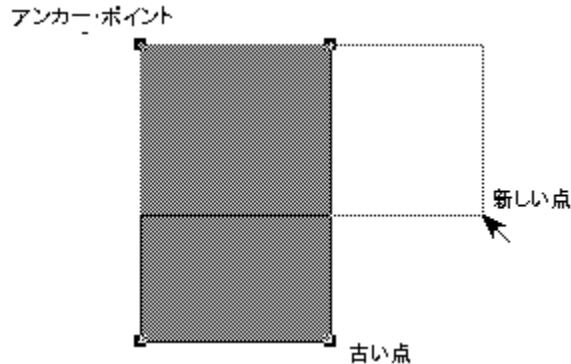
$$(newpt.x-anchor.x) / (oldpt.x-anchor.x) = (OG_Inch+2-OG_Inch) / (OG_Inch+1-OG_Inch) = 2 / 1 = 2$$

$$(newpt.y-anchor.y) / (oldpt.y-anchor.y) = (OG_Inch+2-OG_Inch) / (OG_Inch+1-OG_Inch) = 2 / 1 = 2$$

アンカー・ポイントとオブジェクトの各制御ポイント間の距離は、この係数を使用してスケールリングされます。上の例では、スケール変更係数が2で、オブジェクトの右上の制御ポイントがアンカー・ポイントの1.5インチ右にある場合、この制御ポイントはアンカー・ポイントの3インチ右の位置まで移動します。オブジェクトの他の制御ポイントも、同じように移動します。

なお、制御ポイントをアンカー・ポイントとして使用すると、制御ポイントとアンカー・ポイントの距離が0になるため、制御ポイントの位置は変わりません。

また、このプロシージャを使用すると、「選択」ツールを使用して制御ポイントを選択し、レイアウト上の新しい位置までドラッグしたときと同じように、指定したオブジェクトのサイズを変更できます。アンカー・ポイントは、操作中に移動しない制御ポイントです。*oldpt*は移動前の制御ポイント、*newpt*は移動先の新しい位置です。



OG_Scaleの例

```
/* Suppose you want to double the size of the object that the user selects.
**Assume the object's center is at (OG_Inch, OG_Inch), and use this point
as
**the anchor. The following button procedure will double the size of the
object:
*/
```

```
PROCEDURE double (buttonobj IN OG_Object, hitobj IN
  OG_Object, win IN OG_Window, eventinfo IN OG_Event) IS
  anchor OG_Point;
  newpt OG_Point;
  oldpt OG_Point;
BEGIN
  anchor.x:=OG_Inch;
  anchor.y:=OG_Inch;
  oldpt.x:=OG_Inch+1;
  oldpt.y:=OG_Inch+1;
  newpt.x:=OG_Inch+2;
  newpt.y:=OG_Inch+2;
  OG_Scale(hitobj, anchor, oldpt, newpt);
END;
```

OG_Set_Edgecolor

説明

このプロシージャで、指定したオブジェクトの枠のカラーを設定します。つまり、枠のパターンを「透明」に、枠のバックグラウンド・カラーを指定したカラーに設定します。

構文

```
PROCEDURE OG_Set_Edgecolor
```

```
(object      OG_Object,
color       VARCHAR2
damage      BOOLEAN    :=  TRUE,
update_bbox BOOLEAN    :=  TRUE);
```

パラメータ

<i>object</i>	変更するオブジェクトのハンドル。
<i>color</i>	カラーの名前。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

OG_Set_Edgecolorの例

```
/* The following procedure sets the edge color of the specified object:
*/

PROCEDURE example(object OG_Object) IS
BEGIN
  OG_Set_Edgecolor(Object, 'red');
END;
```

OG_Set_Fillcolor

説明

このプロシージャで、指定したオブジェクトの塗りカラーを設定します。つまり、塗りパターンを「透明」に、塗りバックグラウンド・カラーを指定したカラーに設定します。

構文

```
PROCEDURE OG_Set_Fillcolor
(object      OG_Object,
color       VARCHAR2
damage      BOOLEAN    :=  TRUE,
update_bbox BOOLEAN    :=  TRUE);
```

パラメータ

<i>object</i>	変更するオブジェクトのハンドル。
<i>color</i>	カラーの名前。
<i>Damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

OG_Set_Fillcolorの例

```
/* The following procedure sets the fill color of the specified object:
*/

PROCEDURE example(object OG_Object) IS
BEGIN
  OG_Set_Fillcolor(Object, 'red');
END;
```

OG_Set_Property

説明

このプロシージャで、オブジェクトのユーザー定義プロパティの値を設定します。

構文

```
PROCEDURE OG_Set_Property                                date
(object_hdl OG_Object,
 prop_name  VARCHAR2,
 prop_value VARCHAR2,
 date_fmt   VARCHAR2  := 'DD-MON-YY');

PROCEDURE OG_Set_Property                                number
(object_hdl OG_Object,
 prop_name  VARCHAR2,
 prop_value NUMBER);

PROCEDURE OG_Set_Property                                char
(object_hdl OG_Object, :
 prop_name  VARCHAR2,
 prop_value VARCHAR2,);
```

パラメータ

<i>object_hdl</i>	プロパティを設定するオブジェクトのハンドル。
<i>prop_name</i>	設定するプロパティ名。
<i>prop_value</i>	プロパティに設定する値。
<i>date_fmt</i>	<i>prop_value</i> 文字列を日付に変換するための日付書式マスク。

使用上の注意

指定したプロパティが存在すれば、このプロシージャによりその値が変更されます。プロパティが存在しなければ、エラーを出力せずに、プロパティが作成され値が設定されます。

OG_Set_Propertyの例

```
/* The following procedure gets the 'priority' property in each child object
in a
**group, and then sets the priority to one greater than its current value:
*/

PROCEDURE example(group_name VARCHAR2) IS
    group_obj    OG_Object;
    child_count  NUMBER;
    child_obj    OG_Object;
    current_p    NUMBER;
BEGIN
    group_obj := OG_Get_Object(Group_Name);
    child_count := OG_Get_Childcount(Group_Obj);

    FOR i IN 0..child_count-1 LOOP
        child_obj := OG_Get_Child(Group_Obj, i);
        current_p := OG_Get_Num_Property(Child_Obj, 'priority');
        OG_Set_Property(Child_Obj, 'priority', current_p + 1);
    END LOOP;

END;
```

OG_Synchronize

説明

このプロシージャで、すべてのウィンドウの領域を再描画します。つまり、画面に表示されている画像を全オブジェクトの内部表現と同期化します。

構文

```
PROCEDURE OG_Synchronize;
```

パラメータ

なし。

使用上の注意

ユーザーが作成したPL/SQLプロシージャでは、その処理が終わった後にOG_Synchronizeが暗黙的に実行されます。

OG_Synchronizeの例

```
/* Suppose you want to move an object across the display in ten 1/4" increments.
** Instead of moving it multiple times and having it update visually only
```

```
at the end
**of the procedure, you may want to "synchronize" the layout with the internal
**representation of the object after each move.
*/

PROCEDURE slide_across(the_object IN OG_Object) IS
    offset    OG_Point;
BEGIN
    offset.x:=(1/4)*OG_Inch;
    offset.y:=0;
    FOR i IN 1..10 LOOP
        move(the_object, offset);
        OG_Synchronize;
    END LOOP;
END;
```

OG_Update_Bbox

説明

このファンクションで、オブジェクトのバウンディング・ボックスを更新します。オブジェクトがグループの場合は、そのオブジェクトの子孫のバウンディング・ボックスもすべて更新します。

構文

```
PROCEDURE OG_Update_Bbox
(object_hdl  OG_Object,
which_bbox  NUMBER);
```

パラメータ

<i>object_hdl</i>	更新するオブジェクトのハンドル
<i>which_bbox</i>	内側または外側のバウンディング・ボックスを更新するかどうかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Bothbbox 内側と外側のバウンディング・ボックスを更新します。 OG_Innerbbox 内側のバウンディング・ボックスのみを更新します。 OG_Outerbbox 外側のバウンディング・ボックスのみを更新します。

OG_Update_Bboxの例

```
/*Suppose you want to move an object.The default behavior
**of the built-in procedure OG_Move is to update the bounding
**boxes of all of the modified object's antecedants, including
**the layer on which the object resides.To update a layer's
**bounding boxes, Graphics Builder must examine every object on that layer.
**If the layer contains a large number of objects, this operation
**can be very time-consuming.*/
```

```
/*To make your application more efficient, you can move the
**object while inhibiting this automatic bounding box update,
**then explicitly update only that object's bounding boxes.
**(Note that since the automatic bounding box update does not
**occur, the bounding boxes of the object's antecedents
**may be inaccurate.)
*/

/*When you modify an object with a FALSE bounding box update
**flag, you may also want to use a FALSE damage flag. In this case,
**when you are through modifying the object, you would invoke
**OG_Damage to explicitly damage the object.
*/

PROCEDURE move_efficiently (the_object OG_Object) IS
    offset    OG_Point;
BEGIN
    offset.x:=OG_Inch;
    offset.y:=OG_Inch;
    OG_Move(The_Object, offset, FALSE, FALSE)
    OG_Update_Bbox(The_Object, OG_Bothbbox);
    OG_Damage(The_Object);
END;
```


レイヤー・ビルトイン

OG_Activate_Layer

OG_Get_Layer

OG_Hide_Layer

OG_Show_Layer

OG_Activate_Layer

説明

このプロシージャで、指定したウィンドウ内の指定したレイヤーをアクティブにします。

構文

```
PROCEDURE OG_Activate_Layer
(layer_hdl  OG_Layer,
window_hdl OG_Window
damage     BOOLEAN := TRUE);

PROCEDURE OG_Activate_Layer
(layer_hdl  OG_Layer,
damage     BOOLEAN := TRUE);
```

パラメータ

<i>layer_hdl</i>	アクティブにするレイヤーのハンドル。
<i>window_hdl</i>	アクティブにするレイヤーが含まれているウィンドウのハンドル。指定しなければ、すべてのウィンドウでそのレイヤーがアクティブにされます。
<i>damage</i>	ダメージ・フラグ。

使用上の注意

隠れたレイヤーをアクティブにすると、そのレイヤーが画面に表示されます。アクティブにできるレイヤーは一度に1つのみです。

また、グループ・オブジェクトを画面のルート・オブジェクトの子として挿入した場合は、OG_Get_Layerを使用してそのグループのレイヤー・ハンドルを取得できます。このようなグループ・オブジェクトをアクティブにすると、Graphics Builderではこのオブジェクトが1つのレイヤーとして認識されます。

OG_Activate_Layerの使用例

```
/* Suppose your layout contains several layers,
**each of which contains a set of buttons.
**If you want certain buttons to be active at
**a specific time, you can activate the layer
**that contains those buttons.If the user
**selects on a button object that is not in the active
**layer, nothing will happen.
*/

PROCEDURE activate_a_layer(layer_num NUMBER, the_window OG_Window) IS
    layer_name  VARCHAR2(6);
    the_layer   OG_Layer;
BEGIN
    layer_name:='layer'||TO_CHAR(layer_num);
    the_layer:=OG_Get_Layer(Layer_Name);
    OG_Activate_Layer(The_Layer, the_window);
END;
```

OG_Get_Layer

説明

OG_Get_Objectファンクションにパラメータとしてグループ・オブジェクト名を渡すと、レイヤーを1つのグループ・オブジェクトとして扱えます。

構文

```
FUNCTION OG_Get_Layer
    (layer_name VARCHAR2)
RETURN OG_Layer;
```

パラメータ

<i>layer_name</i>	レイヤーの名前。このレイヤーのハンドルが戻ります。
-------------------	---------------------------

戻り値

指定されたレイヤーのハンドル。指定されたレイヤーが存在しない場合、このファンクションを使用するとNULLハンドルが戻ります。

使用上の注意

OG_Get_Layerを使用して、グループ・オブジェクトのレイヤー・ハンドルを取得し、さらにそのグループを表示またはアクティブにすると、そのグループをレイヤーとして扱えます。

OG_Get_Layerの使用例

```
/* Suppose you want to hide "layer1".
*/

PROCEDURE make_layer1_invis (the_window  OG_Window) IS
    my_layer  OG_Layer;
BEGIN
    my_layer:=OG_Get_Layer('Layer1');
    OG_Hide_Layer(My_Layer, the_window);
END;
```

OG_Hide_Layer

説明

このプロシージャで指定したレイヤーを隠します。

構文

```
PROCEDURE OG_Hide_Layer
    (layer_hdl  OG_Layer);

PROCEDURE OG_Hide_Layer
    (layer_hdl  OG_Layer,
     window_hdl OG_Window);
```

パラメータ

<i>layer_hdl</i>	隠すレイヤーのハンドル。
<i>window_hdl</i>	隠すレイヤーが含まれているウィンドウのハンドル。指定しなければ、すべてのウィンドウでそのレイヤーが隠されます。

使用上の注意

指定したレイヤーが複数のウィンドウに表示されている場合は、指定したウィンドウ内のレイヤーのみが隠されます。また、アクティブなレイヤーは隠せないなので、まず、別のレイヤーをアクティブにしてから実行してください。

OG_Hide_Layerの使用例

```
/* Suppose "layer1" contains information that is no longer useful to view.
**The following procedure will hide it:
*/

PROCEDURE make_layer1_invis(the_window  OG_Window) IS
```

```
my_layer OG_Layer;  
BEGIN  
  my_layer:=OG_Get_Layer('Layer1');  
  OG_Hide_Layer(My_Layer, the_window);  
END;
```

OG_Show_Layer

説明

このプロシージャで、指定したレイヤーを表示します。

構文

```
PROCEDURE OG_Show_Layer  
  (layer_hdl OG_Layer,  
   window_hdl OG_Window);
```

パラメータ

<i>layer_hdl</i>	表示するレイヤーのハンドル。
<i>window_hdl</i>	表示するレイヤーが含まれているウィンドウのハンドル。指定しなければ、すべてのウィンドウでそのレイヤーが表示されます。

使用上の注意

指定したレイヤーが複数のウィンドウで隠されていた場合は、指定したウィンドウ内のレイヤーのみが表示されます。

また、グループ・オブジェクトを画面のルート・オブジェクトの子として挿入した場合は、OG_Get_Layerを使用してそのグループのレイヤー・ハンドルを取得できます。このようなグループ・オブジェクトを表示すると、Graphics Builderではこのオブジェクトが1つのレイヤーとして認識されます。

OG_Show_Layerの使用例

```
/* Suppose you want to show "layer1".  
*/  
  
PROCEDURE make_layer_visible (the_window OG_Window) IS  
  my_layer OG_Layer;  
BEGIN  
  my_layer:=OG_Get_Layer('Layer1');  
  OG_Show_Layer(My_Layer, the_window);  
END;
```

その他のビルトイン

DOSQL

OG_Append_Directory

OG_Append_File

OG_Center

OG_Damage (領域)

OG_Get_Attr (アプリケーション)

OG_Get_Attr (軸)

OG_Get_Attr (図表)

OG_Get_Attr (フィールド・テンプレート)

OG_Get_Attr (枠テンプレート)

OG_Get_Attr (オブジェクト)

OG_Get_Attr (プリンタ)

OG_Get_Attr (問合せ)

OG_Get_Attr (参照線)

OG_Get_Attr (サウンド)

OG_Get_Attr (タイマー)

OG_Get_Attr (ウィンドウ)

OG_Get_Buttonproc

OG_Help

OG_Host

OG_Pause

OG_Print

OG_Quit

- OG_Root_Object
- OG_Set_Attr (アプリケーション)
- OG_Set_Attr (軸)
- OG_Set_Attr (チャート要素)
- OG_Set_Attr (図表)
- OG_Set_Attr (フィールド・テンプレート)
- OG_Set_Attr (枠テンプレート)
- OG_Set_Attr (オブジェクト)
- OG_Set_Attr (プリンタ)
- OG_Set_Attr (問合せ)
- OG_Set_Attr (参照線)
- OG_Set_Attr (サウンド)
- OG_Set_Attr (タイマー)
- OG_Set_Attr (ウィンドウ)
- OG_Translate_Envvar
- OG_User_Exit

DO_SQL

説明

このプロシージャは、指定されたSQL文を実行します。

構文

```
PROCEDURE do_sql
  (sql_stmt VARCHAR2);
```

パラメータ

sql_stmt	有効なSQL文。これにはDML（データ操作言語）文またはDDL（データ定義言語）文が含まれます。
----------	--

使用上の注意

標準PL/SQLではPL/SQLプログラム単位にDML文を含めることができますが、DDL文を実行できないので、かわりにこのプロシージャを使用してDDL文を実行します。ただし、PL/SQLプログラム単位にDML文を組み込むことができます。一般に、DML文はDO_SQLプロシージャを使用するよりプログラム単位内で効率的に実行されます。

Do_Sql Examples

```
/* The following procedure creates a table from within Graphics Builder:
*/

PROCEDURE create_table(table_name VARCHAR2) IS
BEGIN
  do_sql('create table' || table_name ||
        ' (empno number(4),' ||
        '     ename varchar2(10));');
END;
```

OG_Append_Directory

説明

このファンクションは、ファイル・システムでパス名を指定する文字列を作成します。

構文

```
FUNCTION OG_Append_Directory
  (dir      VARCHAR2,
   subdir   VARCHAR2)
RETURN VARCHAR2;
```

パラメータ

<i>dir</i>	<i>subdir</i> が追加されるディレクトリを指定する文字列。この引数には有効ディレクトリの完全名が含まれている必要があります。
<i>subdir</i>	<i>dir</i> に追加されるサブディレクトリを指定する文字列。

戻り値

完全なディレクトリ・パスを含む文字列。

使用上の注意

ディレクトリおよびサブディレクトリの名前を指定した場合、このファンクションは、使用しているシステムに合ったディレクトリ・セパレータを使用してこれらを連結します。

OG_Append_Directoryの例

```
/* Suppose you create a display that is run on several different systems,
**and one function of that display is to import an image from the file
**'my_image'. Assume the identical directory structure exists on all systems
**on which the display is run; however, each system requires a different
**directory separator. The following procedure creates a valid directory
**string for each system:
*/

PROCEDURE import_my_image(file_path VARCHAR2) IS
    the_image    OG_Object;
BEGIN
    file_path:=OG_Append_Directory(File_Path, 'home');
    file_path:=OG_Append_Directory(File_Path, 'smith');
    file_path:=OG_Append_Directory(File_Path, 'images');
    file_path:=OG_Append_File(File_Path, 'my_image');
    the_image:=OG_Import_Image(File_Path, OG_Filesystem,
    OG_Tiff_Iformat);
END;

/*Assume the initial value of file_path is 'C:¥'. On MS-DOS systems,
**the value of **file_path that is passed to OG_Import_Image is:
**C:¥home¥smith¥images¥my_image.
*/

/*Assume the initial value of file_path is 'disk$ic1[]'. On VMS systems,
**the value of file_path that is passed to OG_Import_Image is:
**disk$ic1:[home.smith.images]my_image.
*/
```

OG_Append_File

説明

このファンクションは、使用しているファイル・システムのファイルのパス名を指定する文字列を作成します。

構文

```
FUNCTION OG_Append_File
    (dir          VARCHAR2,
     filename     VARCHAR2)
```



```
RETURN VARCHAR2;
```

パラメータ

<i>dir</i>	<i>filename</i> が追加されるディレクトリを指定する文字列。この引数には有効ディレクトリの完全名が含まれている必要があります。
<i>filename</i>	<i>dir</i> に追加されるファイル名を指定する文字列。

戻り値

完全なファイル・パスを含む文字列。

使用上の注意

ディレクトリおよびファイルの名前を指定した場合、このファンクションは、使用しているシステムに合ったディレクトリ・セパレータを使用してこれらを連結します。

OG_Append_Fileの例

```
/* Suppose you create a display that is run on several different systems,
**and one function of that display is to import an image from the file
'my_image'.
**Assume the identical directory structure exists on all systems on which
the display
**is run; however, each system requires a different directory separator.
**The following procedure creates a valid directory string for each system:
*/
```

```
PROCEDURE import_my_image(file_path VARCHAR2) IS
    the_image  OG_Object;
BEGIN
    file_path:=OG_Append_Directory(File_Path, 'home');
    file_path:=OG_Append_Directory(File_Path, 'smith');
    file_path:=OG_Append_Directory(File_Path, 'images');
    file_path:=OG_Append_File(File_Path, 'my_image');
    the_image:=OG_Import_Image(File_Path, OG_FileSystem,
        OG_Tiff_Iformat);
END;
```

```
/*Assume the initial value of file_path is 'C:¥'.On MS-DOS systems, the value of
**file_path that is passed to OG_Import_Image is:C:¥home¥smith¥images¥my_image.
*/
```

```
/*Assume the initial value of file_path is 'disk$ic1[]'.On VMS systems, the value of
**file_path that is passed to OG_Import_Image is:disk$ic1:[home.smith.images]my_image.
*/
```

OG_Center

説明

このプロシージャは、*center_pt*で表された図表のポイントがウィンドウ中央に表示されるように、指定されたウィンドウ内の図表を再描画します。

構文

```
PROCEDURE OG_Center
  (window_hdl  OG_Window,
   center_pt   OG_Point);
```

パラメータ

<i>window_hdl</i>	ウィンドウに対するハンドル。
<i>center_pt</i>	ウィンドウ中央に表示させるための図表のポイント。

OG_Centerの例

```
/* Suppose you have a chart that you want to appear in the center of a
window.
**To do this, you need to get the location and dimensions of the chart's
outer
**bounding box, calculate its center point, then use center_pt to place
this
**point in the center of the window.
*/

PROCEDURE center_chart (my_window IN og_window, my_chart IN
og_object) IS
  center_point  og_point;
  chart_record  og_chart_ca;
BEGIN
  chart_record.chart_caob.mask:=OG_OBBOX_GENERICA;
  chart_record.chart_caog_mask:=OG_NONE_GROUPA;
  chart_record.chart_caoc_mask:=OG_NONE_CHARTA;
  og_get_attr (my_chart, chart_record);
  center_pt.x:=chart_record.chart_caob.obbox.x +
(chart_record.chart_caob.obbox.width / 2);
  center_pt.y:=chart_record.chart_caob.obbox.y +
(chart_record.chart_caob.obbox.height / 2);
  og_center (my_window, center_point);
END;
```

OG_Damage (領域)

説明

このプロシージャは、レイアウトの四角形領域を破損します。

構文

```
PROCEDURE OG_Damage
  (region  OG_Rectangle);

PROCEDURE OG_Damage
  (region      OG_Rectangle,
   layer_hdl   OG_Layer);
```

パラメータ

<i>region</i>	破損した四角形領域。
<i>layer_hdl</i>	四角形領域が破損したレイヤー。 <i>layer_hdl</i> を指定していない場合、すべてのレイヤーの領域が破損されます。

使用上の注意

詳細は「ダメージ制御フラグ」を参照してください。

OG_Damage (領域)の例

```
/* The following procedure damages a 3"x2"
** area in the upper-left corner of the layout:
*/

PROCEDURE example IS
  damage_region  OG_Rectangle;
BEGIN
  damage_region.x := 0;
  damage_region.y := 0;
  damage_region.width := 3 * OG_Inch;
  damage_region.height := 2 * OG_Inch;

  OG_Damage (Damage_Region);
END;
```

OG_Get_Attr (アプリケーション)

説明

構文

```
PROCEDURE OG_Get_Attr
  (attr IN OUT  OG_App_Attr);
```

パラメータ

<i>attr</i>	アプリケーションの属性で満たされた属性レコード。
-------------	--------------------------

使用上の注意

このプロシージャは、現在実行中のGraphics Builder実行可能プログラムの属性値を取得します。属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (アプリケーション)の例

```
/* The following function returns the number of screen resolution units
** (i.e., pixels) for the current display device:
*/

FUNCTION example RETURN NUMBER IS
  rec OG_App_Attr;
BEGIN
  rec.mask := OG_Screen_Res_Appa;
  OG_Get_Attr(Rec);
  RETURN(rec.hscreen_res);
END;
```

OG_Get_Attr (軸)

説明

このプロシージャは、指定された軸の属性値を取得します。

構文

```
PROCEDURE OG_Get_Attr                                generic
  (axis_hdl IN      OG_Axis,
   attr      IN OUT  OG_Axis_Attr);
```

```
PROCEDURE OG_Get_Attr                                continuous
  (axis_hdl IN      OG_Axis,
   attr      IN OUT OG_Contaxis_Ca);

PROCEDURE OG_Get_Attr                                date
  (axis_hdl IN      OG_Axis,
   attr      IN OUT OG_Dateaxis_Ca);

PROCEDURE OG_Get_Attr                                discrete
  (axis_hdl IN      OG_Axis,
   attr      IN OUT OG_Discaxis_Ca);
```

パラメータ

<i>axis_hdl</i>	取得する属性を持つ軸に対するハンドル。
<i>attr</i>	軸の属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (軸)の例

```
/* The following function returns the custom label for the specified axis:
 */

FUNCTION example(axis OG_Axis) RETURN CHAR IS
  rec OG_Contaxis_Ca;
BEGIN
  rec.ca_axis.mask := OG_Custlabel_Axisa;
  rec.ca_cont.mask := OG_None_Contaxisa;
  OG_Get_Attr(Axis, rec);
  RETURN(rec.ca_axis.custlabel);
END;
```

OG_Get_Attr (図表)

説明

このプロシージャは、現在の図表の属性値を取得します。

構文

```
PROCEDURE OG_Get_Attr
  (attr IN OUT OG_Display_Attr);
```

パラメータ

<i>attr</i>	図表の属性で満たされた属性レコード。
-------------	--------------------

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (図表)の例

```
/* The following function returns the display width:
*/

FUNCTION example RETURN NUMBER IS
  rec OG_Display_Attr;
BEGIN
  rec.mask := OG_Size_Displaya;
  OG_Get_Attr(Rec);
  RETURN(rec.width);
END;
```

OG_Get_Attr (フィールド・テンプレート)

説明

このプロシージャは、指定されたフィールド・テンプレートの属性値を取得します。

構文

```
PROCEDURE OG_Get_Attr                                一般
  (ftemp_hdl IN      OG_Ftemp,
   attr      IN OUT  OG_Ftemp_Attr);

PROCEDURE OG_Get_Attr                                軸
  (ftemp_hdl IN      OG_Ftemp,
   attr      IN OUT  OG_Axisftemp_Ca);
```

パラメータ

<i>ftemp_hdl</i>	取得する属性を持つフィールド・テンプレートに対するハンドル。
<i>attr</i>	フィールド・テンプレートの属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (フィールド・テンプレート)の例

```
/* The following function returns the number format mask for the specified
field template:
*/

FUNCTION example(ftemp OG_Ftemp) RETURN CHAR IS
  rec OG_Axisftemp_Ca;
BEGIN
  rec.ca_ftemp.mask := OG_Numfmt_Ftempa;
  rec.ca_aftemp.mask := OG_None_Axisftempa;
  OG_Get_Attr(Ftemp, rec);
  RETURN(rec.ca_ftemp.numfmt);
END;
```

OG_Get_Attr (枠テンプレート)

説明

このプロシージャは、指定された枠テンプレートの属性値を取得します。

構文

PROCEDURE OG_Get_Attr (template_hdl IN OG_Template, attr IN OUT OG_Frame_Attr);	一般枠
PROCEDURE OG_Get_Attr (template_hdl IN OG_Template, attr IN OUT OG_Axisframe_Ca);	軸枠
PROCEDURE OG_Get_Attr (template_hdl IN OG_Template, attr IN OUT OG_Pieframe_Ca);	円グラフ枠
PROCEDURE OG_Get_Attr (template_hdl IN OG_Template, attr IN OUT OG_Tableframe_Ca);	表枠

パラメータ

<i>template_hdl</i>	取得する枠属性を持つチャート・テンプレートに対するハンドル。
<i>attr</i>	枠テンプレートの属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (枠テンプレート)の例

```
/* The following function returns the depth size for the specified template
frame:
*/

FUNCTION example(temp OG_Template) RETURN NUMBER IS
    rec OG_Axisframe_Ca;
BEGIN
    rec.ca_frame.mask := OG_Depthsize_Framea;
    rec.ca_axis.mask := OG_None_Framea;
    OG_Get_Attr(Temp, rec);
    RETURN(rec.ca_frame.depthsize);
END;
```

OG_Get_Attr (オブジェクト)

説明

このプロシージャは、指定されたオブジェクトの属性値を取得します。

構文

<pre>PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Arc_Ca);</pre>	円弧
<pre>PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Chart_Ca);</pre>	チャート
<pre>PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Generic_Attr);</pre>	一般
<pre>PROCEDURE OG_Get_Attr</pre>	図形

(object_hdl IN OG_Object, attr IN OUT OG_Graphic_Ca);	
PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Group_Ca);	グループ
PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Image_Ca);	イメージ
PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Line_Ca);	線
PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Poly_Ca);	多角形/折れ線
PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Rect_Ca);	四角形
PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Rrect_Ca);	丸い四角形
PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Symbol_Ca);	記号
PROCEDURE OG_Get_Attr (object_hdl IN OG_Object, attr IN OUT OG_Text_Ca);	テキスト/テキストフィールド

パラメータ

<i>object_hdl</i>	取得する属性を持つオブジェクトに対するハンドル。
<i>attr</i>	オブジェクトの属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (オブジェクト)の例

```
/* Suppose the user selects an object whose color determines what information
**the user is interested in viewing.The following function will take an
object
```

```

**as its **argument and return the name of its foreground fill color.The
color
**can then be determined, and the appropriate information displayed.
*/

FUNCTION get_color(the_object IN OG_Object) RETURN VARCHAR2
IS
    obj_record    OG_Graphic_Ca;
BEGIN
    obj_record.graphic_caoh.mask:=OG_Ffcolor_Graphica;
    obj_record.generic_caob.mask:=OG_None_Generica;
    OG_Get_Attr(The_Object, obj_record);
    RETURN(obj_record.graphic_caoh.ffcolor);
END;
```

OG_Get_Attr (プリンタ)

説明

このプロシージャは、現在のプリンタの属性値を取得します。

構文

```

PROCEDURE OG_Get_Attr
    (attr IN OUT OG_Printer_Attr);
```

パラメータ

attr	プリンタの属性で満たされた属性レコード。
------	----------------------

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (プリンタ)の例

```

/* The following function returns the name of the current printer:
*/

FUNCTION example RETURN CHAR IS
    rec OG_Printer_Attr;
BEGIN
    rec.mask := OG_Name_Printera;
    OG_Get_Attr(Rec);
```

```
    RETURN (rec.name);  
END;
```

OG_Get_Attr (問合せ)

説明

このプロシージャは、指定された問合せの属性値を取得します。

構文

```
PROCEDURE OG_Get_Attr  
  (query_hdl IN      OG_Query,  
   attr      IN OUT  OG_Query_Attr);
```

パラメータ

<i>query_hdl</i>	取得する属性を持つ問合せに対するハンドル。
<i>attr</i>	問合せの属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (問合せ)の例

```
/* The following function returns the SQL statement that defines the specified  
query:  
*/
```

```
FUNCTION example(query OG_Query) RETURN CHAR IS  
  rec OG_Query_Attr;  
BEGIN  
  rec.mask := OG_Querysource_Querya;  
  OG_Get_Attr(Query, rec);  
  RETURN (rec.querysource);  
END;
```

OG_Get_Attr (参照線)

説明

このプロシージャは、指定された参照線の属性値を取得します。

構文

```
PROCEDURE OG_Get_Attr
  (refline_hdl  IN      OG_Refline,
   attr         IN OUT  OG_Refline_Attr);
```

パラメータ

<i>refline_hdl</i>	取得する属性を持つ参照線に対するハンドル。
<i>attr</i>	参照線の属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (参照線)の例

```
/* The following function returns the label for the specified reference
line.
*/

FUNCTION example(refline OG_Refline) RETURN CHAR IS
  rec  OG_Refline_Attr;
BEGIN
  rec.mask := OG_Label_Reflinea;
  OG_Get_Attr(Refline, rec);
  RETURN(rec.label);
END;
```

OG_Get_Attr (サウンド)

説明

このプロシージャは、指定されたサウンドの属性値を取得します。

構文

```
PROCEDURE OG_Get_Attr
  (sound_hdl IN      OG_Sound,
   attr      IN OUT  OG_Sound_Attr);
```

パラメータ

<i>sound_hdl</i>	取得する属性を持つサウンドに対するハンドル。
<i>attr</i>	サウンドの属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (サウンド)の例

```
/* The following function returns the name of the specified sound:
*/
```

```
FUNCTION example(sound OG_Sound) RETURN CHAR IS
  rec OG_Sound_Attr;
BEGIN
  rec.mask := OG_Name_Sounda;
  OG_Get_Attr(Sound, rec);
  RETURN(rec.name);
END;
```

OG_Get_Attr (タイマー)

説明

このプロシージャは、指定されたタイマーの属性値を取得します。

構文

```
PROCEDURE OG_Get_Attr
  (timer_hdl IN      OG_Timer,
   attr      IN OUT  OG_Timer_Attr);
```

パラメータ

<i>timer_hdl</i>	取得する属性を持つタイマーに対するハンドル。
<i>attr</i>	タイマーの属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (タイマー)の例

```
/* The following function returns the procedure name assigned to the specified
timer:
*/

FUNCTION example(timer OG_Timer) RETURN CHAR IS
  rec OG_Timer_Attr;
BEGIN
  rec.mask := OG_Timerproc_Timera;
  OG_Get_Attr(Timer, rec);
  RETURN(rec.timerproc);
END;
```

OG_Get_Attr (ウィンドウ)

説明

このプロシージャは、指定されたウィンドウの属性値を取得します。

構文

```
PROCEDURE OG_Get_Attr
  (window_hdl IN      OG_Window,
   attr       IN OUT  OG_Window_Attr);
```

パラメータ

<i>window_hdl</i>	取得する属性を持つウィンドウに対するハンドル。
<i>attr</i>	ウィンドウの属性で満たされた属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが取り出されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Get_Attr (ウィンドウ)の例

```

/* The following function returns the specified window's position:
*/

FUNCTION example(window OG_Window) RETURN OG_Point IS
    rec OG_Window_Attr;
BEGIN
    rec.mask := OG_Position_Windowowa;
    OG_Get_Attr(Window, rec);
    RETURN(rec.position);
END;
```

OG_Get_Buttonproc

説明

このボタン・プロシージャは、定義済みである必要があり、PL/SQLパッケージ内に存在してはいけません。

構文

```

FUNCTION OG_Get_Buttonproc
    (proc_name VARCHAR2)
RETURN OG_Buttonproc;
```

パラメータ

<i>proc_name</i>	ハンドルが返されるPL/SQLボタン・プロシージャの名前。
------------------	-------------------------------

戻り値

指定されたボタン・プロシージャに対するハンドル。指定されたボタン・プロシージャが存在しない場合はNULLが返されます。

OG_Get_Buttonprocの例

```

/* Suppose you have written a button procedure named 'show_sales_data'
**and you want to assign that procedure to an object; then, when the user
**selects the object, the procedure will be executed.
*/

PROCEDURE make_object_button(my_obj IN OG_Object) IS
    obj_rec          OG_Generic_Attr;
    my_buttonproc    OG_Buttonproc;
BEGIN
```

```
my_buttonproc:=OG_Get_Buttonproc('Show_Sales_Data');
obj_rec.mask:=OG_Button_Generica;
obj_rec.button:=my_buttonproc;
obj_rec.events:=OG_Mouse_Down;
OG_Set_Attr(My_Obj, obj_rec);
END;
```

OG_Help

説明

このプロシージャは「ヘルプ・システム」を起動し、指定したハイパーテキスト・ターゲットで、ランタイム・ヘルプを表示します。

構文

```
PROCEDURE OG_Help
  (target VARCHAR2);
```

パラメータ

<i>target</i>	表示されるランタイム・ヘルプ内のハイパーテキスト・ターゲット。
---------------	---------------------------------

OG_Helpの例

```
/* Suppose you want the user to be able to select a button and invoke
**the Help system.You could write the following button procedure:
*/

PROCEDURE get_help (buttonobj IN OG_Object, hitobj IN
OG_Object, win IN OG_Window, eventinfo IN OG_Event) IS
BEGIN
  IF eventinfo.event_type=OG_Mouse_Up THEN
    OG_Help('Topic_1');
  END IF;
END;
```

OG_Host

説明

このプロシージャは、指定されたコマンドをオペレーティング・システムに渡します。

構文

```
PROCEDURE OG_Host
  (command VARCHAR2);
```


パラメータ

<i>command</i>	実行するコマンドを含むテキスト文字列。
----------------	---------------------

OG_Hostの例

```
/* Suppose you want to be notified via electronic mail when a user closes
a display.
**You could create a script named 'mail_me' in your file system that sends
you mail,
**and then invoke it with the following Close Display trigger:
*/
```

```
PROCEDURE send_me_mail IS
BEGIN
    OG_Host('Mail_Me');
END;
```

OG_Pause

説明

このプロシージャは、指定された秒数、図表の実行を中断します。

構文

```
PROCEDURE OG_Pause
    (secs NUMBER);
```

パラメータ

<i>secs</i>	中断する秒数。
-------------	---------

OG_Pauseの例

```
/* The following procedure suspends display execution for seven seconds:
*/
```

```
PROCEDURE example IS
BEGIN
    OG_Pause(7);
END;
```

OG_Print

説明

このプロシージャは、現在選択されている出力デバイスにレイアウト内容を出します。

構文

```
PROCEDURE OG_Print;  
PROCEDURE OG_Print  
    (window_hdl OG_Window);
```

パラメータ

<i>window_hdl</i>	出力されるウィンドウに対するハンドル。
-------------------	---------------------

使用上の注意

ウィンドウのハンドルを指定すると、そのウィンドウに表示されているレイヤーのみが出力されるか、またはレイヤーが含まれているウィンドウの種類やレイヤーの表示/非表示にかかわらず、図表のすべてのレイヤーが出力されます。

OG_Printの例

```
/* Suppose you want to print the contents of the main layout window.  
*/  
  
PROCEDURE print_main_window IS  
    the_window OG_Window;  
BEGIN  
    the_window:=OG_Get_Window('Main Layout');  
    OG_Print(The_Window);  
END;
```

OG_Quit

説明

このプロシージャは、現行のGraphics Builderセッションを終了します。

構文

```
PROCEDURE OG_Quit;
```

パラメータ

なし。

OG_Quitの例

```
/* Suppose you want to provide the user with a button that-when selected
**commits database changes and quits Graphics Builder.You could write the
**following button procedure:
*/
```

```
PROCEDURE commit_and_quit (hitobj IN OG_Object, buttonobj
    IN OG_Object, win IN OG_Window, eventinfo IN OG_Event) IS
BEGIN
    COMMIT;
    OG_Quit;
END;
```

OG_Root_Object

説明

このファンクションは、図表のルート・オブジェクトに対するハンドルを返します。

構文

```
FUNCTION OG_Root_Object
RETURN OG_Object;
```

パラメータ

なし。

戻り値

図表のルート・オブジェクトに対するハンドル。

使用上の注意

ルート・オブジェクトは、図表の最上位のオブジェクトです。そのすぐ下の子が図表のレイヤーです。

OG_Root_Objectの例

```
/* The following procedure moves the topmost layer in the display to the
bottom of the layer list:
*/
```

```
PROCEDURE example IS
    root  OG_Object;
    layer OG_Object;
BEGIN
    root := OG_Root_Object;
    layer := OG_Get_Child(Root, 0);
    OG_Insert_Child(Root, layer, OG_Last);
END;
```

OG_Set_Attr (アプリケーション)

説明

このプロシージャは、現在実行中のGraphics Builder実行可能プログラムの属性を設定します。

構文

```
PROCEDURE OG_Set_Attr
    (attr OG_App_Attr);
```

パラメータ

<i>attr</i>	新規属性値を含む属性レコード。
-------------	-----------------

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (アプリケーション)の例

```
/* The following procedure sets the cursor to 'busy':
*/

PROCEDURE example IS
    attr OG_App_Attr;
BEGIN
    attr.cursor := 'busy';
    attr.mask := OG_Cursor_Appa;
    OG_Set_Attr(attr);
END;
```

OG_Set_Attr (軸)

説明

このプロシージャは、指定された軸の属性値を設定します。

構文

```
PROCEDURE OG_Set_Attr                                generic
    (axis_hdl   OG_Axis,
      attr      OG_Axis_Attr);

PROCEDURE OG_Set_Attr                                continuous
    (axis_hdl   OG_Axis,
      attr      OG_Contaxis_Ca);

PROCEDURE OG_Set_Attr                                date
    (axis_hdl   OG_Axis,
      attr      OG_Dateaxis_Ca);

PROCEDURE OG_Set_Attr                                discrete
    (axis_hdl   OG_Axis,
      attr      OG_Discaxis_Ca);
```

パラメータ

<i>axis_hdl</i>	設定する属性を持つ軸に対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (軸)の例

```
/* The following procedure sets the custom label for the specified axis:
*/

PROCEDURE example(axis OG_Axis, label VARCHAR2) IS
    rec OG_Contaxis_Ca;
BEGIN
    rec.ca_axis.custlabel := 'New Label';
    rec.ca_axis.mask := OG_Custlabel_Axisa;
    rec.ca_cont.mask := OG_None_Contaxisa;
    OG_Set_Attr(Axis, rec);
END;
```

OG_Set_Attr (チャート要素)

説明

このプロシージャは、横棒や円弧などチャート要素の属性を設定します。

構文

```
PROCEDURE OG_Set_Attr
  (chart_hdl  OG_Object,
   row_num    NUMBER,
   col_name   VARCHAR2,
   attr       OG_Chelement_Ca);
```

パラメータ

<i>chart_hdl</i>	設定する属性を持つデータ値を含むチャートに対するハンドル。
<i>row_num</i>	設定する属性を持つデータ値の行数。
<i>col_name</i>	設定する属性を持つデータ値の列の名前。
<i>attr</i>	新規属性値を含む属性レコード。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

対応するデータ値のチャート、行および列とチャート設定属性レコードを指定する必要があります。属性レコードには、指定されたデータ値を表すチャート要素に適用される図形および他の属性が含まれます。たとえば、横棒に対応するデータ値の属性レコードを指定することにより、棒グラフの横棒のカラーを設定できます。

チャート要素の変更は、OG_Update_Chartのコールによりチャートが更新されるまで適用されないことに注意してください。

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (チャート要素)の例

```
/* The following procedure loops through all rows in a chart's query,
**and then sets the color of each bar in the chart based on its value:
*/

PROCEDURE OGTRIGGERPROC0 IS
chart og_object;
```

```
query og_query;
rec og_chelement_ca;
total number;
bar_val number;

BEGIN
  chart := og_get_object ('Employees');
  query:= og_get_query ('query0');
  og_execute_query (query);
  og_start_from(query, OG_NEWDATA);
  total := og_numrows (query, OG_NEWDATA);
  for i in 0..total-1 loop
    bar_val:=og_get_numcell (query, OG_NEWDATA,'SAL');
    IF bar_val>2000 THEN^M
      rec.chelement_cagr.mask :=OG_BFCOLOR_GRAPHICA;
      rec.chelement_cace.mask :=OG_NONE_CHELEMENTA;
      rec.chelement_cagr.bfcolor := 'cyan';
    og_set_attr (chart, i, 'SAL', rec);
    END; IF;
    og_next_row(query, OG_NEWDATA);
  END LOOP;
  og_update_chart (chart, OG_ALL_CHUPDA);
END;
```

OG_Set_Attr (図表)

説明

このプロシージャは、現在の図表の属性を設定します。

構文

```
PROCEDURE OG_Set_Attr
  (attr OG_Display_Attr);
```

パラメータ

<i>attr</i>	新規属性値を含む属性レコード。
-------------	-----------------

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (図表)の例

```
/* The following procedure sets the display width:
*/

PROCEDURE example IS
  rec OG_Display_Attr;
BEGIN
  rec.width := 4 * OG_Inch;
  rec.height := 5 * OG_Inch;
  rec.mask := OG_Size_Displaya;
  OG_Set_Attr(Rec);
END;
```

OG_Set_Attr (フィールド・テンプレート)

説明

このプロシージャは、指定されたフィールド・テンプレートの属性値を設定します。

構文

```
PROCEDURE OG_Set_Attr                                generic
  (ftemp_hdl OG_Ftemp,
   attr      OG_Ftemp_Attr);

PROCEDURE OG_Set_Attr                                axis
  (ftemp_hdl OG_Ftemp,
   attr      OG_Axisftemp_Ca);
```

パラメータ

<i>ftemp_hdl</i>	設定する属性を持つフィールド・テンプレートに対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (フィールド・テンプレート)の例

```
/* The following procedure sets the number format mask for the specified
field template:
*/
```



```

PROCEDURE example(ftemp OG_Ftemp) IS
  rec OG_Axisftemp_Ca;
BEGIN
  rec.ca_ftemp.numfmt := '9,990';
  rec.ca_ftemp.mask := OG_Numfmt_Ftempa;
  rec.ca_aftemp.mask := OG_None_Axisftempa;
  OG_Set_Attr(Ftemp, rec);
END;
```

OG_Set_Attr (枠テンプレート)

説明

このプロシージャは、指定された枠テンプレートの属性値を設定します。

構文

```

PROCEDURE OG_Set_Attr                                generic frame
  (template_hdl OG_Template,
   attr          OG_Frame_Attr);

PROCEDURE OG_Set_Attr                                axis frame
  (template_hdl OG_Template,
   attr          OG_Axisframe_Ca);

PROCEDURE OG_Set_Attr                                pie frame
  (template_hdl OG_Template,
   attr          OG_Pieframe_Ca);

PROCEDURE OG_Set_Attr                                table frame
  (template_hdl OG_Template,
   attr          OG_Tableframe_Ca);
```

パラメータ

<i>template_hdl</i>	設定する枠属性を持つチャート・テンプレートに対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (枠テンプレート)の例

```

/* The following procedure sets the depth size for the specified template
frame:
```

```
*/  
  
PROCEDURE example(temp OG_Template) IS  
  rec OG_Axisframe_Ca;  
BEGIN  
  rec.ca_frame.depthsize := OG_Large_Depthsize;  
  rec.ca_frame.mask := OG_Depthsize_Framea;  
  rec.ca_axis.mask := OG_None_Framea;  
  OG_Set_Attr(Temp, rec);  
END;
```

OG_Set_Attr (オブジェクト)

説明

このプロシージャは、指定されたオブジェクトの属性を設定します。

構文

```
PROCEDURE OG_Set_Attr                                arc  
  (object_hdl OG_Object,  
   attr       OG_Arc_Ca,  
   damage     BOOLEAN      := TRUE,  
   update_bbox BOOLEAN      := TRUE);  
  
PROCEDURE OG_Set_Attr                                chart  
  (object_hdl OG_Object,  
   attr       OG_Chart_Ca,  
   damage     BOOLEAN      := TRUE,  
   update_bbox BOOLEAN      := TRUE);  
  
PROCEDURE OG_Set_Attr                                generic  
  (object_hdl OG_Object,  
   attr       OG_Generic_Attr,  
   damage     BOOLEAN      := TRUE,  
   update_bbox BOOLEAN      := TRUE);  
  
PROCEDURE OG_Set_Attr                                graphic  
  (object_hdl OG_Object,  
   attr       OG_Graphic_Ca,  
   damage     BOOLEAN      := TRUE,  
   update_bbox BOOLEAN      := TRUE);  
  
PROCEDURE OG_Set_Attr                                group  
  (object_hdl OG_Object,  
   attr       OG_Group_Ca,  
   damage     BOOLEAN      := TRUE,  
   update_bbox BOOLEAN      := TRUE);  
  
PROCEDURE OG_Set_Attr                                image
```

```

(object_hdl  OG_Object,
 attr       OG_Image_Ca,
 damage     BOOLEAN      := TRUE,
 update_bbox BOOLEAN      := TRUE);

PROCEDURE OG_Set_Attr                                line
(object_hdl  OG_Object,
 attr       OG_Line_Ca,
 damage     BOOLEAN      := TRUE,
 update_bbox BOOLEAN      := TRUE);

PROCEDURE OG_Set_Attr                                polygon/polyline
(object_hdl  OG_Object,
 attr       OG_Poly_Ca,
 damage     BOOLEAN      := TRUE,
 update_bbox BOOLEAN      := TRUE);

PROCEDURE OG_Set_Attr                                rectangle
(object_hdl  OG_Object,
 attr       OG_Rect_Ca,
 damage     BOOLEAN      := TRUE,
 update_bbox BOOLEAN      := TRUE);

PROCEDURE OG_Set_Attr                                rounded rectangle
(object_hdl  OG_Object,
 attr       OG_Rrect_Ca,
 damage     BOOLEAN      := TRUE,
 update_bbox BOOLEAN      := TRUE);

PROCEDURE OG_Set_Attr                                symbol
(object_hdl  OG_Object,
 attr       OG_Symbol_Ca,
 damage     BOOLEAN      := TRUE,
 update_bbox BOOLEAN      := TRUE);

PROCEDURE OG_Set_Attr                                text/text field
(object_hdl  OG_Object,
 attr       OG_Text_Ca,
 damage     BOOLEAN      := TRUE,
 update_bbox BOOLEAN      := TRUE); パラメータ

```

パラメータ

<i>object_hdl</i>	設定する属性を持つオブジェクトに対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (オブジェクト)の例

```
/* Suppose you have a map of the world and you want to change the
**color of one of the countries.First, you would get the handle to the
**country object, then you would change its color.
*/

PROCEDURE color_country (country_name) IS
  my_object      OG_Object;
  obj_record     OG_Graphic_Ca;
BEGIN
  my_object:=OG_Get_Object(Country_Name);
  obj_record.graphic_caob.mask:=OG_None_Generica;
  obj_record.graphic_caoh.mask:=OG_Ffcolor_Graphica;
  obj_record.graphic_caoh.ffcolor:='red';
  OG_Set_Attr(My_Object, obj_record);
END;
```

OG_Set_Attr (プリンタ)

説明

このプロシージャは、現在のプリンタの属性値を設定します。

構文

```
PROCEDURE OG_Set_Attr
  (attr OG_Printer_Attr);
```

パラメータ

attr	新規属性値を含む属性レコード。
------	-----------------

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (プリンタ)の例

```
/* The following procedure tells sets the number of copies to print:
*/

PROCEDURE example IS
    rec OG_Printer_Attr;
BEGIN
    rec.copies := 2;
    rec.mask := OG_Copies_Printera;
    OG_Set_Attr(rec);
END;
```

OG_Set_Attr (問合せ)

説明

このプロシージャは、指定された問合せの属性を設定します。

構文

```
PROCEDURE OG_Set_Attr
    (query_hdl OG_Query,
      attr      OG_Query_Attr);
```

パラメータ

<i>query_hdl</i>	設定する属性を持つ問合せに対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (問合せ)の例

```
/* The following procedure sets the SQL statement that defines the specified
query:
*/

PROCEDURE example(query OG_Query) IS
    rec OG_Query_Attr;
BEGIN
    rec.querysource := 'select ename, sal from emp';
```

```
rec.mask := OG_Querysource_Queryya;  
OG_Set_Attr(Query, rec);  
END;
```

OG_Set_Attr (参照線)

説明

このプロシージャは、指定された参照線の属性を設定します。

構文

```
PROCEDURE OG_Set_Attr  
  (refline_hdl  OG_Refline,  
   attr         OG_Refline_Attr);
```

パラメータ

<i>refline_hdl</i>	設定する属性を持つ参照線に対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (参照線)の例

```
/* The following procedure sets the label for the specified reference line.  
*/  
  
PROCEDURE example(refline OG_Refline) IS  
  rec OG_Refline_Attr;  
BEGIN  
  rec.label := 'Average';  
  rec.mask := OG_Label_Reflinea;  
  OG_Set_Attr(Refline, rec);  
END;
```

OG_Set_Attr (サウンド)

説明

このプロシージャは、指定されたサウンドの属性を設定します。

構文

```
PROCEDURE OG_Set_Attr
  (sound_hdl  OG_Sound,
   attr       OG_Sound_Attr);
```

パラメータ

<i>sound_hdl</i>	設定する属性を持つサウンドに対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (サウンド)の例

```
/* The following procedure sets the name of the specified sound:
*/

PROCEDURE example(sound OG_Sound) IS
  rec  OG_Sound_Attr;
BEGIN
  rec.name := 'Alert';
  rec.mask := OG_Name_Sounda;
  OG_Set_Attr(Sound, rec);
END;
```

OG_Set_Attr (タイマー)

説明

このプロシージャは、指定されたタイマーの属性を設定します。

構文

```
PROCEDURE OG_Set_Attr
```

```
(timer_hdl  OG_Timer,  
  attr      OG_Timer_Attr);
```

パラメータ

<i>timer_hdl</i>	設定する属性を持つタイマーに対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (タイマー)の例

```
/* The following procedure sets the procedure name assigned to the specified  
timer:  
*/  
  
PROCEDURE example(timer OG_Timer) IS  
  rec OG_Timer_Attr;  
BEGIN  
  rec.timerproc := 'update_proc';  
  rec.mask := OG_Timerproc_Timera;  
  OG_Set_Attr(timer, rec);  
END;
```

OG_Set_Attr (ウィンドウ)

説明

このプロシージャは、指定されたウィンドウの属性を設定します。

構文

```
PROCEDURE OG_Set_Attr  
  (window_hdl  OG_Window,  
   attr        OG_Window_Attr);
```

パラメータ

<i>window_hdl</i>	設定する属性を持つウィンドウに対するハンドル。
<i>attr</i>	新規属性値を含む属性レコード。

使用上の注意

属性レコード内のマスク属性値によって指定される属性値のみが設定されます。マスクが設定されていない属性レコード内のフィールドは、このプロシージャのコールによって影響を受けません。

OG_Set_Attr (ウィンドウ)の例

```
/* The following procedure sets the specified window's size:
*/

PROCEDURE example(window OG_Window) IS
  rec OG_Window_Attr;
BEGIN
  rec.width := 4 * OG_Get_Ap_Hscreen_Res;
  rec.height := 5 * OG_Get_Ap_Vscreen_Res;
  rec.mask := OG_Size_Windowowa;
  OG_Set_Attr(Window, rec);
END;
```

OG_Translate_Envvar

説明

このファンクションは、指定された環境変数値を返します。

構文

```
FUNCTION OG_Translate_Envvar
  (envvar VARCHAR2)
RETURN VARCHAR2;
```

パラメータ

<i>envvar</i>	変換する環境変数。
---------------	-----------

戻り値

環境変数値を含む文字列。

使用上の注意

環境変数の取り扱いやその存在は、システムによって異なります。詳細は、使用しているオペレーティング・システムのGraphics Builderマニュアルを参照してください。

環境変数を検索する場合、Graphics Builderは最初にオペレーティング・システムをチェックし、環境変数が定義されているかどうかを確認します。定義されていない場合は、環境設定ファイル内を検索します。

OG_Translate_Envvarの例

```
/* Suppose your system has an environment variable named IMAGE_DIR
**that specifies the directory path of the image file 'my_image'.The
following
**procedure imports that image:
*/

PROCEDURE import_my_image IS
    the_image  OG_Object;
    file_path  VARCHAR2(50);
BEGIN
    file_path:=OG_Translate_Envvar('Image_Dir');
    file_path:=OG_Append_File(File_Path, 'my_image');
    the_image:=OG_Import_Image(File_Path, OG_FileSystem,
        OG_Tiff_Iformat);
END;
```

OG_User_Exit

説明

このプロシージャは、ユーザー定義実行可能プログラムを実行します。

構文

```
PROCEDURE OG_User_Exit
    (command VARCHAR2);
```

パラメータ

<i>command</i>	ユーザー・イグジットおよびそれに渡す引数の名前。
----------------	--------------------------

OG_User_Exitの例

```
/* Suppose your display controls the operation of hardware
**components connected to your system.When the user selects
**a button, you may want to invoke the hardware controller
**routine, which you have linked in as a user exit.In addition,
**you may want to pass an argument to this user exit.The following
**procedure invokes the user exit 'hw_ctrl' with the parameter 'signal':
*/

PROCEDURE control_hw(buttonobj IN OG_Object, hitobj IN OG_Object, win IN
```

```
OG_Window, eventinfo IN OG_Event) IS
BEGIN
    OG_User_Exit('Hw_Ctrl' || :signal);
END;
```

パラメータ・ビルトイン

OG_Delete_Param
OG_Get_Char_Param
OG_Get_Date_Param
OG_Get_Num_Param
OG_Get_Param_Type
OG_Param_Exists
OG_Set_Param

OG_Delete_Param

説明

このプロシージャで、指定したパラメータを削除します。

構文

```
PROCEDURE OG_Delete_Param  
  (param_name VARCHAR2);
```

パラメータ

<i>param_name</i>	削除するパラメータの名前。
-------------------	---------------

OG_Delete_Paramの例

```
/* The following procedure deletes the parameter 'param0':  
*/
```

```
PROCEDURE example IS  
BEGIN  
  OG_Delete_Parameter('Param0');  
END;
```

OG_Get_Char_Param

説明

このファンクションは、指定したデータ型がCHAR型のパラメータの値を取得するために使用します。このファンクションは、パラメータに対するバインド参照と等価です。

構文

```
FUNCTION OG_Get_Char_Param  
  (param_name VARCHAR2)  
RETURN VARCHAR2;
```

パラメータ

<i>param_name</i>	取得する値のパラメータ名。
-------------------	---------------

戻り値

指定したパラメータの値。

使用上の注意

このファンクションは、ライブラリに保存されているプログラム単位内でパラメータを参照する場合に便利です。バインド参照の場合、ライブラリのコンテキスト内ではコンパイルは失敗しますが、このファンクションの場合はコンパイルは成功します。

OG_Get_Char_Paramの例

```
/* The following procedure gets the value of the parameter 'status',  
**and changes the color of the specified object based on its value:  
*/
```

```
PROCEDURE example(object OG_Object) IS  
  stat VARCHAR2(10);  
BEGIN  
  stat := OG_Get_Char_Param('Status');  
  IF stat = 'obsolete' THEN  
    OG_Set_Fillcolor(Object, 'red');  
  END IF;  
END;
```

OG_Get_Date_Param

説明

このファンクションは、指定したデータ型がDATE型のパラメータの値を取得するために使用します。このファンクションは、パラメータに対するバインド参照と等価です。

構文

```
FUNCTION OG_Get_Date_Param
  (param_name  VARCHAR2,
   fmt         VARCHAR2)
RETURN DATE;
```

パラメータ

<i>param_name</i>	取得したい値のパラメータ名。
-------------------	----------------

戻り値

指定したパラメータの値。

使用上の注意

このファンクションは、ライブラリに保存されているプログラム単位内でパラメータを参照する場合に便利です。バインド参照の場合、ライブラリのコンテキスト内ではコンパイルは失敗しますが、このファンクションの場合はコンパイルは成功します。

OG_Get_Date_Paramの例

```
/* The following procedure gets the value of the parameter 'due_date',
**and changes the color of the specified object based on its value:
*/

*/

PROCEDURE example(object OG_Object) IS
  due DATE;
BEGIN
  due := OG_Get_Date_Param('Due_Date');
  IF due < sysdate THEN
    OG_Set_Fillcolor(Object, 'red');
  END IF;
END;
```

OG_Get_Num_Param

説明

このファンクションは、指定したデータ型がNUMBER型のパラメータの値を取得するために使用します。このファンクションは、パラメータに対するバインド参照と等価です。

構文

```
FUNCTION OG_Get_Num_Param  
  (param_name VARCHAR2)  
RETURN NUMBER;
```

パラメータ

<i>param_name</i>	取得する値のパラメータ名。
-------------------	---------------

戻り値

指定したパラメータの値。

使用上の注意

このファンクションは、ライブラリに保存されているプログラム単位内でパラメータを参照する場合に便利です。バインド参照の場合、ライブラリのコンテキスト内ではコンパイルは失敗しますが、このファンクションの場合はコンパイルは成功します。

OG_Get_Num_Paramの例

```
/* The following procedure gets the value of the parameter 'priority',  
and increases it by 1:  
*/  
  
*/  
  
PROCEDURE example IS  
  val NUMBER;  
BEGIN  
  val := OG_Get_Num_Param('Priority');  
  OG_Set_Param('Priority', val + 1);  
END;
```

OG_Get_Param_Type

説明

このファンクションを使用すると、パラメータのデータ型が戻ります。

構文

```
FUNCTION OG_Get_Param_Type  
  (param_name VARCHAR2)  
RETURN NUMBER;
```

パラメータ

<i>param_name</i>	パラメータの名前です。
-------------------	-------------

戻り値

次のビルトイン定数のいずれか。

- OG_Char_Paramtype
- OG_Date_Paramtype
- OG_Num_Paramtype

OG_Get_Param_Typeの例

```
/* The following procedure retrieves the datatype of the parameter 'param0',  
**then increases it by one if the type is NUMBER:  
*/  
  
*/  
  
PROCEDURE example IS  
  dtype NUMBER;  
BEGIN  
  dtype := OG_Get_Param_Type('Param0');  
  IF dtype = OG_Num_Paramtype THEN  
    :param0 := :param0 + 1;  
  END IF;  
END;
```


OG_Param_Exists

説明

このファンクションを使用して、特定のパラメータが作成されているかどうかを判別します。

構文

```
FUNCTION OG_Param_Exists  
  (param_name VARCHAR2)  
RETURN BOOLEAN;
```

パラメータ

<i>param_name</i>	パラメータの名前です。
-------------------	-------------

戻り値

TRUE	そのパラメータが存在する場合
FALSE	そのパラメータが存在しない場合

OG_Param_Existsの例

```
/* The following procedure assigns drill-down behavior to a chart, but  
first verifies  
**that the parameter it sets exists (and creates it if it doesn't exist):  
*/  
  
*/  
  
PROCEDURE example(chart OG_Object, param_name VARCHAR2) IS  
  chelement_group OG_Object;  
BEGIN  
  IF NOT OG_Param_Exists(param_name) THEN  
    OG_Set_Param(param_name, 10);  
  END IF;  
  
  chelement_group := OG_Get_Object('Sal_Bars', chart);  
  OG_Set_Setparam(Chelement_Group, param_name);  
  OG_Set_Keycol(Chelement_Group, 'DEPTNO');  
END;
```

OG_Set_Param

説明

このプロシージャで、指定したパラメータの値を設定します。指定したパラメータが存在しない場合、そのパラメータが作成されます。

構文

```
PROCEDURE OG_Set_Param                                date
  (param_name    VARCHAR2,
   param_value    DATE,
   param_format   VARCHAR2  :=  'DD-MON-YY');

PROCEDURE OG_Set_Param                                number
  (param_name    VARCHAR2,
   param_value    NUMBER);

PROCEDURE OG_Set_Param                                char
  (param_name    VARCHAR2,
   param_value    VARCHAR2);
```

パラメータ

<i>param_name</i>	設定する値のパラメータ名。
<i>param_value</i>	パラメータに設定する値。
<i>param_format</i>	<i>param_value</i> を日付パラメータに変換するための書式マスク。

使用上の注意

このプロシージャは、ライブラリに保存されているプログラム単位内でパラメータを参照する場合に便利です。バインド参照の場合、ライブラリのコンテキスト内ではコンパイルは失敗しますが、このプロシージャではコンパイルは成功します。

OG_Set_Paramの例

```
PROCEDURE example IS
  val NUMBER;
BEGIN
  val := OG_Get_Num_Param('Priority');
  OG_Set_Param('Priority', val + 1);
END;
```

ビルトイン問合せ

OG_Append_Row
OG_Clear_Query
OG_Data_Changed
OG_Data_Queried
OG_Destroy (問合せ)
OG_Execute_Query
OG_Get_Charcell
OG_Get_Datecell
OG_Get_Newrows
OG_Get_Numcell
OG_Get_Query
OG_Get_Schema
OG_Insert_Column
OG_Make_Query
OG_Next_Row
OG_Numcols
OG_Numrows
OG_Set_Charcell
OG_Set_Datecell
OG_Set_Numcell
OG_Set_Schema
OG_Start_From

OG_Append_Row

説明

このプロシージャは、現在の行バッファをカスタム問合せの1番下に追加します。

構文

```
PROCEDURE OG_Append_Row  
  (query_hdl IN OG_Query);
```

パラメータ

<code>query_hdl</code>	行バッファが追加される問合せに対するハンドルです。
------------------------	---------------------------

使用上の注意

OG_Set_Charcell、OG_Set_DatecellおよびOG_Set_Numcellを使用して、行バッファの内容を指定してください。

OG_Append_Rowの例

```
/* Suppose you want to create a custom query using the ENAME, SAL, and  
**HIREDATE columns in the existing query 'query0' as a basis. However, **in  
the new query, you want to double every SAL value. The following **procedure  
is a custom query procedure you could use:  
*/
```

```
*/
```

```
PROCEDURE OGQUERYPROC0 (query IN OG_Query) IS  
  other_ename   VARCHAR2(10);  
  other_sal     NUMBER(7,2);  
  other_query   OG_Query;  
  other_hiredate DATE;  
  row_count     NUMBER;  
BEGIN  
  OG_Clear_Query(Query);  
  
  other_query := OG_Get_Query('Query0');  
  row_count := OG_Numrows(Other_Query, OG_Newdata);  
  OG_Start_From(Other_Query, OG_Newdata, 0);  
  
  FOR i IN 0..row_count loop  
    other_ename := OG_Get_Charcell(Other_Query, 'ENAME');  
    other_sal := OG_Get_Numcell(Other_Query, 'SAL');  
    other_hiredate := OG_Get_Numcell(Other_Query, 'HIREDATE');  
    OG_Set_Charcell(Query, 'ENAME', other_ename);
```

```

        OG_Set_Numcell(Query, 'SAL', other_sal * 2);
        OG_Set_Datecell(Query, 'HIREDATE', other_hiredate);
        OG_Append_Row(Query);
        OG_Next_Row(Other_Query, OG_Newdata);
    END LOOP;

```

```
END;
```

OG_Clear_Query

説明

このプロシージャは、指定された問合せからすべての行を削除します。

構文

```

PROCEDURE OG_Clear_Query
    (query_hdl   OG_Query);

```

パラメータ

<i>query_hdl</i>	消去する問合せに対するハンドル。
------------------	------------------

OG_Clear_Queryの例

```

/* Suppose you want to create a custom query using the ENAME, SAL, and
**HIREDATE columns in the existing query 'query0' as a basis. However, **in
the new query, you want to double every SAL value. The following **procedure
is a custom query procedure you could use:
*/

```

```
*/
```

```

PROCEDURE OGQUERYPROC0(query IN OG_Query) IS
    other_ename    VARCHAR2(10);
    other_sal      NUMBER(7,2);
    other_query    OG_Query;
    other_hiredate DATE;
    row_count      NUMBER;
BEGIN
    OG_Clear_Query(Query);

    other_query := OG_Get_Query('Query0');
    row_count := OG_Numrows(Other_Query, OG_Newdata);
    OG_Start_From(Other_Query, OG_Newdata, 0);

    FOR i IN 0..row_count loop
        other_ename := OG_Get_Charcell(Other_Query, 'ENAME');
        other_sal := OG_Get_Numcell(Other_Query, 'SAL');
    
```

```
other_hiredate := OG_Get_Numcell (Other_Query, 'HIREDATE');
OG_Set_Charcell (Query, 'ENAME', other_ename);
OG_Set_Numcell (Query, 'SAL', other_sal * 2);
OG_Set_Datecell (Query, 'HIREDATE', other_hiredate);
OG_Append_Row (Query);
OG_Next_Row (Other_Query, OG_Newdata);
END LOOP;

END;
```

OG_Data_Changed

説明

このファンクションは、指定された問合せのOG_Execute_Queryの最新のコールから生じた新規データと旧データを比較します。データ・セットが異なる場合はTRUEを、データ・セットが同じ場合はFALSEを返します。

構文

```
FUNCTION OG_Data_Changed
  (query_hdl OG_Query)
RETURN BOOLEAN;
```

パラメータ

<i>query_hdl</i>	問合せに対するハンドル。
------------------	--------------

戻り値

TRUE	データが変更された場合。
FALSE	データが変更されていない場合。

使用上の注意

このファンクションは、新規データと旧データの次の項目を比較し、不一致を検出した時点で停止します。

- 1 返された行数
- 2 問合せスキーマ
- 3 データのセルごとの比較 (データ・セットが大きい場合は、この比較は時間がかかることに注意してください)

OG_Data_Changedの例

```
/* Suppose you want to update a chart periodically, but only if the
**data has changed.You could write the following timer trigger:
*/

*/

PROCEDURE my_timer IS
    my_query    OG_Query;
    my_chart    OG_Object;
BEGIN
    my_query:=OG_Get_Query('Emp_Query');
    OG_Execute_Query(My_Query);
    IF OG_Data_Changed(My_Query) THEN
        my_chart:=OG_Get_Object('Emp_Chart');
        OG_Update_Chart(My_Chart, OG_All_Chupda);
    END IF;
END;
```

OG_Data_Queried

説明

このファンクションは、指定された問合せのOG_Execute_Queryの最新のコールによって、指定されたデータ項目が問合せをされたかどうかを判別します。

構文

```
FUNCTION OG_Data_Queried
    (query_hdl    OG_Query,
     which_data   NUMBER)
RETURN BOOLEAN;
```

パラメータ

<i>query_hdl</i>	問合せに対するハンドル。
<i>which_data</i>	新規データまたは旧データのステータスをチェックするかどうかを指定します。Graphics Builderには、この引数の値として使用可能なビルトイン数値定数が2つあります。OG_NewdataとOG_Olddataです。

戻り値

TRUE	データが問合せをされた場合。
FALSE	データが問合せをされなかった場合。

使用上の注意

OG_Execute_Queryにより問合せが実行されなかった場合、データ項目の問合せも実行されずに、このファンクションはFALSEを返します。問合せが1回のみ実行された場合、このファンクションは新規データにTRUEを、旧データにFALSEを返します。問合せが2回以上実行された場合、このファンクションは常にTRUEを返します。

OG_Data_Queriedの例

```
/* Suppose you want to use OG_Data_Changed to check if a query's
**data has changed, and then update a chart that uses that query.
**Before you do so, you may want to make sure that both the old and
**new data for the query have been queried.
*/

PROCEDURE check_and_update IS
    my_query    OG_Query;
    my_chart    OG_Object;
BEGIN
    my_query:=OG_Get_Query('Sales Query');
    IF OG_Data_Queried(My_Query, OG_Olddata) AND
    OG_Data_Queried(My_Query, OG_Newdata) THEN
        IF data_changed(my_query) THEN
            my_chart:=OG_Get_Object('Sales Chart');
            OG_Update_Chart(My_Chart, OG_All_Chupda);
        END IF;
    END IF;
END;
```

OG_Destroy (問合せ)

説明

このプロシージャは、指定された問合せを破棄します。

構文

```
PROCEDURE OG_Destroy
    (query_hdl OG_Query);
```

パラメータ

query_hdl	破棄する問合せに対するハンドル。
-----------	------------------

OG_Destroy (Query)の例

```
/* The following procedure destroys the specified query:
*/
```



```
*/

PROCEDURE destroy_query(query_name VARCHAR2) IS
    query OG_Query;
BEGIN
    query := OG_Get_Query(Query_Name);
    OG_Destroy(Query);
END;
```

OG_Execute_Query

説明

このプロシージャは、指定された問合せを実行し、その結果を内部に格納します。

構文

```
PROCEDURE OG_Execute_Query
    (query_hdl OG_Query);
```

パラメータ

query_hdl	実行する問合せに対するハンドル。
-----------	------------------

使用上の注意

Builderで問合せを定義する必要があります。問合せがデータベースへのアクセスを必要としているときにデータベースが接続されていない場合、OG_No_Database_Connection例外状況が発生します。このプロシージャはデータを取り出すのみです。データをチャートに適用したり、他の方法でデータを操作しないでください。

各問合せでは、現在の問合せの結果（"新規"データ）および前回の問合せの結果（"旧"データ）の2つのデータ・セットが格納されます。したがって、前回問合せが実行された後でデータが変更されたときのみチャートを更新するなど、データの変更に基づく操作を実行できます。データの操作およびチェックが可能な他のビルトイン・プロシージャおよびファンクションにより、使用するデータ・セットを指定できます。

このプロシージャのコールによって問合せが実行されなかった場合は、問合せ用の新規データおよび旧データは存在しません。問合せが最初に実行されたとき、その結果は新規データとして格納されますが、旧データは存在しません。その後、問合せが実行されるたびに旧データが破棄され、既存の新規データが旧データになり、問合せの最新結果が新規データとして格納されます。

問合せが実行されるたびに、新規データの暗黙的カーソルが作成されます。（他のプロシージャやファンクションのなかには、このカーソルを操作してデータをチェックできるものもありま

す)。新規データが旧データとして再分類されるときに、カーソル（およびデータ・リスト内のカーソルの現在位置）は維持されます。一方、新規カーソルはデータの有効行を自動的に指し示さないことに注意してください。使用するカーソルを準備するには、OG_Start_Fromを使用します。

OG_Execute_Queryの例

```
/* Suppose you want to update a chart periodically.
**You could write the following timer trigger:
*/

*/

PROCEDURE every_30_secs IS
    the_query    OG_Query;
    the_chart    OG_Object;
BEGIN
    the_query:=OG_Get_Query('Emp_Query');
    the_chart:=OG_Get_Object('Emp_Chart');
    OG_Execute_Query(The_Query);
    OG_Update_Chart(The_Chart, OG_All_Chupda);
END;
```

OG_Get_Charcell

説明

このファンクションは、指定された列の現在のデータ行で、指定された問合せの文字データ値を返します。

構文

```
FUNCTION OG_Get_Charcell
(query_hdl    OG_Query,
 col_name    VARCHAR2)
RETURN VARCHAR2;

FUNCTION OG_Get_Charcell
(query_hdl    OG_Query,
 which_data   NUMBER,
 col_name    VARCHAR2)
RETURN VARCHAR2;
```

パラメータ

<i>query_hdl</i>	返すデータを含む問合せに対するハンドル。
<i>col_name</i>	返すデータを含む列の名前。

<i>which_data</i>	セル値を旧データまたは新規データのどちらから取り出すかを指定します。指定しなかった場合、この引数値にはデフォルトのOG_Newdataが設定されます。この引数の値には、次のビルトイン定数のいずれかを指定してください。
OG_Newdata	セル値を新規データから取り出します。
OG_Olddata	セル値を旧データから取り出します。

戻り値

指定されたデータ・セルの内容。

使用上の注意

現在の行は、問合せを実行するためにOG_Execute_Queryを使用するときに、最初に作成される問合せの暗黙的カーソルにより決定されます。

データ表に表示された問合せ結果を表示する場合、このファンクションは現在の行と指定された列の交差するセルに含まれるデータを返します。

このファンクションを使用して、CHAR、VARCHAR2またはRAW型の列からデータ値を返すことができます。

OG_Get_Charcellの例

```

/*Suppose that you have a chart that
**displays employee salaries.The following
**procedure uses a format trigger to paint
**a specific employee's salary column
**yellow (in this case, "Scott's").
*/

PROCEDURE CharCell (elem IN og_object,
query IN og_query) IS
ename varchar2(10);
BEGIN
ename := og_get_charcell(query, OG_NEWDATA, 'ename');
if ename = 'SCOTT' then
og_set_bfcolor(elem, 'yellow');
end if;
END;
```

OG_Get_Datecell

説明

このファンクションは、指定された列の現在のデータ行で、指定された問合せの日付データ値を返します。

構文

```
FUNCTION OG_Get_Datecell
  (query_hdl   OG_Query,
   which_data  NUMBER,
   col_name    VARCHAR2)
RETURN DATE;
```

パラメータ

<i>query_hdl</i>	返すデータを含む問合せに対するハンドル。
<i>col_name</i>	返すデータを含む列の名前。
<i>which_data</i>	セル値を旧データまたは新規データのどちらから取り出すかを指定します。 指定しなかった場合、この引数値にはデフォルトのOG_Newdataが設定されます。 この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Newdata セル値を新規データから取り出します。 OG_Olddata セル値を旧データから取り出します。

戻り値

指定されたデータ・セルの内容。

使用上の注意

現在の行は、問合せを実行するためにOG_Execute_Queryを使用するときに、最初に作成される問合せの暗黙的カーソルにより決定されます。

データ表に表示された問合せ結果を表示する場合、このファンクションは現在の行と指定された列の交差するセルに含まれるデータを返します。

OG_Get_Datecellの例

```
/* Suppose you have a bar chart that
**shows the hire dates of several employees.
**The following format trigger changes the
**column color for employees with
**a HireDate prior to '28-SEP-81'
**to green.
*/
```

```
PROCEDURE DateCell (elem IN og_object,  
                    query IN og_query) IS  
    HireDate date;  
BEGIN  
    HireDate := og_get_datecell(query, OG_NEWDATA, 'HIREDATE');  
    if HireDate < '28-SEP-81' then  
        og_set_bfcolor(elem, 'green');  
    end if;  
END;
```

OG_Get_Newrows

説明

このファンクションは、問合せに追加されるデータの新規行の数を決定します。

構文

```
FUNCTION OG_Get_Newrows  
    (query OG_Query)  
RETURN NUMBER;
```

パラメータ

<i>query</i>	問合せに対するハンドル。
--------------	--------------

戻り値

前回実行された問合せに追加された行数。

使用上の注意

このファンクションは、問合せプロパティで新規データを旧データに追加するように指定している場合に便利です。新規データが旧データを置換する場合、このファンクションはOG_Numrowsと同じ結果を返します。

OG_Get_Newrowsの例

```
/* Suppose you have a query that appends new data old data, but you want  
**to know a cell value for the first new row returned.The following  
**function sets the query's cursor to start at the first new row returned:  
*/  
  
*/
```

```
FUNCTION example(query OG_Query) RETURN CHAR IS
  total_rows  NUMBER;
  new_rows    NUMBER;
  new_name    VARCHAR2(10);
BEGIN
  OG_Execute_Query(Query);
  total_rows := OG_Numrows(Query, OG_Newdata);
  new_rows := OG_Get_Newrows(Query);
  OG_Start_From(Query, OG_Newdata, total_rows - new_rows);

  new_name := OG_Get_Charcell(Query, 'ENAME');
  RETURN(new_name);
END;
```

OG_Get_Numcell

説明

このファンクションは、指定された列の現在のデータ行の、指定された問合せの数値データ値を返します。

構文

```
FUNCTION OG_Get_Numcell
  (query_hdl  OG_Query,
   col_name   VARCHAR2)
RETURN NUMBER;

FUNCTION OG_Get_Numcell
  (query_hdl  OG_Query,
   which_data NUMBER,
   col_name   VARCHAR2)
RETURN NUMBER;
```

パラメータ

<i>query_hdl</i>	返すデータを含む問合せに対するハンドル。
<i>col_name</i>	返すデータを含む列の名前。
<i>which_data</i>	セル値を旧データまたは新規データのどちらから取り出すかを指定します。指定しなかった場合、この引数値にはデフォルトのOG_Newdataが設定されます。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Newdata セル値を新規データから取り出します。 OG_Olddata セル値を旧データから取り出します。

戻り値

指定されたデータ・セルの内容。

使用上の注意

現在の行は、問合せを実行するためにOG_Execute_Queryを使用するときに、最初に作成される問合せの暗黙的カーソルにより決定されます。

データ表に表示された問合せ結果を表示する場合、このファンクションは現在の行と指定された列の交差するセルに含まれるデータを返します。

OG_Get_Numcellの例

```
/* The following is an example of a
**format trigger that changes the color
**of the chosen chart element (e.g. pie
**slice or bar) to red if its value is
**greater than 200.

PROCEDURE format_point (elem IN og_object, query IN
    og_query) IS
    st_price NUMBER;
BEGIN
    st_price:=OG_GET_NUMCELL (query, OG_NEWDATA, 'sell_pr');
    IF st_price > 200 THEN
        OG_SET_BFCOLOR (elem,'red');
    END IF;
END;
```

OG_Get_Query

説明

このファンクションは、指定された問合せに対するハンドルを返します。

構文

```
FUNCTION OG_Get_Query
    (query_name VARCHAR2)
RETURN OG_Query;
```

パラメータ

query_name	返されるハンドルを持つ問合せの名前。 注意: QUERY_NAMEは大文字と小文字を区別します。
------------	--

戻り値

指定された問合せに対するハンドル。

使用上の注意

問合せが存在しない場合、このファンクションはNULLハンドルを返します。

OG_Get_Queryの例

```
/* Suppose you want to update a chart periodically.
**You could write the following timer trigger:
*/

*/

PROCEDURE every_30_secs IS
  the_query  OG_Query;
  the_chart  OG_Object;
BEGIN
  the_query:=OG_Get_Query('Emp_Query');
  the_chart:=OG_Get_Object('Emp_Chart');
  OG_Execute_Query(The_Query);
  OG_Update_Chart(The_Chart, OG_All_Chupda);
```

OG_Get_Schema

説明

このファンクションは、問合せの特定の列のスキーマに関する情報を返します。

構文

```
FUNCTION OG_Get_Schema
  (query_hdl  OG_Query,
   which_data  NUMBER,
   col_num    NUMBER)
RETURN OG_Colschema;
```

パラメータ

<i>query_hdl</i>	列を含む問合せに対するハンドル。
<i>which_data</i>	スキーマが取り出される列が問合せの旧データまたは新規データのどちらに存在するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Newdata 列が問合せの新規データに存在します。 OG_Olddata 列が問合せの旧データに存在します。
<i>col_num</i>	取り出す列のスキーマを指定します。最初の列の番号は0で、2番目の列の番号は1です。

戻り値

指定された問合せ列のスキーマ。

OG_Get_Schemaの例

```
/* Suppose you want to query a database table, and then use the name
**of the first column elsewhere in your application.Assume you have
**defined a parameter named 'my_query' that is of type CHAR,
**and that you have defined the following SQL query named 'query0':
*/

*/

&my_query

/* The following function takes a table name as an argument and
**returns the name of the table's first column:
*/

FUNCTION get_col_name (table_name IN VARCHAR2) RETURN VARCHAR2 IS
    my_schema    OG_Colschema;
    star_query    OG_Query;
BEGIN
    :my_query:='select * from ' || table_name;
    star_query:=OG_Get_Query('Query0');
    OG_Execute_Query(Star_Query);
    my_schema:=OG_Get_Schema(Star_Query, OG_Newdata, 0);
    RETURN(my_schema.colname);
END;
```

OG_Insert_Column

説明

このプロシージャは、列をカスタム問合せに挿入します。

構文

```
PROCEDURE OG_Insert_Column
    (query_hdl    OG_Query,
     indx         NUMBER,
     schema       OG_Colschema);
```

パラメータ

<i>query_hdl</i>	列を挿入する問合せに対するハンドル。
------------------	--------------------

<i>indx</i>	新規列を問合せ列リストに挿入する索引です。この引数は、0~ <i>n</i> (両端の値を含む)の整数値である必要があります。ここで、 <i>n</i> は挿入より前の問合せ列数です。次のビルトイン定数のいずれかを指定することもできます。 OG_First 問合せ列リストの最初に新規列を挿入します(索引 = 0)。 OG_Last 問合せ列リストの最後に新規列を挿入します(索引 = 挿入より前の問合せ列数)。
<i>schema</i>	挿入する列のスキーマ。

OG_Insert_Columnの例

```
/* The following procedure creates 'query0', containing the columns ENAME
**and SAL:
*/

*/

PROCEDURE example IS
  query  OG_Query;
  col    OG_Colschema;
BEGIN
  query := OG_Make_Query('Query0', NULL);
  OG_Set_Querytype(Query, OG_Custom_Qtype);

  col.colname := 'ENAME';
  col.coltype := OG_Char_Coltype;
  col.maxlen := 10;

  OG_Insert_Column(Query, OG_Last, col);

  col.colname := 'SAL';
  col.coltype := OG_Number_Coltype;
  col.precision := 7;
  col.scale := 2;

  OG_Insert_Column(Query, OG_Last, col);

END;
```

OG_Make_Query

説明

このファンクションは、問合せを作成します。

構文

```
FUNCTION OG_Make_Query
```

```
(querytype    NUMBER,  
 querysource  VARCHAR2  
RETURN  OG_Query;
```

パラメータ

querytype	問合せタイプ。値は、次のビルトイン定数のうちのいずれかを指定してください。 OG_Custom_Qtype 問合せはカスタム問合せです。 OG_Exsql_Qtype 問合せによりSQL SELECT文を含むテキスト・ファイルからデータが取り出されます。 OG_Prn_Qtype PRNファイルに基づいて問合せが行われます。 OG_Sql_Qtype 問合せはSQL SELECT文です。 OG_Sylk_Qtype SYLKファイルに基づいて問合せが行われます。 OG_Wks_Qtype WKSファイルに基づいて問合せが行われます。
querysource	問合せデータのソース。データベースからのデータの場合は、このプロパティに問合せのSQL SELECT文のテキストが含まれている必要があります。データがファイル・システムに格納されている場合は、このプロパティにデータ・ファイルのパスと名前を指定する必要があります。

戻り値

新規に作成された問合せに対するハンドル。

OG_Make_Queryの例

```
/* The following function creates a SQL query:  
*/  
  
*/  
  
FUNCTION example(query_name VARCHAR2) RETURN OG_Query IS  
  query  OG_Query;  
  qtype  NUMBER;  
  qsource VARCHAR2(2000);  
BEGIN  
  qtype := OG_Sql_Qtype;  
  qsource := 'select ename, sal from emp';  
  
  query := OG_Make_Query(qtype, qsource);  
  
  OG_Set_Name(Query, query_name);  
  OG_Execute_Query(Query);  
  RETURN(query);  
END;
```

OG_Next_Row

説明

構文

このプロシージャは、指定された問合せに関連付けられた暗黙的カーソルをデータの次の行に進めます。

```
PROCEDURE OG_Next_Row
  (query_hdl   OG_Query,
   which_data  NUMBER);
```

パラメータ

<i>query_hdl</i>	問合せに対するハンドル。
<i>which_data</i>	旧データまたは新規データのどちらを処理するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Newdata 問合せの新規データのカーソルを進めます。 OG_Olddata 問合せの旧データのカーソルを進めます。

使用上の注意

カーソルが問合せのデータの最後の行を指し示している場合は、OG_Next_Rowの次のコールによってカーソルの位置は変わりません。データが存在しない行にはカーソルは進まないのので、エラーや例外は発生しません。カーソルがデータの最後の行を指し示していることを確認するには、OG_Numrowsを使用して正確な行数を決定してから、カーソルを進めるためにOG_Next_Rowを使用した回数を算出します。

OG_Next_Rowの例

```
/* Suppose you want to name each bar in a bar chart so that when
**the user selects one of the bars you can determine which one it is
**by checking its name.For this example, assume the query for the chart
** is:
*/

*/

SELECT ENAME, SAL FROM EMP

/*The following procedure gives each bar the name of its category,
**which in this case is its associated ENAME:
*/
*/

PROCEDURE name_theBars (my_chart IN OG_Object, my_query IN
  OG_Query) IS
  bar_rec   OG_Chelement_Ca;
```

```
curr_row  NUMBER;
total     NUMBER;
bar_name  VARCHAR2(15);
BEGIN
  OG_Execute_Query(My_Query);
  OG_Start_From(My_Query, OG_Newdata, 0);
  total:=OG_Numrows(My_Query, OG_Newdata);
  FOR curr_row IN 0..total-1 LOOP
    bar_name:=OG_Get_Charcell(My_Query, OG_Newdata,
      'ENAME');
    bar_rec.chelement_cagr.mask:=OG_None_Graphica;
    bar_rec.chelement_cace.mask:=OG_Name_Chelementa;
    bar_rec.chelement_cace.name:=bar_name;
    OG_Set_Attr(My_Chart, curr_row, 'ENAME', bar_rec);
    OG_Next_Row(My_Query, OG_Newdata);
  END LOOP;
  OG_Update_Chart(My_Chart, OG_All_Chupda);
END;
```

OG_Numcols

説明

このファンクションは、問合せに存在する列数を返します。

構文

```
FUNCTION OG_Numcols
  (query_hdl  OG_Query,
   which_data NUMBER)
RETURN NUMBER;
```

パラメータ

query_hdl	問合せに対するハンドル。
which_data	新規データまたは旧データのどちらをチェックするかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Newdata 問合せの新規データの列数を返します。 OG_Olddata 問合せの旧データの列数を返します。

戻り値

指定された問合せ列数。

OG_Numcolsの例

```
/* Suppose Reports will pass data to your display, and you want to chart
** it.
```

```
**Since you may not be sure what columns your display will receive,
**you can make your charting procedure generic.You can write one
**procedure that creates an chart, then pass the query and chart to
**another procedure that inserts the query's columns as fields.The
**following procedure inserts the columns (it assumes the first column
**is the independent field, and the rest are dependent fields):
*/

*/

PROCEDURE add_columns(the_query OG_Query, the_chart OG_Object) IS
    num_of_cols    NUMBER(1);
    the_field      OG_Field;
    the_column     OG_Colschema;
BEGIN
    OG_Execute_Query(The_Query);
    num_of_cols:=OG_Numcols(The_Query, OG_Newdata);
    FOR i IN 0..num_of_cols-1 LOOP
        the_column:=OG_Get_Schema(The_Query, OG_Newdata, i);
        the_field.colname:=the_column.colname;
        IF i=0 THEN
            the_field.field_type:=OG_Independent;
        ELSE
            the_field.field_type:=OG_Dependent;
            the_field.ftname:='line';
        END IF;
        OG_Insert_Field(The_Chart, the_field, i);
    END LOOP;
    OG_Update_Chart(The_Chart, OG_All_Chupda);
END;
```

OG_Numrows

説明

このファンクションは、問合せに存在する行数を返します。

構文

```
FUNCTION OG_Numrows
    (query_hdl    OG_Query,
     which_data   NUMBER)
RETURN NUMBER;
```

パラメータ

query_hdl	問合せに対するハンドル。
-----------	--------------

<i>which_data</i>	新規データまたは旧データのどちらをチェックするかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。
OG_Newdata	問合せの新規データの行数を返します。
OG_Olddata	問合せの旧データの行数を返します。

戻り値

指定された問合せのデータの行数。

OG_Numrowsの例

```

/* Suppose you want to name each bar in a bar chart so that when
**the user selects one of the bars you can determine which one it is
**by checking its name.For this example, assume the query for the chart
** is:
*/

*/

SELECT ENAME, SAL FROM EMP

/*The following procedure gives each bar the name of its category,
**which in this case is its associated ENAME:
*/
*/

PROCEDURE name_the_bars(my_chart IN OG_Object, my_query IN
  OG_Query) IS
  bar_rec    OG_Chelement_Ca;
  curr_row   NUMBER;
  total      NUMBER;
  bar_name   VARCHAR2(15);
BEGIN
  OG_Execute_Query(My_Query);
  OG_Start_From(My_Query, OG_Newdata, 0);
  total:=OG_Numrows(My_Query, OG_Newdata);
  FOR curr_row IN 0..total-1 LOOP
    bar_name:=OG_Get_Charcell(My_Query, OG_Newdata,
      'ENAME');
    bar_rec.chelement_cagr.mask:=OG_None_Graphica;
    bar_rec.chelement_cace.mask:=OG_Name_Chelementa;
    bar_rec.chelement_cace.name:=bar_name;
    OG_Set_Attr(My_Chart, curr_row, 'ENAME', bar_rec);
    OG_Next_Row(My_Query, OG_Newdata);
  END LOOP;
  OG_Update_Chart(My_Chart, OG_All_Chupda);
END;
```

OG_Set_Charcell

説明

このプロシージャは、行バッファの文字列型セル値を設定します。

構文

```
PROCEDURE OG_Set_Charcell
  (query_hdl    OG_Query,
   col_name     VARCHAR2
   cell_value   VARCHAR2);
```

パラメータ

<i>query_hdl</i>	セル値を設定する問合せに対するハンドル。
<i>col_name</i>	設定するセルを含む列の名前。
<i>cell_value</i>	セルが含む値。

使用上の注意

バッファのすべてのセルに値を設定した後、OG_Append_Rowを使用して、カスタム問合せの最後に新規の行としてバッファを追加します。

OG_Set_Charcellの例

```
/* Suppose you want to create a custom query using the ENAME, SAL,
**and HIREDATE columns in the existing query 'query0' as a basis.
**However, in the new query, you want to double every SAL value.
**The following procedure is a custom query procedure you could use:
*/

*/

PROCEDURE OGQUERYPROC0(query IN OG_Query) IS
  other_ename    VARCHAR2(10);
  other_sal      NUMBER(7,2);
  other_query    OG_Query;
  other_hiredate DATE;
  row_count      NUMBER;
BEGIN
  OG_Clear_Query(Query);

  other_query := OG_Get_Query('Query0');
  row_count := OG_Numrows(Other_Query, OG_Newdata);
  OG_Start_From(Other_Query, OG_Newdata, 0);
  FOR i IN 0..row_count-1 loop
```



```

other_ename := OG_Get_Charcell(Other_Query, 'ENAME');
other_sal := OG_Get_Numcell(Other_Query, 'SAL');
other_hiredate := OG_Get_Numcell(Other_Query, 'HIREDATE');
OG_Set_Charcell(Query, 'ENAME', other_ename);
OG_Set_Numcell(Query, 'SAL', other_sal * 2);
OG_Set_Datecell(Query, 'HIREDATE', other_hiredate);
OG_Append_Row(Query);
OG_Next_Row(Other_Query, OG_Newdata);
END LOOP;

END;
```

OG_Set_Datecell

説明

このプロシージャは、行バッファの日付型セル値を設定します。

構文

```

PROCEDURE OG_Set_Datecell
  (query_hdl   OG_Query,
   col_name    VARCHAR2
   cell_value  DATE);
```

パラメータ

<i>query_hdl</i>	セル値を設定する問合せに対するハンドル。
<i>col_name</i>	設定するセルを含む列の名前。
<i>cell_value</i>	セルが含む値。

使用上の注意

バッファのすべてのセルに値を設定した後、OG_Append_Rowを使用して、カスタム問合せの最後に新規の行としてバッファを追加します。

OG_Set_Datecellの例

```

/* Suppose you want to create a custom query using the ENAME, SAL,
**and HIREDATE columns in the existing query 'query0' as a basis.
**However, in the new query, you want to double every SAL value.
**The following procedure is a custom query procedure you could use:
*/

*/
```

```
PROCEDURE OGQUERYPROC0(query IN OG_Query) IS
  other_ename    VARCHAR2(10);
  other_sal      NUMBER(7,2);
  other_query    OG_Query;
  other_hiredate DATE;
  row_count      NUMBER;
BEGIN
  OG_Clear_Query(Query);

  other_query := OG_Get_Query('Query0');
  row_count := OG_Numrows(Other_Query, OG_Newdata);
  OG_Start_From(Other_Query, OG_Newdata, 0);
  FOR i IN 0..row_count-1 loop
    other_ename := OG_Get_Charcell(Other_Query, 'ENAME');
    other_sal := OG_Get_Numcell(Other_Query, 'SAL');
    other_hiredate := OG_Get_Numcell(Other_Query, 'HIREDATE');
    OG_Set_Charcell(Query, 'ENAME', other_ename);
    OG_Set_Numcell(Query, 'SAL', other_sal * 2);
    OG_Set_Datecell(Query, 'HIREDATE', other_hiredate);
    OG_Append_Row(Query);
    OG_Next_Row(Other_Query, OG_Newdata);
  END LOOP;

END;
```

OG_Set_Numcell

説明

このプロシージャは、行バッファの数値型セル値を設定します。

構文

```
PROCEDURE OG_Set_Numcell
  (query_hdl    OG_Query,
   col_name     VARCHAR2,
   cell_value   NUMBER);
```

パラメータ

<i>query_hdl</i>	セル値を設定する問合せに対するハンドル。
<i>col_name</i>	設定するセルを含む列の名前。
<i>cell_value</i>	セルが含む値。

使用上の注意

バッファのすべてのセルに値を設定した後、OG_Append_Rowを使用して、カスタム問合せの最後に新規の行としてバッファを追加します。

OG_Set_Numcellの例

```
/* Suppose you want to create a custom query using the ENAME, SAL,
**and HIREDATE columns in the existing query 'query0' as a basis.
**However, in the new query, you want to double every SAL value.
**The following procedure is a custom query procedure you could use:
*/
```

```
*/
```

```
PROCEDURE OGQUERYPROC0(query IN OG_Query) IS
  other_ename    VARCHAR2(10);
  other_sal      NUMBER(7,2);
  other_query    OG_Query;
  other_hiredate DATE;
  row_count      NUMBER;
BEGIN
  OG_Clear_Query(Query);

  other_query := OG_Get_Query('Query0');
  row_count := OG_Numrows(Other_Query, OG_Newdata);
  OG_Start_From(Other_Query, OG_Newdata, 0);
  FOR i IN 0..row_count-1 loop
    other_ename := OG_Get_Charcell(Other_Query, 'ENAME');
    other_sal := OG_Get_Numcell(Other_Query, 'SAL');
    other_hiredate := OG_Get_Numcell(Other_Query, 'HIREDATE');
    OG_Set_Charcell(Query, 'ENAME', other_ename);
    OG_Set_Numcell(Query, 'SAL', other_sal * 2);
    OG_Set_Datecell(Query, 'HIREDATE', other_hiredate);
    OG_Append_Row(Query);
    OG_Next_Row(Other_Query, OG_Newdata);
  END LOOP;

END;
```

OG_Set_Schema

説明

このプロシージャは、カスタム問合せ列のスキーマを設定します。

構文

```
PROCEDURE OG_Set_Schema
  (query_hdl   OG_Query,
   col_num     NUMBER,
   schema      OG_Colschema);
```

パラメータ

<i>query_hdl</i>	設定するスキーマを持つ問合せに対するハンドル。
<i>col_num</i>	設定する列の索引。
<i>schema</i>	列を設定する新規スキーマ。

OG_Set_Schemaの例

```
/* The following procedure changes the name of the fist column
**in a custom query from ENAME to EMPLOYEE:
*/

*/

PROCEDURE example(query OG_Query) IS
  schema OG_Colschema;
BEGIN
  schema.colname := 'EMPLOYEE';
  schema.coltype := OG_Char_Coltype;
  schema.maxlen := 10;

  OG_Set_Schema(Query, 0, schema);

END;
```

OG_Start_From

説明

このプロシージャは、指定された問合せに関連付けられた暗黙的カーソルが指定されたデータ行を指し示すように設定します。

構文

```
PROCEDURE OG_Start_From
  (query_hdl   OG_Query,
   which_data   NUMBER,
   start_row    NUMBER);
```

パラメータ

<i>query_hdl</i>	問合せに対するハンドル。
------------------	--------------

<i>which_data</i>	旧データまたは新規データのどちらを処理するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Newdata 問合せの新規データのカーソルを設定します。 OG_Olddata 問合せの旧データのカーソルを設定します。
<i>start_row</i>	カーソルが位置する行番号。

使用上の注意

データの一番最初の行を指し示すには、オフセット値0を使用します。データの最後の行は、OG_Numrowsが返した値から1を引いたオフセット値になります。OG_Execute_Queryを使用して問合せを実行するたびにカーソル位置は廃棄されるので、設定し直す必要があることに注意してください。問合せを実行するたびに、問合せにより取り出される行数は変わる可能性があることに注意してください。

OG_Start_Fromの例

```

/* Suppose you want to name each bar in a bar chart so that
**when the user selects one of the bars you can determine
**which one it is by checking its name.For this example,
**assume the query for the chart is:
*/

*/

SELECT ENAME, SAL FROM EMP

/*The following procedure gives each bar the name of its
**category, which in this case is its associated ENAME:
*/
*/

PROCEDURE name_the_bars (my_chart IN OG_Object, my_query IN
OG_Query) IS
    bar_rec    OG_Chelement_Ca;
    curr_row   NUMBER;
    total      NUMBER;
    bar_name   VARCHAR2(15);
BEGIN
    OG_Execute_Query(My_Query);
    OG_Start_From(My_Query, OG_Newdata, 0);
    total:=OG_Numrows(My_Query, OG_Newdata);
    FOR curr_row IN 0..total-1 LOOP
        bar_name:=OG_Get_Charcell(My_Query, OG_Newdata,
            'ENAME');
        bar_rec.chelement_cagr.mask:=OG_None_Graphica;
        bar_rec.chelement_cace.mask:=OG_Name_Chelementa;
    
```

```
        bar_rec.chelement_cace.name:=bar_name;
        OG_Set_Attr(My_Chart, curr_row, 'ENAME', bar_rec);
        OG_Next_Row(My_Query, OG_Newdata);
    END LOOP;
    OG_Update_Chart(My_Chart, OG_All_Chupda);
END;
```

サウンド・ビルトイン

OG_Destroy (サウンド)

OG_Export_Sound

OG_Get_Sound

OG_Import_Sound

OG_Make_Sound

OG_Play_Sound

OG_Record_Sound

OG_Stop_Sound

OG_Destroy (サウンド)

説明

このプロシージャで、指定されているサウンドを破棄します。

構文

```
PROCEDURE OG_Destroy  
  (sound_hdl OG_Sound);
```

パラメータ

<i>sound_hdl</i>	破棄するサウンドのハンドルです。
------------------	------------------

OG_Destroy (サウンド) の例

```
/* The following procedure destroys the specified sound:  
*/  
  
*/  
  
PROCEDURE destroy_sound(sound_name VARCHAR2) IS  
  sound OG_Sound;  
BEGIN  
  sound := OG_Get_Sound(Sound_Name);  
  OG_Destroy(Sound);  
END;
```

OG_Export_Sound

説明

このプロシージャで、サウンドをエクスポートします。

構文

```
PROCEDURE OG_Export_Sound
  (name          VARCHAR2,
   repository    NUMBER,
   format        NUMBER,
   sound_hdl     OG_Sound);
```

パラメータ

<i>name</i>	サウンドのエクスポート先の名前。サウンドがデータベースに格納される場合、この引数には、サウンド・モジュール名のみを指定してください。サウンドがファイル・システムに格納される場合は、この引数には、サウンド・ファイルの絶対パス名が相対パス名を指定してください。
<i>repository</i>	サウンドを、ファイル・システムとデータベースのどちらに格納するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db サウンドはデータベースに格納されます。 OG_FileSystem サウンドはファイル・システムに格納されます。
<i>format</i>	サウンドのエクスポート形式を指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Aiff_Sformat サウンドがAIFF形式で保存されていることを示します。 OG_Aiffc_Sformat サウンドがオーディオ交換ファイル形式-cで保存されていることを示します。 OG_Au_Sformat サウンドが、SUN au形式で保存されていることを示します。 OG_Wave_Sformat サウンドが、PCM WAVE形式で保存されていることを示します。
<i>sound_hdl</i>	エクスポートされるサウンドのハンドルです。

OG_Export_Soundの例

```
/* Suppose you want to export the sound named 'sound0' to the AIFF-c file
** 'my_sound' so that you can later import it into some other application.
** The following procedure does this:
*/

*/

PROCEDURE export_the_sound IS
  the_sound  OG_Sound;
BEGIN
  the_sound:=OG_Get_Sound('Sound0');
```



```
OG_Export_Sound('My_Sound', OG_FileSystem,  
OG_Aiffc_Sformat, the_sound);  
END;
```

OG_Get_Sound

説明

この関数は、指定されたサウンドのハンドルを戻します。

構文

```
FUNCTION OG_Get_Sound  
    (sound_name VARCHAR2)  
RETURN OG_Sound;
```

パラメータ

<i>sound_name</i>	戻るサウンドのハンドル名。
-------------------	---------------

戻り値

指定されたサウンドのハンドル。指定のサウンドが存在しない場合は、NULLハンドルが戻ります。

OG_Get_Soundの例

```
/* Suppose you want to play a warning sound, to indicate  
**low inventory or an illegal action by the user.  
*/
```

```
PROCEDURE warning IS  
    the_sound OG_Sound;  
BEGIN  
    the_sound:=OG_Get_Sound('Warning_Snd');  
    OG_Play_Sound(the_sound);  
END;
```

OG_Import_Sound

説明

この関数は、データベースまたはファイルからサウンドをインポートします。

構文

```
FUNCTION OG_Import_Sound
  (name          VARCHAR2,
   repository    NUMBER,
   format        NUMBER,
   sound_name    VARCHAR2)
RETURN OG_Sound;
```

パラメータ

<i>name</i>	格納されているサウンドの名前。サウンドがデータベースに格納されている場合、この引数にはサウンド名のみを指定してください。サウンドがファイル・システムに格納されている場合、この引数には、サウンド・ファイルの絶対パス名か相対パス名を指定してください。
<i>repository</i>	サウンドが、ファイル・システムとデータベースのどちらに格納されているかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db サウンドがデータベースに格納されていることを示します。 OG_FileSystem サウンドがファイル・システムに格納されていることを示します。
<i>format</i>	サウンドの保存形式を指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Aiff_Sformat サウンドがAIFF形式で保存されていることを示します。 OG_Aifc_Sformat サウンドがAIFF-c形式で保存されていることを示します。 OG_Any_Sformat Graphics Builderが自動的にサウンドの形式を決定することを示します。 注意: 現在は使用されていないOracleフォーマットでサウンドがエクスポートされた場合は、この形式を指定してください。 OG_Au_Sformat サウンドが、SUN au形式で保存されていることを示します。 OG_Oracle_Sformat サウンドが、他のOracle製品で使用されているOracleフォーマットで保存されていることを示します。 OG_Wave_Sformat サウンドがWAV形式で保存されていることを示します。
<i>sound_name</i>	Graphics Builderでサウンドに割り当てられる名前。この名前をもつ別のサウンドがすでに存在している場合、Graphics Builderでは、そのサウンドはインポート・サウンドに置換されます。

戻り値

インポートされたサウンドのハンドル。

OG_Import_Soundの例

```
/* Suppose you want to import the contents of the AIFF-c file
** 'my_sound' into your display as the sound named 'sound0'.
** The following procedure does this:
```

```
*/

*/

PROCEDURE import_the_sound IS
    the_sound  OG_Sound;
BEGIN
    the_sound:=OG_Import_Sound('My_Sound', OG_Filesystem,
    OG_Aiffc_Sformat, 'sound0');
END;
```

OG_Make_Sound

説明

この関数は、データベース表に格納されているデータからサウンドを作成します。

構文

```
FUNCTION OG_Make_Sound
    (query      OG_Query,
     which_data  NUMBER,
     colname     VARCHAR2)
RETURN OG_Sound;
```

パラメータ

<i>query</i>	データベースの表からサウンドを検索する問合せのハンドル。なお、この表はユーザーの表であることが必要です。データベースにモジュールを保存またはエクスポートするときにGraphics Builderによって使用されるプライベート表は指定できません。
<i>which_data</i>	作成されるサウンドが、問合せの新規データ・セット内に含まれているか、旧データ・セットに含まれているかを指定します。Graphics Builderには、この属性の値として使用可能なビルトイン数値定数が2つあります。 OG_Newdata サウンドが問合せの新規データ・セットに含まれていることを示します。 OG_Olddata サウンドが問合せの旧データ・セットに含まれていることを示します。
<i>colname</i>	サウンド・データが入る問合せ列の名前。作成されるサウンドは、この属性により指定される列と問合せのカーソルが指している行の交差位置の問合せセルに含まれているものです。

戻り値

新規に作成されたサウンドのハンドル。

OG_Make_Soundの例

```
/* The following function creates a sound from data in the sixth
**row of the query 'sound_query' in the column SOUND_COLUMN:
*/

*/

FUNCTION example(sound_name VARCHAR2) RETURN OG_Sound IS
  query  OG_Query;
  sound  OG_Sound;
BEGIN
  query := OG_Get_Query('Sound_Query');
  OG_Execute_Query(Query);
  OG_Start_From(Query, OG_Newdata, 5);
  sound := OG_Make_Sound(Query, OG_Newdata, 'SOUND_COLUMN');

  OG_Set_Name(Sound, sound_name);
  RETURN(sound);
END;
```

OG_Play_Sound

説明

このプロシージャで、ユーザーが指定したサウンド出力デバイスを使用して、指定のサウンドを再生します。

構文

```
PROCEDURE OG_Play_Sound
  (sound_hdl OG_Sound);
```

パラメータ

<i>sound_hdl</i>	再生されるサウンドのハンドル。
------------------	-----------------

OG_Play_Soundの例

```
/* Suppose you want to play a warning sound, to indicate
**low inventory or an illegal action by the user.
*/

PROCEDURE warning IS
  the_sound  OG_Sound;
BEGIN
  the_sound:=OG_Get_Sound('Warning_Snd');
  OG_Play_Sound(The_Sound);
```

```
END;
```

OG_Record_Sound

説明

このプロシージャで、サウンド・ダイアログ・ボックスの表示と新規サウンドの録音ができます。

構文

```
PROCEDURE OG_Record_Sound (sound_hdl OG_Sound);
```

パラメータ

<i>sound_hdl</i>	サウンドのハンドル。
------------------	------------

使用上の注意

新規サウンドは、*sound_hdl*で指したサウンドに上書きされます。また、*sound_hdl*が指すサウンドは、Builderまたはビルトイン・ファンクションOG_Import_SoundおよびOG_Makeで以前作成されたものです。

OG_Record_Soundの例

```
/* Suppose you want the user to record a sound to be played as a warning
**when data changes.You could write the following procedure:
*/

*/

PROCEDURE record_warning IS
    warn_sound OG_Sound;
BEGIN
    warn_sound:=OG_Get_Sound('Warning');
    IF not OG_Isnull(Warn_Sound) THEN
        OG_Record_Sound(Warn_Sound);
    END IF;
END;
```

OG_Stop_Sound

説明

このプロシージャで、再生処理中のサウンドの再生を取り消します。

構文

```
PROCEDURE OG_Stop_Sound  
    (sound_hdl   OG_Sound);
```

パラメータ

<i>sound_hdl</i>	再生を停止するサウンドのハンドル。
------------------	-------------------

OG_Stop_Soundの例

```
/* Suppose you want to create a button that the user  
**can select to cancel a sound that is currently playing.  
**The following button procedure does this:  
*/  
  
*/  
  
PROCEDURE OGBUTTONPROC0 (buttonobj IN OG_Object, hitobj IN OG_Object, win  
IN OG_Window, eventinfo IN OG_Event) IS  
    sound   OG_Sound;  
BEGIN  
    sound := OG_Get_Sound('Alarm');  
    OG_Stop_Sound(sound);  
END;
```

テンプレート・ビルトイン

- OG_Clone (テンプレート)
- OG_Delete_Ftemp
- OG_Delete_Refline
- OG_Destroy (テンプレート)
- OG_Export_Template
- OG_Get_Axis
- OG_Get_Ftemp
- OG_Get_Refline
- OG_Get_Template
- OG_Import_Template
- OG_Insert_Ftemp
- OG_Insert_Refline
- OG_Make_Template

OG_Clone (テンプレート)

説明

このファンクションは指定したテンプレートと一致する、新規チャート・テンプレートを作成します。

構文

```
FUNCTION OG_Clone
  (template_hdl OG_Template,
   name          VARCHAR2)
RETURN OG_Template;
```

パラメータ

<i>template_hdl</i>	ひな型となるチャート・テンプレートのハンドル。
<i>name</i>	新規テンプレートに付ける名前。

戻り値

新規に作成されたテンプレートのハンドル。

OG_Clone (テンプレート) の例

```
/* Suppose you have created a template, and you want to create another
** identical template without having to again specify the same properties.
*/

PROCEDURE dup_template (old_template IN OG_Template) IS
    new_template    OG_Template;
BEGIN
    new_template:=OG_Clone (Old_Template);
END;
```

OG_Delete_Ftemp

説明

このプロシージャで、指定されたチャート・テンプレートから1つ以上のフィールド・テンプレートを削除します。

構文

```
PROCEDURE OG_Delete_Ftemp
(
    template_hdl    OG_Template,
    indx            NUMBER,
    total            NUMBER);
```

パラメータ

<i>template_hdl</i>	チャート・テンプレートのハンドル。
<i>indx</i>	削除する最初のフィールド・テンプレートの索引。
<i>total</i>	削除するフィールド・テンプレートの合計数。

OG_Delete_Ftempの例

```
/* The following procedure deletes a column from the template 'template0':
*/

*/

PROCEDURE example (ft_num NUMBER) IS
```



```
template OG_Template;
BEGIN
  template := OG_Get_Template('Template0');
  OG_Delete_Ftemp(Template, ft_num, 1);
END;
```

OG_Delete_Refline

説明

このプロシージャで、指定されたチャート・テンプレートから1つ以上の参照線を削除します。

構文

```
PROCEDURE OG_Delete_Refline
  (template_hdl OG_Template,
   indx         NUMBER,
   total        NUMBER);
```

パラメータ

<i>template_hdl</i>	チャート・テンプレートのハンドル。
<i>indx</i>	削除する最初の参照線の索引。
<i>total</i>	削除する参照線の合計数。

OG_Delete_Reflineの例

```
/* The following procedure deletes a reference line template 'template0':
*/

*/

PROCEDURE example(rl_num NUMBER) IS
  template OG_Template;
BEGIN
  template := OG_Get_Template('Template0');
  OG_Delete_Refline(Template, rl_num, 1);
END;
```

OG_Destroy (テンプレート)

説明

このプロシージャで、指定されたチャート・テンプレートを破棄します。

構文

```
PROCEDURE OG_Destroy
  (template_hdl  OG_Template);
```

パラメータ

<i>template_hdl</i>	破棄するチャート・テンプレートのハンドル。
---------------------	-----------------------

OG_Destroy (テンプレート) の例

```
/* The following procedure destroys the specified template:
*/

*/

PROCEDURE destroy_template(template_name VARCHAR2) IS
  template  OG_Template;
BEGIN
  template := OG_Get_Template(Template_Name);
  OG_Destroy(Template);
END;
```

OG_Export_Template

説明

このプロシージャで、チャート・テンプレートをエクスポートします。

構文

```
PROCEDURE OG_Export_Template
  (name          VARCHAR2,
   repository    NUMBER,
   template_hdl  OG_Template);
```

パラメータ

<i>name</i>	テンプレートのエクスポート先の名前。テンプレートがデータベースに格納される場合、この引数には、テンプレート名のみを指定してください。テンプレートがファイル・システムに格納される場合は、テンプレート・ファイルの絶対パス名または相対パス名を指定してください。
-------------	---

<i>repository</i>	テンプレートをファイル・システムとデータベースのどちらに格納するかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_Db テンプレートはデータベースに格納されます。 OG_Filessystem テンプレートはファイル・システムに格納されます。
<i>template_hdl</i>	エクスポートされるテンプレートのハンドル。

OG_Export_Templateの例

```
/* Suppose you want to export the chart template named 'template0'
**to the file 'my_temp' so that you can later import it into some other
**Graphics Builder display.The following procedure does this:
*/

*/

PROCEDURE export_the_template IS
    the_temp  OG_Template;
BEGIN
    the_temp:=OG_Get_Template('Template0');
    OG_Export_Template('My_Temp', OG_Filessystem, the_temp);
END;
```

OG_Get_Axis

説明

このファンクションは、チャート・テンプレートにある軸のハンドルを戻します。

構文

```
FUNCTION OG_Get_Axis
    (template_hdl  OG_Template,
     which_axis    NUMBER)
RETURN OG_Axis;
```

パラメータ

<i>template_hdl</i>	ハンドルが戻される軸を含んでいるチャート・テンプレートのハンドル。
<i>which_axis</i>	どの軸が戻されるかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 OG_X_Axis OG_Y1_Axis OG_Y2_Axis

戻り値

指定された軸のハンドル。指定されたボタン・プロシージャが存在しない場合はNULLが返されます。

OG_Get_Axisの例

```
/* The following function returns a handle to the specified template's
X axis:
*/

*/

FUNCTION example(template_name VARCHAR2) RETURN OG_Axis IS
    template OG_Template;
    axis      OG_Axis;
BEGIN
    template := OG_Get_Template(Template_Name);
    axis := OG_Get_Axis(Template, OG_X_Axis);
    RETURN(axis);
END;
```

OG_Get_Ftemp

説明

このファンクションは、チャート・テンプレート内のフィールド・テンプレートのハンドルを戻します。

構文

```
FUNCTION OG_Get_Ftemp
    (template_hdl OG_Template,
     indx         NUMBER)
RETURN OG_Ftemp;
```

パラメータ

<i>template_hdl</i>	ハンドルが戻されるフィールド・テンプレートを含んでいるチャート・テンプレートのハンドル。
<i>indx</i>	チャートのフィールド・テンプレート・リストの中で戻されるフィールド・テンプレートの索引。

戻り値

指定されたフィールド・テンプレートのハンドル。

OG_Get_Ftempの例

```
/* The following function returns the handle to the
**first field template in the specified chart template:
*/

*/

FUNCTION example(temp_name VARCHAR2) RETURN OG_Ftemp IS
  template OG_Template;
  ftemp    OG_Ftemp;
BEGIN
  template := OG_Get_Template(Temp_Name);
  ftemp := OG_Get_Ftemp(Template, 0);
  RETURN(ftemp);
END;
```

OG_Get_Refline

説明

このファンクションは、チャート・テンプレート内の参照線のハンドルを戻します。

構文

```
FUNCTION OG_Get_Refline
  (template_hdl OG_Template,
   indx         NUMBER)
RETURN OG_Refline;
```

パラメータ

<i>template_hdl</i>	戻される参照線が含まれているチャート・テンプレートのハンドル。
<i>indx</i>	戻されるチャートの参照線リストの中で戻される参照線の索引。

戻り値

指定された参照線のハンドル。

OG_Get_Reflineの例

```
/* The following function returns the handle to the
**first reference line in the specified chart template:
*/

*/

FUNCTION example(temp_name VARCHAR2) RETURN OG_Refline IS
  template OG_Template;
  refline OG_Refline;
BEGIN
  template := OG_Get_Template(Temp_Name);
  refline := OG_Get_Refline(Template, 0);
  RETURN(refline);
END;
```

OG_Get_Template

説明

このファンクションは、指定されたテンプレートのハンドルを戻します。

構文

```
FUNCTION OG_Get_Template
  (template_name VARCHAR2)
RETURN OG_Template;
```

パラメータ

<i>template_name</i>	ハンドルが戻されるチャート・テンプレートの名前。
----------------------	--------------------------

戻り値

指定されたチャート・テンプレートのハンドル。そのテンプレートが存在しない場合は、NULL
ハンドルが戻ります。

OG_Get_Templateの例

```
/* Suppose you want to create a chart programmatically. You would need to
assign attribute values (including a template) to a chart combined attribute
record, then pass that record to OG_Make.
```

```
*/

PROCEDURE create_emp_chart IS
    chart_rec      OG_Chart_Ca;
    the_template   OG_Template;
    the_query      OG_Query;
    the_chart      OG_Object;
BEGIN
    the_template:=OG_Get_Template('Emp_Template');
    the_query:=OG_Get_Query('Emp_Query');
    chart_rec.chart_caoc.template:=the_template;
    chart_rec.chart_caoc.query:=the_query;
    chart_rec.chart_caob.mask:=OG_None_Generica;
    chart_rec.chart_caog.mask:=OG_None_Groupa;
    chart_rec.chart_caoc.mask:=OG_Template_Charta+
                                OG_Query_Charta;
    the_chart:=OG_Make (Chart_Rec) ;
END;
```

OG_Import_Template

説明

このファンクションは、チャート・テンプレートをインポートします。

構文

```
FUNCTION OG_Import_Template
    (name          VARCHAR2,
     repository    NUMBER,
     template_name VARCHAR2)
RETURN OG_Template;
```

パラメータ

<i>name</i>	テンプレートの格納時の名前。テンプレートがデータベースに格納されている場合、この引数には、テンプレート名のみを指定してください。テンプレートがファイル・システムに格納されている場合は、テンプレート・ファイルの絶対パス名または相対パス名を指定してください。	
<i>repository</i>	テンプレートがファイル・システムとデータベースのどちらに格納されているかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。	
	OG_Db	テンプレートがデータベースに格納されていることを示します。
	OG_Filessystem	テンプレートがファイル・システムに格納されていることを示します。

<i>template_name</i>	Graphics Builderでテンプレートに割り当てられる名前。この名前のテンプレートがすでに存在している場合、Graphics Builderでは、そのテンプレートはインポートされたテンプレートに置換されます。
----------------------	--

戻り値

インポートされたテンプレートのハンドル。

OG_Import_Templateの例

```
/* Suppose you want to import the chart template file 'my_temp'
**into your display as the template named 'template0'.
** The following procedure does this:
*/

*/

PROCEDURE import_the_template IS
    the_temp    OG_Template;
BEGIN
    the_temp:=OG_Import_Template('My_Temp', OG_Fileystem,
    'template0');
END;
```

OG_Insert_Ftemp

説明

このプロシージャで、新規フィールド・テンプレートを指定されたチャート・テンプレートに挿入します。

構文

```
PROCEDURE OG_Insert_Ftemp                                pie/table chart
(template_hdl    OG_Template,
attr            OG_Ftemp_Attr,
indx           NUMBER);

PROCEDURE OG_Insert_Ftemp                                axis chart
(template_hdl    OG_Template,
attr            OG_Axisftemp_Ca,
indx           NUMBER);
```


パラメータ

<i>template_hdl</i>	チャート・テンプレートのハンドル。
<i>attr</i>	フィールド・テンプレートの属性を含むレコード。
<i>indx</i>	新規フィールド・テンプレートをチャート・テンプレートに挿入するときの索引。この引数は、0 ~ <i>n</i> の整数(0, <i>n</i> を含む)を指定してください。 <i>n</i> は、チャート・テンプレート内の挿入前のフィールド・テンプレート数です。次のビルトイン定数のいずれかを指定することもできます。 OG_First 新規フィールド・テンプレートを、チャート・テンプレートの先頭に挿入することを示します (index = 0)。 OG_Last 新規フィールド・テンプレートをチャート・テンプレートの最後に挿入することを示します(index = チャート・テンプレート内の挿入前のフィールド・テンプレート数)。

OG_Insert_Ftempの例

```
/* The following procedure inserts a new field
**template into the specified chart template.
*/

PROCEDURE example(template OG_Template) IS
  attr OG_Axisftemp_Ca;
BEGIN
  attr.ca_ftime.plotttype := OG_Bar_Plotttype;
  attr.ca_ftemp.name := 'column';
  attr.ca_ftime.mask:= OG_Plotttype_Axisftempa;
  attr.ca_ftemp.mask := OG_Name_Ftempa;

  OG_Insert_Ftemp(Template, attr, 0);
END;
```

OG_Insert_Refline

説明

このプロシージャで、新規の参照線を指定されたチャート・テンプレートに挿入します。

構文

```
PROCEDURE OG_Insert_Refline
  (template_hdl OG_Template,
  attr          OG_Refline_Attr,
  indx          NUMBER);
```

パラメータ

<i>template_hdl</i>	チャート・テンプレートのハンドル。
<i>attr</i>	参照線の属性を含んでいるレコード。
<i>indx</i>	<p>新規の参照線をチャート・テンプレートに挿入するときの索引。この引数は、0 ~ <i>n</i>の整数(0、<i>n</i>を含む)を指定してください。<i>n</i> は、チャート・テンプレート内の挿入前の参照線の数です。次のビルトイン定数のいずれかを指定することもできます。</p> <p>OG_First 新規の参照線を、チャート・テンプレートの先頭(index = 0)に挿入することを示します。</p> <p>OG_Last 新規の参照線をチャート・テンプレートの最後に挿入することを示します(index = チャート・テンプレート内の挿入前の参照線の数)。</p>

OG_Insert_Reflineの例

```
/* The following procedure inserts a new refernce
**line into the specified chart template.
*/

PROCEDURE example(template OG_Template) IS
    attr OG_Refline_Attr;
BEGIN
    attr.numvalue := 1000;
    attr.label := 'Average';
    attr.mask:= OG_Value_Reflinea+
                OG_Label_Reflinea;

    OG_Insert_Refline(Template, attr, 0);
END;
```

OG_Make_Template

説明

このファンクションはチャート・テンプレートを作成します。

構文

```
FUNCTION OG_Make_Template
  (name          VARCHAR2,
   chart_type    NUMBER
RETURN  OG_Template;
```

パラメータ

<i>name</i>	テンプレート名。
-------------	----------

<i>chart_type</i>	テンプレートのチャート型。値は、次のビルトイン定数のうちのいずれかを指定してください。 OG_Column OG_Column_Stacked OG_Column_Overlap OG_Column_Pct OG_Column_Zero OG_Column_Shadow OG_Column_3d OG_Column_Line OG_Bar OG_Bar_Stacked OG_Bar_Overlap OG_Bar_Pct OG_Bar_Zero OG_Bar_Shadow OG_Bar_3d OG_Bar_Line OG_Line OG_Line_Symbol OG_Line_Stacked OG_Line_Fill OG_Step OG_Step_Symbol OG_Step_Stacked OG_Step_Fill OG_Spline OG_Spline_Symbol OG_Spline_Stacked OG_Spline_Fill OG_Mixed_Line OG_Mixed_Fill OG_Mixed_Spline OG_Mixed_Spfill OG_Doubley_Column OG_Doubley_Overlap OG_Doubley_Line OG_Doubley_Symbol OG_Highlow_Symbol OG_Highlow_Spike OG_Highlow_Both OG_Highlow_Line
-------------------	---

<i>chart_type</i> (続き)	OG_Highlow_Fill
	OG_Scatter_Symbol
	OG_Scatter_Curvefit
	OG_Scatter_Linear
	OG_Scatter_Log
	OG_Scatter_Loglog
	OG_Scatter_Connect
	OG_Gantt
	OG_Gantt_Shadow
	OG_Gantt_Depth
	OG_Pie
	OG_Pie_Shadow
	OG_Pie_Depth OG_Pie_Depth
	OG_Table
	OG_Table_Shadow
	OG_Table_Depth

戻り値

新規に作成されたテンプレートのハンドル。

OG_Make_Templateの例

```
/* The following function creates a column
**chart template with shadows on the bars:
*/

*/

FUNCTION example RETURN OG_Template IS
  template OG_Template;
BEGIN
  template := OG_Make_Template('Template0', OG_Column_Shadow);
  RETURN(template);
END;
```

タイマー・ビルトイン

- OG_Activate_Timer
- OG_Deactivate_Timer
- OG_Destroy (タイマー)
- OG_Get_Timer
- OG_Make_Timer

OG_Activate_Timer

説明

このプロシージャで、指定されたタイマーをアクティブにします。

構文

```
PROCEDURE OG_Activate_Timer
  (timer_hdl  OG_Timer);
```

パラメータ

timer_hdl	アクティブにするタイマーのハンドル。
-----------	--------------------

使用上の注意

図表がオープンされると、デフォルトですべてのタイマーがアクティブになります。
OG_Deactivate_Timerプロシージャを使用して明示的にタイマーを非アクティブにしない限りは、
タイマーをアクティブにする必要はありません。

OG_Activate_Timerの例

```
/* Suppose you have created timers to update all of your charts
**every 30 seconds, and that you have deactivated the timers
**for charts that are not displayed.When you display a chart,
**however, you want to re-activate its timer.
*/

PROCEDURE activate_emp_timer IS
  my_timer  OG_Timer;
BEGIN
```

```
my_timer:=OG_Get_Timer('Emp_Timer');  
OG_Activate_Timer(My_Timer);  
END;
```

OG_Deactivate_Timer

説明

このプロシージャで、指定されたタイマーを非アクティブにします。

構文

```
PROCEDURE OG_Deactivate_Timer  
  (timer_hdl OG_Timer);
```

パラメータ

<i>timer_hdl</i>	タイマーのハンドル。
------------------	------------

使用上の注意

図表をオープンすると、すべてのタイマーは自動的にアクティブになります。タイマーを非アクティブにするには、このプロシージャを実行して明示的に非アクティブにする必要があります。図表の表示されている部分でタイマーが必要なければ、非アクティブにしたほうが効率的です。システムで不要なタイマーの処理に時間を費やす必要がなくなります。

OG_Deactivate_Timerの例

```
/* Suppose you have created timers to update all of your  
**charts every 30 seconds, and that you have deactivated  
**the timers for charts that are not displayed.  
*/
```

```
PROCEDURE deactivate_emp_timer IS  
  my_timer OG_Timer;  
BEGIN  
  my_timer:=OG_Get_Timer('Emp_Timer');  
  OG_Deactivate_Timer(My_Timer);  
END;
```

OG_Destroy (タイマー)

説明

このプロシージャで、指定されたタイマーを破棄します。

構文

```
PROCEDURE OG_Destroy  
    (timer_hdl OG_Timer);
```

パラメータ

<i>timer_hdl</i>	破棄するタイマーのハンドル。
------------------	----------------

OG_Destroy (タイマー) の例

```
/* The following procedure destroys the specified timer:  
*/  
  
*/  
  
PROCEDURE destroy_timer(timer_name VARCHAR2) IS  
    timer OG_Timer;  
BEGIN  
    timer := OG_Get_Timer(Timer_Name);  
    OG_Destroy(timer);  
END;
```

OG_Get_Timer

説明

このファンクションは、指定されたタイマーのハンドルを戻します。

構文

```
FUNCTION OG_Get_Timer  
    (timer_name VARCHAR2)  
RETURN OG_Timer;
```

パラメータ

<i>timer_name</i>	ハンドルが戻されるタイマーの名前。
-------------------	-------------------

戻り値

指定されたタイマーのハンドル。指定されたタイマーが存在しない場合、このファンクションは NULLハンドルを戻します。

OG_Get_Timerの例

```
/* Suppose you have created timers to update all of your
**charts every 30 seconds, and that you have deactivated
**the timers for charts that are not displayed.When you
**display a chart, however, you want to re-activate its timer.
*/

PROCEDURE activate_emp_timer IS
    my_timer    OG_Timer;
BEGIN
    my_timer:=OG_Get_Timer('Emp_Timer');
    OG_Activate_Timer(My_Timer);
END;
```

OG_Make_Timer

説明

このファンクションはタイマーを作成します。

構文

```
FUNCTION OG_Make_Timer
    (interval    OG_Point,
     timerproc   VARCHAR2
RETURN OG_Timer;
```

パラメータ

<i>interval</i>	タイマーが起動する間隔 (秒単位)。
<i>timerproc</i>	設定された時間間隔で実行するプロシージャの名前。

戻り値

新規に作成されたタイマーのハンドル。

OG_Make_Timerの例

```
/* The following procedure creates a timer that executes
**the procedure 'update_proc' every 30 seconds.
*/

PROCEDURE example(timer_name VARCHAR2) IS
    timer OG_Timer;
BEGIN
    timer := OG_Make_Timer(30, 'update_proc');
    OG_Set_Name(timer, timer_name);
END;
```

TOOLS_INTビルトイン

- TOOLINTADDPARAMETER
- TOOLINTCREATEPARAMETERLIST
- TOOLINTDELETEPARAMETER
- TOOLINTDESTROYPARAMETERLIST
- TOOLINTGETPARAMETERATTR
- TOOLINTGETPARAMETERLIST
- TOOLINTISNULL
- TOOLINTRUNPRODUCT
- TOOLINTSETPARAMETERATTR

TOOL_INT.ADD_PARAMETER

説明

このプロシージャで、指定されたパラメータ・リストにパラメータを追加します。

構文

```
PROCEDURE TOOL_INT.add_parameter
(list_hdl    TOOL_INT.PARAMLIST,
 param_name  CHAR,
 attr       TOOL_INT.PARAM_ATTR);

PROCEDURE TOOL_INT.add_parameter
(list_hdl    TOOL_INT.PARAMLIST,
 param_name  CHAR,
 attr       TOOL_INT.PARAM_ATTR);

PROCEDURE TOOL_INT.add_parameter
(list_hdl    TOOL_INT.PARAMLIST,
 param_name  CHAR,
 param_type  NUMBER,
 value      CHAR);
```

パラメータ

<i>list_hdl</i>	パラメータ・リストのハンドル。
<i>param_name</i>	追加するパラメータの名前。

<i>attr</i>	追加するパラメータのタイプと値を含むパラメータ属性レコード。
<i>param_type</i>	追加するパラメータのタイプ。この引数の値には、次のビルトイン定数のいずれかを指定してください。 TOOL_INT.DATA_PARAMETER パラメータが、ある製品から他の製品への問合せのマッピングを表すことを示します。 TOOL_INT.TEXT_PARAMETER パラメータが単一値であることを示します。
<i>value</i>	追加するパラメータの値。

使用上の注意

パラメータのタイプおよび値を含んでいるパラメータ属性レコードを指定するか、パラメータのタイプおよび値を直接指定できます。

Tool_Int.Add_Parameterの例

```
/* The following procedure creates a parameter list and
**adds several parameters to it:
*/

*/

PROCEDURE create_plist IS
    the_list tool_int.paramlist;
BEGIN
    the_list:=tool_int.create_parameter_list('my_plist');
    tool_int.add_parameter(the_list, 'userid',
        TOOL_INT.TEXT_PARAMETER, 'scott/tiger');
    tool_int.add_parameter(the_list, 'destype',
        TOOL_INT.TEXT_PARAMETER, 'printer');
    tool_int.add_parameter(the_list, 'copies',
        TOOL_INT.TEXT_PARAMETER, '2');
    tool_int.add_parameter(the_list, 'my_param',
        TOOL_INT.TEXT_PARAMETER, '67');
    tool_int.add_parameter(the_list, 'Q_1',
        TOOL_INT.DATA_PARAMETER, 'query0');
END;
```

TOOL_INT.CREATE_PARAMETER_LIST

説明

このファンクションは、指定された名前での新規のパラメータ・リストを作成します。

構文

```
FUNCTION TOOL_INT.create_parameter_list
  (name CHAR)
RETURN TOOL_INT.PARAMLIST;
```

パラメータ

<i>name</i>	作成するパラメータ・リストの名前。
-------------	-------------------

戻り値

新規に作成されたパラメータ・リストのハンドル。

TOOL_INT.Create_Parameter_Listの例

```
/* The following procedure creates a parameter list and
**adds several parameters to it:
*/

*/

PROCEDURE create_plist IS
  the_list tool_int.paramlist;
BEGIN
  the_list:=tool_int.create_parameter_list('my_plist');
  tool_int.add_parameter(the_list, 'userid',
    TOOL_INT.TEXT_PARAMETER, 'scott/tiger');
  tool_int.add_parameter(the_list, 'destype',
    TOOL_INT.TEXT_PARAMETER, 'printer');
  tool_int.add_parameter(the_list, 'copies',
    TOOL_INT.TEXT_PARAMETER, '2');
  tool_int.add_parameter(the_list, 'my_param',
    TOOL_INT.TEXT_PARAMETER, '67');
  tool_int.add_parameter(the_list, 'Q_1',
    TOOL_INT.DATA_PARAMETER, 'query0');
END;
```

TOOL_INT.DELETE_PARAMETER

説明

このプロシージャで、指定されたパラメータ・リストから指定されたパラメータを削除します。

構文

```
PROCEDURE TOOL_INT.delete_parameter
```

```
(list_hdl    TOOL_INT.PARAMLIST,  
 param_name  CHAR);
```

パラメータ

<i>list_hdl</i>	パラメータを削除するパラメータ・リストのハンドル。
<i>param_name</i>	削除するパラメータの名前。

Tool_Int.Delete_Parameterの例

```
/* The following procedure deletes the parameter 'username' from a parameter  
list:  
*/  
  
*/  
  
PROCEDURE example IS  
  list tool_int.paramlist;  
BEGIN  
  list := tool_int.get_parameter_list('list0');  
  tool_int.delete_parameter(list, 'username');  
END;
```

TOOL_INT.DESTROY_PARAMETER_LIST

説明

このプロシージャで、指定されたパラメータ・リストを破棄します。

構文

```
PROCEDURE TOOL_INT.destroy_parameter_list  
(list_hdl  TOOL_INT.PARAMLIST);
```

パラメータ

<i>list_hdl</i>	削除するパラメータ・リストのハンドル。
-----------------	---------------------

Tool_Int.Destroy_Parameter_Listの例

```
/* The following procedure creates a parameter list,  
**first destroying an existing list (if any):  
*/  
  
*/
```

```

PROCEDURE example IS
    list tool_int.paramlist;
BEGIN
    list := tool_int.get_parameter_list('list0');

    IF NOT tool_int.isnull(list) THEN
        tool_int.destroy_parameter_list(list);
    END IF;

    list := tool_int.create_parameter_list('list0');
END;
```

TOOL_INT.GET_PARAMETER_ATTR

説明

このプロシージャで、指定されたパラメータ・リスト内の指定パラメータの属性を取得します。

構文

```

PROCEDURE TOOL_INT.get_parameter_attr
    (list_hdl    TOOL_INT.PARAMLIST,
     param_name  CHAR,
     attr        TOOL_INT.PARAM_ATTR);

PROCEDURE TOOL_INT.get_parameter_attr
    (list_hdl    TOOL_INT.PARAMLIST,
     param_name  CHAR,
     param_type  NUMBER,
     value       CHAR);
```

パラメータ

<i>list_hdl</i>	パラメータを取得するパラメータ・リストのハンドル。
<i>param_name</i>	取得するパラメータの名前。
<i>attr</i>	パラメータ属性に設定される属性レコード。
<i>param_type</i>	このプロシージャをコールしたときに、パラメータのタイプ設定される変数。この引数の値には、次のビルトイン定数のいずれかを指定してください。 TOOL_INT.DATA_PARAMETER パラメータが、ある製品から他の製品への問合せのマッピングを表すことを示します。 TOOL_INT.TEXT_PARAMETER パラメータが単一値であることを示します。
<i>value</i>	このプロシージャをコールしたときに、パラメータの値が設定される変数。

使用上の注意

このプロシージャでパラメータの属性が設定されるようにパラメータ属性レコードを指定するか、別の変数を指定できます。

Tool_Int.Get_Parameter_Attrの例

```
/* The following procedure gets the value of the 'priority' parameter,
**and increases its value by one:
*/

*/

PROCEDURE example IS
  list    tool_int.paramlist;
  ptype   NUMBER;
  pvalue  VARCHAR2;
BEGIN
  list := tool_int.get_parameter_list('list0');
  tool_int.get_parameter_attr(list, 'priority', ptype,
    pvalue);

  pvalue := to_char(to_number(pvalue) + 1);

  tool_int.set_parameter_attr(list, 'priority',
    tool_int.TEXT_PARAMETER, pvalue);

END;
```

TOOL_INT.GET_PARAMETER_LIST

説明

このファンクションは、指定された名前のパラメータ・リストのハンドルを戻します。

構文

```
FUNCTION TOOL_INT.get_parameter_list
  (list_name CHAR)
RETURN TOOL_INT.paramlist;
```

パラメータ

<i>list_name</i>	取得するパラメータ・リストの名前。
------------------	-------------------

戻り値

名前付きパラメータ・リストのハンドル。

Tool_Int.Get_Parameter_Listの例

```
/* The following procedure creates a parameter list,
**first destroying an existing list (if any):
*/

*/

PROCEDURE example IS
    list tool_int.paramlist;
BEGIN
    list := tool_int.get_parameter_list('list0');

    IF NOT tool_int.isnull(list) THEN
        tool_int.destroy_parameter_list(list);
    END IF;

    list := tool_int.create_parameter_list('list0');
END;
```

TOOL_INT.ISNULL

説明

このファンクションは、指定されたパラメータ・リストのハンドルがNULLハンドルかどうかを判断します。

構文

```
FUNCTION TOOL_INT.isnull
    (list_hdl TOOL_INT.PARAMLIST)
RETURN BOOLEAN;
```

パラメータ

<i>list_hdl</i>	評価対象のハンドル。
-----------------	------------

戻り値

TRUE	ハンドルがNULLの場合。
FALSE	ハンドルがNULLでない場合。

使用上の注意

TOOL_INT.GET_PARAMETER_LISTがNULLハンドルを戻すのは、取得しようとするパラメータ・リストが存在しない場合です。

Tool_Int.IsNullの例

```
/* The following procedure creates a parameter list,
**first destroying an existing list (if any):
*/

*/

PROCEDURE example IS
    list tool_int.paramlist;
BEGIN
    list := tool_int.get_parameter_list('list0');

    IF NOT tool_int.isnull(list) THEN
        tool_int.destroy_parameter_list(list);
    END IF;

    list := tool_int.create_parameter_list('list0');
END;
```

TOOL_INT.RUN_PRODUCT

説明

このプロシージャで、サポートされている別のOracle製品を起動します。

構文

```
PROCEDURE TOOL_INT.run_product
    (product    NUMBER,
     module     CHAR,
     comm_mode  NUMBER,
     exec_mode  NUMBER,
     repository NUMBER,
     list_hdl   TOOL_INT.PARAMLIST);

PROCEDURE TOOL_INT.run_product
    (product    NUMBER,
     module     CHAR,
     comm_mode  NUMBER,
     exec_mode  NUMBER,
```

```
repository NUMBER,  
list_name CHAR);
```

パラメータ

<i>product</i>	起動されるOracle製品。この引数の値には、次のビルトイン定数のいずれかを指定してください。 TOOL_INT.BOOK Oracle Bookを起動します。 TOOL_INT.FORMS Formsを起動します。 TOOL_INT.REPORTS Reportsを起動します。
<i>module</i>	起動された製品で実行されるモジュール名。モジュールがデータベースに格納されている場合、この引数にはモジュール名のみを指定してください。モジュールがファイル・システムに格納されている場合は、モジュール・ファイルの絶対パス名か相対パス名を指定してください。
<i>comm_mode</i>	製品を起動するときの通信モード。この引数の値には、次のビルトイン定数のいずれかを指定してください。 TOOL_INT.SYNCHRONOUS 製品が同期的に起動されます。 TOOL_INT.ASYNCHRONOUS 製品が非同期的に起動されます。
<i>exec_mode</i>	起動された製品の実行モードです。この引数の値には、次のビルトイン定数のいずれかを指定してください。 TOOL_INT.BATCH 製品がバッチ・モードで起動されます。 TOOL_INT.RUNTIME 製品がランタイム・モードで起動されます。
<i>repository</i>	モジュールがファイル・システムとデータベースのどちらに格納されているかを指定します。この引数の値には、次のビルトイン定数のいずれかを指定してください。 TOOL_INT.DB モジュールはデータベースに格納されていることを示します。 TOOL_INT.FILESYSTEM モジュールはファイル・システムに格納されていることを示します。
<i>list_hdl</i>	起動された製品に渡されるパラメータ・リストのハンドル。
<i>list_name</i>	起動された製品に渡されるパラメータ・リストの名前。

使用上の注意

詳細は、Oracle製品のマニュアルを参照してください。

Tool_Int.Run_Productの例

```
/* The following procedure opens the Oracle Book document named  
**'catalog' and jumps to the hypertext target sailboard:  
*/
```

```
*/

PROCEDURE call_book is
  list  tool_int.paramlist;
BEGIN
  list:=tool_int.create_parameter_list('plist');

  tool_int.add_parameter(list, 'target',
    tool_int.TEXT_PARAMETER, 'sailboard');

  tool_int.RUN_PRODUCT(tool_int.FORMS, 'catalog',
    tool_int.ASYNCHRONOUS, tool_int.RUNTIME,
    tool_int.FILESYSTEM, list);

END;
```

TOOL_INT.SET_PARAMETER_ATTR

説明

このプロシージャで、指定されたパラメータ・リスト内の指定パラメータの属性を設定します。

構文

```
PROCEDURE TOOL_INT.set_parameter_attr
  (list_hdl    TOOL_INT.PARAMLIST,
   param_name  CHAR,
   attr        TOOL_INT.PARAM_ATTR);

PROCEDURE TOOL_INT.set_parameter_attr
  (list_hdl    TOOL_INT.PARAMLIST,
   param_name  CHAR,
   param_type  NUMBER,
   value       CHAR);
```

パラメータ

<i>list_hdl</i>	指定されたパラメータを含むパラメータ・リストのハンドル。	
<i>param_name</i>	設定するパラメータの名前。	
<i>attr</i>	設定するパラメータの属性が入っている属性レコード。	
<i>param_type</i>	設定するパラメータのタイプ。この引数の値には、次のビルトイン定数のいずれかを指定してください。	
	TOOL_INT.DATA_PARAMETER	パラメータが、ある製品から他の製品への問合せのマッピングを表すことを示します。
	TOOL_INT.TEXT_PARAMETER	パラメータが単一値であることを示します。
<i>value</i>	設定するパラメータの値。	

使用上の注意

このプロシージャでパラメータ属性の設定に使用するパラメータ属性レコードを指定するか、属性を含む別の変数を指定できます。

Tool_Int.Set_Parameter_Attrの例

```
/* The following procedure gets the value of the 'priority'
**parameter, and increases its value by one:
*/

*/

PROCEDURE example IS
  list    tool_int.paramlist;
  ptype   NUMBER;
  pvalue  VARCHAR2;
BEGIN
  list := tool_int.get_parameter_list('list0');
  tool_int.get_parameter_attr(list, 'priority', ptype,
    pvalue);

  pvalue := to_char(to_number(pvalue) + 1);

  tool_int.set_parameter_attr(list, 'priority',
    tool_int.TEXT_PARAMETER, pvalue);

END;
```

ウィンドウ・ビルトイン

OG_Destroy (ウィンドウ)

OG_Get_Window

OG_Hide_Window

OG_Make_Window

OG_Show_Window

OG_Destroy (ウィンドウ)

説明

このプロシージャで、指定されたウィンドウを破棄します。これにより、そのウィンドウはクローズされますが、内容には影響しません。

構文

```
PROCEDURE OG_Destroy  
    (window_hdl OG_Window);
```

パラメータ

<i>window_hdl</i>	破棄するウィンドウのハンドル。
-------------------	-----------------

OG_Destroy (ウィンドウ) の例

```
/* Suppose a user selects a country in a map of the world,  
**and the application creates a new window to display  
**information about sales in that country. When the user selects  
**another country, you may want to destroy the old window  
**and create another one containing information about the new country.  
*/  
  
PROCEDURE destroy_USA IS  
    the_window OG_Window;  
BEGIN  
    the_window:=OG_Get_Window('Usa_Window');  
    OG_Destroy(The_Window);  
END;
```

OG_Get_Window

説明

このファンクションは、指定されたウィンドウのハンドルを戻します。

構文

```
FUNCTION OG_Get_Window  
    (window_name VARCHAR2)  
RETURN OG_Window;
```

パラメータ

<i>window_name</i>	ハンドルが戻されるウィンドウの名前。
--------------------	--------------------

戻り値

指定されたウィンドウのハンドル。指定のウィンドウが存在しない場合は、NULLハンドルが戻ります。

使用上の注意

ウィンドウが図表のメイン・ウィンドウ(「主レイアウト」ウィンドウ)や、プログラムによって作成されたウィンドウの場合もあります。

OG_Get_Windowの例

```
/* Suppose the main window-which was previously hidden-contains information  
**that is now useful to view.The following procedure will show it:  
*/  
  
*/  
  
PROCEDURE show_main_window IS  
    the_main_window    OG_Window;  
BEGIN  
    the_main_window:=OG_Get_Window('Main Layout');  
    OG_Show_Window(The_Main_Window);  
END;
```

OG_Hide_Window

説明

このプロシージャで、指定されたウィンドウを隠します。

構文

```
PROCEDURE OG_Hide_Window  
    (window_hdl OG_Window);
```

パラメータ

<i>window_hdl</i>	隠されるウィンドウのハンドル。
-------------------	-----------------

使用上の注意

ウィンドウは破棄されません。そのため、そのウィンドウをすぐに必要としないときは隠し、必要になったときに再び表示できます。

OG_Hide_Windowの例

```
/* Suppose the main layout window contains a chart that the user  
**no longer needs to see.The following procedure will hide it temporarily.  
**Remember that this does not destroy the window; it will still exist  
**and be available to be shown again when needed.  
*/
```

```
PROCEDURE hide_main_window IS  
    the_main_window OG_Window;  
BEGIN  
    the_main_window:=OG_Get_Window('Main Layout');  
    OG_Hide_Window(The_Main_Window);  
END;
```

OG_Make_Window

説明

このファンクションはウィンドウを作成します。

構文

```
FUNCTION OG_Make_Window
```



```
(position OG_Point,  
height NUMBER,  
width NUMBER)  
RETURN OG_Window;
```

パラメータ

<i>position</i>	ウィンドウの左上角のx座標とy座標（画面解像度単位）。
<i>height</i>	ウィンドウの高さ(画面解像度単位)。
<i>width</i>	ウィンドウの幅（画面解像度単位）。

戻り値

新規に作成されたウィンドウのハンドル。

OG_Make_Windowの例

```
/* The following function creates a 5"x4" window  
**in the upper left corner of the screen:  
*/  
  
*/  
  
FUNCTION example(window_name VARCHAR2) RETURN OG_Window IS  
window OG_Window;  
pos OG_Point;  
height NUMBER;  
width NUMBER;  
BEGIN  
pos.x := 0;  
pos.y := 0;  
width := 5 * OG_Get_Ap_Hscreen_Res;  
height := 4 * OG_Get_Ap_Vscreen_Res;  
  
window := OG_Make_Window(Pos, height, width);  
OG_Set_Name(Window, window_name);  
RETURN(window);  
END;
```

OG_Show_Window

説明

このプロシージャで、指定されたウィンドウを表示します。

構文

```
PROCEDURE OG_Show_Window  
    (window_hdl  OG_Window);
```

パラメータ

<i>window_hdl</i>	表示するウィンドウのハンドル。
-------------------	-----------------

OG_Show_Windowの例

```
/* Suppose the main window-which was previously hidden  
**contains information that is now useful to view.  
**The following procedure will show it.  
*/
```

```
PROCEDURE show_main_window IS  
    the_main_window  OG_Window;  
BEGIN  
    the_main_window:=OG_Get_Window('Main Layout');  
    OG_Show_Window(The_Main_Window);  
END;
```

プロパティ

アプリケーション・プロパティ

接続文字列プロパティ

カーソル・プロパティ

水平レイアウト解像度プロパティ

水平画面解像度プロパティ

パスワード・プロパティ

プラットフォーム・プロパティ

ユーザー名プロパティ

垂直レイアウト解像度プロパティ

垂直画面解像度プロパティ

接続文字列プロパティ

説明

現行のデータベース接続用のデータベース接続文字列。接続していないと、このプロパティは NULLになります。

構文

```
FUNCTION OG_Get_Ap_Connection  
RETURN VARCHAR2;
```

パラメータ

なし。

接続文字列プロパティの例

```
/*The following open trigger procedure  
**displays the connection string for  
**the current database connection  
**to a text object.  
*/
```

```

PROCEDURE connection IS connstr  varchar2(10);
BEGIN
  connstr := og_get_ap_connection;
  if connstr = NULL then
    og_set_str(og_get_object('text object'), 'NULL', true, true);
  else
    og_set_str(og_get_object('text object'), connstr, true, true);
  end if;
END;
```

カーソル・プロパティ

説明

使用するマウス・カーソルの名前。このプロパティの値として、次の文字列のいずれかを指定できます。

```

default
insertion
crosshair
help
busy
```

各カーソルの外観は、システムによって異なります。詳細は、使用しているマニュアルを参照してください。このプロパティに無効な値を設定すると、その値は'default'とみなされます。

構文

```

PROCEDURE OG_Set_Ap_Cursor
  (cursor VARCHAR2);

FUNCTION OG_Get_Ap_Cursor
RETURN VARCHAR2;
```

パラメータ

<i>cursor</i>	使用するマウス・カーソル。
---------------	---------------

カーソル・プロパティの例

```

/*The following procedure changes
**the shape of the cursor depending on
**which layer the user selects.
*/

PROCEDURE ChangeCursor (buttonobj IN og_object,
  hitobj IN og_object, win IN og_window,
  eventinfo IN og_event) IS
```

```
    cur varchar2(10);
BEGIN
    mycur := og_get_ap_cursor;
    if cur = 'default' then
        og_set_ap_cursor('insertion');
    elsif cur = 'insertion' then
        og_set_ap_cursor('crosshair');
    elsif cur = 'crosshair' then
        og_set_ap_cursor('help');
    elsif cur = 'help' then
        og_set_ap_cursor('busy');
    elsif cur = 'busy' then
        og_set_ap_cursor('default');
    end if;
END;;
```

水平レイアウト解像度プロパティ

説明

レイアウトの水平解像度。この値は、レイアウトの水平方向の1インチ内のレイアウト単位の数です。

構文

```
FUNCTION OG_Get_Ap_Hlayout_Res
RETURN NUMBER;
```

パラメータ

なし。

水平レイアウト解像度プロパティの例

```
/*The following procedure displays
**current horizontal layout resolution
**to a text object.
*/
```

```
PROCEDURE h_layout IS
    h_layout number;
BEGIN
    h_layout := og_get_ap_hlayout_res;
    og_set_str(og_get_object('text object'), h_layout, true, true);
END;
```

水平画面解像度プロパティ

説明

画面の水平解像度。この値は、画面の水平方向1インチ内の画面解像度（ピクセル）単位の数です。

構文

```
FUNCTION OG_Get_Ap_Hscreen_Res  
RETURN NUMBER;
```

パラメータ

なし。

水平画面解像度プロパティの例

```
/*The following procedure displays  
**current horizontal screen  
**to a text object..  
*/  
  
PROCEDURE HRES IS  
  h_res number;  
BEGIN  
  h_res := og_get_ap_vscreen_res;  
  og_set_str(og_get_object('text object'), h_res, true, true);  
END;
```

パスワード・プロパティ

説明

現行のデータベース接続用のパスワード。接続していない場合や、*Keep_Password*作業環境を「いいえ」に設定している場合、このプロパティはNULLになります。

構文

```
FUNCTION OG_Get_Ap_Password  
RETURN VARCHAR2;
```

パラメータ

なし。

パスワード・プロパティの例

```
/*The following open trigger procedure
** displays the password for the current
**database connection to a text object.
*/

PROCEDURE password IS
    pw          varchar2(10);
BEGIN
    pw := og_get_ap_password;
    if pw = NULL then
        og_set_str(og_get_object('text object'), 'NULL', true, true);
    else
        og_set_str(og_get_object('text object'), pw, true, true);
    end if;
END;
```

プラットフォーム・プロパティ

説明

Graphics Builderが実行されるプラットフォーム。このプロパティには、次のビルトイン定数を指定できます。

OG_Macintosh_Platform	プラットフォームがApple Macintoshである。
OG_Motif_Platform	プラットフォームがOSF/MOTIFである。
OG_Mswindows_Platform	プラットフォームがMicrosoft Windowsである。
OG_Pm_Platform	プラットフォームがPresentation Managerである。
OG_X_Platform	プラットフォームがX Window Systemである。

構文

```
FUNCTION OG_Get_Ap_Platform
RETURN NUMBER;
```

パラメータ

なし。

プラットフォーム・プロパティの例

```
/*The following procedure displays
**the platform type on which Oracle
```



```
**Graphics is currently running to  
**a text object.  
*/
```

```
PROCEDURE platform IS  
  ptform      number;  
BEGIN  
  ptform := og_get_ap_platform;  
  if ptform = og_macintosh_platform then  
    og_set_str(og_get_object('text object'), 'og_macintosh_platform',  
true, true);  
  elsif ptform = og_motif_platform then  
    og_set_str(og_get_object('text object'), 'og_motif_platform', true,  
true);  
  elsif ptform = og_mswindows_platform then  
    og_set_str(og_get_object('text object'), 'og_mswindows_platform',  
true, true);  
  elsif ptform = og_pm_platform then  
    og_set_str(og_get_object('text object'), 'og_pm_platform', true,  
true);  
  elsif ptform = og_x_platform then  
    og_set_str(og_get_object('text object'), 'og_x_platform', true,  
true);  
  end if;  
END;
```

ユーザー名プロパティ

説明

現行のデータベース接続用のユーザー名。接続していないと、このプロパティはNULLになります。

構文

```
FUNCTION OG_Get_Ap_Username  
RETURN VARCHAR2;
```

パラメータ

なし。

ユーザー名プロパティの例

```
/*The following open trigger procedure  
**displays the username for the current  
**database connection to a text object.
```

```
*/

PROCEDURE username IS
  usr  varchar2(10);
BEGIN
  usr := og_get_ap_username;
  if usr = NULL then
    og_set_str(og_get_object('text object'), 'NULL', true, true);
  else
    og_set_str(og_get_object('text object'), usr, true, true);
  end if;
END;
```

垂直レイアウト解像度プロパティ

説明

レイアウトの垂直解像度。この値は、レイアウトの垂直方向の1インチ内のレイアウト単位の数です。

構文

```
FUNCTION OG_Get_Ap_Vlayout_Res
RETURN NUMBER;
```

パラメータ

なし。

垂直レイアウト解像度プロパティの例

```
/*The following procedure displays current
**vertical layout resolution to a text object..
*/

PROCEDURE v_layout IS
  v_layout  number;
BEGIN
  v_layout := og_get_ap_vlayout_res;
  og_set_str(og_get_object('text object'), v_layout, true, true);
END;
```

垂直画面解像度プロパティ

説明

画面の垂直解像度。この値は、画面の垂直方向1インチ内の画面解像度単位（ピクセル）の数です。

構文

```
FUNCTION OG_Get_Ap_Vscreen_Res  
RETURN NUMBER;
```

パラメータ

なし。

垂直画面解像度プロパティの例

```
/*The following procedure displays current  
**vertical screen resolution to a text object:  
*/  
  
PROCEDURE VRES IS  
  v_res number;  
BEGIN  
  v_res := og_get_ap_vscreen_res;  
  og_set_str(og_get_object('text object'), v_res, true, true);  
END;
```

円弧プロパティ

- 基本円弧プロパティ
- 閉じ形状プロパティ
- 塗り形状プロパティ

基本円弧プロパティ

説明

左上角のx座標とy座標、および円弧がカットされる楕円の元として使用する四角形の高さと幅。

構文

```
PROCEDURE OG_Set_Basearc
(
  arc OG_Object,
  basearc OG_Arc,
  damage BOOLEAN := TRUE,
  update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Basearc
(
  arc OG_Object)
RETURN OG_Arc;
```

パラメータ

arc	描写される円弧オブジェクト。
basearc	左上角のx座標とy座標、および楕円から円弧をカットする際に基準として使用する四角形の高さと幅。
damage	ダメージ・フラグ。
update_bbox	バウンディング・ボックス更新フラグ。

基本円弧プロパティの例

```
/*The following procedure reads
**information from an existing arc,
**reduces all data by half, and
**updates the arc object.
*/

PROCEDURE base_arc IS
  arc og_arc;
BEGIN
```

```
arc := og_get_basearc(og_get_object('arc'));
arc.x := arc.x/2;
arc.y := arc.y/2;
arc.height :=arc.height/2;
arc.width := arc.width/20;
arc.sangle := arc.sangle/2;
arc.eangle := arc.eangle/2;
og_set_basearc(og_get_object('arc'), arc);
END;
```

閉じ形状プロパティ

説明

円弧の閉じ形状。このプロパティには、次のいずれかを指定できます。

TRUE 円弧が閉じた形状になる。

FALSE 円弧が開いた形状になる。

構文

```
PROCEDURE OG_Set_Arc_Closed
(arc OG_Object,
closed BOOLEAN,
damage BOOLEAN := TRUE,
update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Arc_Closed
(arc OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>arc</i>	描写される円弧オブジェクト。
<i>closed</i>	円弧の閉じ形状。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

閉じ形状プロパティの例

```
/*The following procedure reads the
**closure of an existing arc.If closure
**equals TRUE, it fills the arc with red
**and sets the closure value to FALSE;
**if closure equals FALSE, it fills the
**arc with blue and sets the value to TRUE.
*/
```

```
PROCEDURE closure IS
  cls  BOOLEAN; arc  BOOLEAN;
  arc  og_object;
BEGIN
  arc := og_get_object('arc');
  cls := og_get_arc_closed(arc);
  if cls = TRUE then
    og_set_bfcolor(arc, 'red');
    og_set_arc_closed(arc, FALSE);
  else
    og_set_bfcolor(arc, 'blue');
    og_set_arc_closed(arc, TRUE);
  end if;
END;
```

塗り形状プロパティ

説明

円弧の塗り形状。このプロパティには、次のビルトイン定数を指定できます。

OG_Chord_Arcfill 円弧の塗り形状が弦型になる。

OG_Pie_Arcfill 円弧の塗り形状が完全な扇型になる。

構文

```
PROCEDURE OG_Set_Arcfill
  (arc OG_Object,
   arcfill NUMBER,
   damage BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Arcfill
  (arc OG_Object)
RETURN NUMBER;
```

パラメータ

<i>arc</i>	描写される円弧オブジェクト。
<i>arcfill</i>	円弧の形状。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

塗り形状プロパティの例

*The following procedure reads the

```
**arcfill from an arc, prints the value to a  
**text object, assigns a different value  
** to the arcfill value..  
*/
```

```
PROCEDURE fill IS
```

```
text  og_object;  
  arc  og_object;  
  num  NUMBER;
```

```
BEGIN
```

```
  text := og_get_object('text object');  
  arc  := og_get_object('arc');  
  num  := og_get_arcfill(arc);  
  og_set_str(text, num);  
  og_set_arcfill(arc, og_chord_arcfill);
```

```
END;
```

軸（日付）プロパティ

自動最大値プロパティ

自動最小値プロパティ

自動主目盛間隔プロパティ

カスタム書式プロパティ

曜日書式プロパティ

年の最初の月プロパティ

ラベル・プロパティ

最大値プロパティ

最小値プロパティ

月書式プロパティ

四半期書式プロパティ

週末をスキップ・プロパティ

ステップ・プロパティ

年書式プロパティ

自動最大値プロパティ

説明

軸最大値を「自動」に設定するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Date_Automax  
  (axis OG_Axis,  
   automax BOOLEAN,  
   maximun DATE);  
  
FUNCTION OG_Get_Date_Automax  
  (axis OG_Axis)  
RETURN BOOLEAN;
```


パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>automax</i>	軸最大値を「自動」に設定するかどうかを指定します。
<i>maximum</i>	最大軸値を指定します（ただし、 <i>automax</i> がFALSEの場合のみ）。

自動最大値プロパティの例

```
/*The following procedure checks if axis
**Y1's datemaximum is set to auto.If
**the return value is TRUE, it resets the
**value to FALSE with default_max;
**if the return value isFALSE, it resets
**the value to TRUE after reading the
**specified maximum axis value.
/*

PROCEDURE datemax IS
    template    og_template;
    axis og_axis;
    val1 date;
    val2 boolean;
    default_max date := '06-dec-99';
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_y1_axis);
    val2 := og_get_date_automax(axis);
    if val2 = true then
        og_set_date_automax(axis, false, default_max);
        val1 := og_get_date_maximum(axis);
    elsif val2 = false then
        og_set_date_automax(axis, true, default_max);
    end if;
END;
```

自動最小値プロパティ

説明

軸最小値を「自動」に設定するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Date_Automin
    (axis OG_Axis,
    automin BOOLEAN,
    minimum DATE);
```

```
FUNCTION OG_Get_Date_Automin
  (axis OG_Axis)
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>automin</i>	軸最小値を「自動」に設定するかどうかを指定します。
<i>minimum</i>	最小軸値を指定します（ただし、 <i>automin</i> がFALSEの場合のみ）。

自動最小値プロパティの例

```
/*The following procedure checks if axis
**Y1's dateminimum is set to auto.If the
**return value is TRUE, it resets thevalue
**to FALSE with default_min; if the return
**value isFALSE, it resets the value to
**TRUE after reading the specifiedminimum
**axis value.
*/

PROCEDURE datemin IS
  template og_template;
  axis og_axis;
  val1 date;
  val2 boolean;
  default_min date := '01-dec-79';
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val2 := og_get_date_automin(axis);
  if val2 = true then
    og_set_date_automin(axis, false, default_min);
    val1 := og_get_date_minimum(axis);
  elsif val2 = false then
    og_set_date_automin(axis, true, default_min);
  end if;
END;
```

自動主目盛間隔プロパティ

説明

主目盛間隔を「自動」に設定するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Date_Autostep
```

```

(axis      OG_Axis,
 autostep  BOOLEAN,
 step      NUMBER);

FUNCTION OG_Get_Date_Autostep
(axis OG_Axis)
RETURN BOOLEAN;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>autostep</i>	主目盛間隔を「自動」に設定するかどうかを指定します。
<i>step</i>	主目盛間隔を指定します（ただし、 <i>autostep</i> がFALSEの場合のみ）。

自動主目盛間隔プロパティの例

```

/*The following procedure checks if axis
**Y1's date step is set to auto.If the
**return value is TRUE, it resets the value
**to FALSE with default_step; if the return
**value is FALSE,it resets the value
**to TRUE after reading the specified step
**value.
*/

PROCEDURE datestep IS
  template  og_template;
  axis og_axis;
  val  boolean;
  num  number;
  default_step number := og_second_step;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val:= og_get_date_autostep(axis);
  if val = true then
    og_set_date_autostep(axis, false, default_step);
    num := og_get_date_step(axis);
  elsif val = false then
    og_set_date_autostep(axis, true, default_step);
  end if;
END;

```

カスタム書式プロパティ

説明

軸目盛きざみラベルのカスタム日付書式です。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Custfmt
  (axis      OG_Axis,
   custfmt   VARCHAR2);

FUNCTION OG_Get_Custfmt
  (axis OG_Axis)
RETURN VARCHAR2;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>custfmt</i>	軸目盛きざみラベルのカスタム日付書式です。

カスタム書式プロパティの例

```
/*The following procedure reads the
**Custom format value andcompares it
**with the variable 'default_format';
**ifthe two value are not equal,
**it resets the current format to the
**value of the 'default_format'.
*/

PROCEDURE customfmt IS
  template   og_template;
  axis og_axis;
  val  varchar2(10);
  default_format varchar2(10) := 'DD_YY_MM';
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val := og_get_custfmt(axis);
  if val != default_format then
    og_set_custfmt(axis, default_format);
  end if;
END;
```

曜日書式プロパティ

説明

曜日ラベルの表示形式を決めます。このプロパティには、次のビルトイン定数を指定できます。

OG_Firstletter_Fmt

OG_Threeletter_Fmt

構文

```
PROCEDURE OG_Set_Dayfmt
  (axis    OG_Axis,
   dayfmt  NUMBER);

FUNCTION OG_Get_Dayfmt
  (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>dayfmt</i>	曜日ラベルの表示形式を決めます。

曜日書式プロパティの例

```
/*The following procedure checks the
**day-of-week format.If the current format
**is First-Letter format, it
**resets the value to Three-Letter
**format, and vice versa.
*/

PROCEDURE dayfmt IS
  template    og_template;
  axis og_axis;
  num  number;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  num:= og_get_monthfmt(axis);
  if num = og_firstletter_fmt then
    og_set_monthfmt(axis, og_threeletter_fmt);
  elsif num = og_threeletter_fmt then
    og_set_monthfmt(axis, og_firstletter_fmt);
  end if;
END;
```

年の最初の月プロパティ

説明

新しい年度が始まる最初の月です。このプロパティには、次のビルトイン定数を指定できます。

OG_Jan_Month

OG_Feb_Month

OG_Mar_Month

OG_Apr_Month

OG_May_Month

OG_Jun_Month

OG_Jul_Month

OG_Aug_Month

OG_Sep_Month

OG_Oct_Month

OG_Nov_Month

OG_Dec_Month

構文

```
PROCEDURE OG_Set_Firstmon
  (axis      OG_Axis,
   firstmon  NUMBER);

FUNCTION OG_Get_Firstmon
  (axis  OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>firstmon</i>	新しい年度が始まる最初の月です。

年の最初の月プロパティの例

```
/*The following reads the first month
**value and resets the value to the next
**acceptable value.
*/
```

```
PROCEDURE firstmonth IS
    template    og_template;
    axis og_axis;
    num number;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_y1_axis);
    num:= og_get_firstmon(axis);
    if num = og_jan_month then
        og_set_firstmon(axis, og_feb_month);
    elsif num = og_feb_month then
        og_set_firstmon(axis, og_mar_month);
    elsif num = og_mar_month then
        og_set_firstmon(axis, og_apr_month);
    elsif num = og_apr_month then
        og_set_firstmon(axis, og_may_month);
    elsif num = og_may_month then
        og_set_firstmon(axis, og_jun_month);
    elsif num = og_jun_month then
        og_set_firstmon(axis, og_jul_month);
    elsif num = og_jul_month then
        og_set_firstmon(axis, og_aug_month);
    elsif num = og_aug_month then
        og_set_firstmon(axis, og_sep_month);
    elsif num = og_sep_month then
        og_set_firstmon(axis, og_oct_month);
    elsif num = og_oct_month then
        og_set_firstmon(axis, og_nov_month);
    elsif num = og_nov_month then
        og_set_firstmon(axis, og_dec_month);
    else og_set_firstmon(axis, og_jan_month);
    end if;
END;
```

ラベル・プロパティ

説明

主目盛間隔を指定すると同時に、目盛きざみラベルの表示形式を指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_No_Labels

OG_Second_Labels

OG_Minute_Labels

OG_Hour_Labels

OG_Ampm_Labels

OG_Day_Labels

OG_Dayofweek_Labels

OG_Week_Labels

OG_Month_Labels

OG_Quarter_Labels

OG_Year_Labels

OG_Custom_Labels (*labels*をこの値に設定する場合は、カスタム書式プロパティでカスタム日付書式を指定する必要があります。)

構文

```
PROCEDURE OG_Set_Labels
  (axis      OG_Axis,
   labels    NUMBER);

FUNCTION OG_Get_Labels
  (axis      OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>labels</i>	主目盛きざみと目盛きざみラベルが表示される軸に沿った主目盛間隔のみでなく、目盛きざみラベルの表示も指定します。

ラベル・プロパティの例

```
/*The following procedure determines
**if any label boxesare checked.
**If checked label boxes are found,
**it unchecksall labels; if no checked
**labels are found, it checks the 'Year'
**check box.
*/

PROCEDURE labels IS
  template   og_template;
  axis og_axis;
  num  number;
BEGIN
  template := og_get_template('template0');
```



```

axis := og_get_axis(template, og_y1_axis);

num:= og_get_labels(axis);
if num != og_no_labels then
  og_set_labels(axis, og_no_labels);
else og_set_labels(axis, og_year_labels);
end if;
END;

```

最大値プロパティ

説明

最大軸値を指定します（ただし、「自動最大値」がFALSEの場合のみ）。

構文

（前述のOG_Set_Date_Automaxを参照してください。）

```

FUNCTION OG_Get_Date_Maximum
  (axis OG_Axis)
RETURN DATE;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
-------------	-----------------

最大値プロパティの例

```

/*The following procedure checks if
**axis Y1's datemaximum is set to auto.
**If the return value is TRUE,
**itresets the value to FALSE with
**default_max; if the return valueis
**FALSE, it resets the value to
**TRUE after reading thespecified
**maximum axis value.
/*

PROCEDURE datemax IS
  template   og_template;
  axis og_axis;
  val1 date;
  val2 boolean;
  default_max date := '06-dec-99';
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);

```

```
val2 := og_get_date_automax(axis);
if val2 = true then
  og_set_date_automax(axis, false, default_max);
  val1 := og_get_date_maximum(axis);
elsif val2 = false then
  og_set_date_automax(axis, true, default_max);
end if;
END;
```

最小値プロパティ

説明

最小軸値を指定します（ただし、「自動最小値」がFALSEの場合のみ）。

構文

（OG_Set_Date_Autominを参照してください。）

```
FUNCTION OG_Get_Date_Minimum
(axis OG_Axis)
RETURN DATE;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
-------------	-----------------

最小値プロパティの例

```
/*The following procedure checks if
**axis Y1's dateminimum is set to auto.
**If the return value is TRUE, it resets
**the value to FALSE with default_min;
**if the return value isFALSE, it resets
**the value to TRUE after reading the
**specified minimum axis value.
*/

PROCEDURE datemin IS
  template   og_template;
  axis og_axis;
  val1 date;
  val2 boolean;
  default_min date := '01-dec-79';
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val2 := og_get_date_automin(axis);
```

```

if val2 = true then
  og_set_date_automin(axis, false, default_min);
  val1 := og_get_date_minimum(axis);
elsif val2 = false then
  og_set_date_automin(axis, true, default_min);
end if;
END;

```

月書式プロパティ

説明

月ラベルの表示形式を決めます。このプロパティには、次のビルトイン定数を指定できます。

OG_Firstletter_Fmt

OG_Threeletter_Fmt

構文

```

PROCEDURE OG_Set_Monthfmt
  (axis OG_Axis,
   monthfmt

NUMBER); FUNCTION OG_Get_Monthfmt
  (axis OG_Axis)
RETURN NUMBER;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>monthfmt</i>	月ラベルの表示形式を決めます。

月書式プロパティの例

```

/*The following procedure checks the
**Month format.If the current format
**is First-Letter format, it
**resets the value to Three-Letter
**format, and vice versa.
*/

PROCEDURE monthfmt IS
  template   og_template;
  axis og_axis;
  num number;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);

```

```
num:= og_get_monthfmt(axis);
if num = og_firstletter_fmt then
  og_set_monthfmt(axis, og_threeletter_fmt);
elsif num = og_threeletter_fmt then
  og_set_monthfmt(axis, og_firstletter_fmt);
end if;
END;
```

四半期書式プロパティ

説明

四半期ラベルの表示形式を決めます。このプロパティには、次のビルトイン定数を指定できます。

OG_Arabic_Fmt

OG_Roman_Fmt

構文

```
PROCEDURE OG_Set_Qtrfmt
  (axis    OG_Axis,
   qtrfmt  NUMBER);

FUNCTION OG_Get_Qtrfmt
  (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>qtrfmt</i>	四半期ラベルの表示形式を決めます。

四半期書式プロパティの例

```
/*The following procedure checks the
**Quarter format.If the current
**format is Arabic format, it resets
**the value to Roman format, and vice versa.
*/

PROCEDURE qtrfmt IS
  template og_template;
  axis og_axis;
  num number;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  num:= og_get_qtrfmt(axis);
```

```

if num = og_arabic_fmt then
  og_set_qtrfmt(axis, og_roman_fmt);
elsif num = og_roman_fmt then
  og_set_qtrfmt(axis, og_arabic_fmt);
end if;
END;

```

週末をスキップ・プロパティ

説明

軸値を計算するときに週末を無視するかどうかを指定します。

構文

```

PROCEDURE OG_Set_Skipwknds
  (axis OG_Axis,
   skipwknds BOOLEAN);

FUNCTION OG_Get_Skipwknds
  (axis OG_Axis)
RETURN BOOLEAN;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>skipwknds</i>	軸の値を計算するときに週末を無視するかどうかを指定します。

週末をスキップ・プロパティの例

```

/*The following procedure checks whether
**weekends are ignored when calculating
**axis values.If the value of weekend
**is set to TRUE (ignored), the procedure
**resets the value toFALSE (include
**in the calculation) and vice versa.
*/

PROCEDURE skipwknds IS
  template og_template;
  axis og_axis;
  val boolean;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val:= og_get_skipwknds(axis);
  if val = true then
    og_set_skipwknds(axis, false);

```

```
elseif val = false then
  og_set_skipwknds(axis, true);
end if;
END;
```

ステップ・プロパティ

説明

主目盛間隔を指定します（ただし、「自動ステップ」がFALSEの場合のみ）。このプロパティには、次のビルトイン定数を指定できます。

OG_Second_Step

OG_Minute_Step

OG_Hour_Step

OG_Day_Step

OG_Week_Step

OG_Month_Step

OG_Quarter_Step

OG_Year_Step

構文

（OG_Set_Date_Autostepを参照してください。）

```
FUNCTION OG_Get_Date_Step
  (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
-------------	-----------------

ステップ・プロパティの例

```
/*The following procedure checks if
**axis Y1's date step is set to auto.
**If the return value is TRUE, it resets
**the value to FALSE with default_step;
**if the return value is FALSE,
**it resets the value to TRUE
**after reading the specified step
**value.
```

```

*/

PROCEDURE datestep IS
    template    og_template;
    axis og_axis;
    val boolean;
    num number;
    default_step number := og_second_step;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_y1_axis);
    val:= og_get_date_autostep(axis);
    if val = true then
        og_set_date_autostep(axis, false, default_step);
        num := og_get_date_step(axis);
    elsif val = false then
        og_set_date_autostep(axis, true, default_step);
    end if;
END

```

年書式プロパティ

説明

年ラベルの表示形式を決めます。このプロパティには、次のビルトイン定数を指定できます。

OG_Fourdigit_Fmt

OG_Twodigit_Fmt

構文

```

PROCEDURE OG_Set_Yearfmt
    (axis OG_Axis,
     yearfmt NUMBER);

FUNCTION OG_Get_Yearfmt
    (axis OG_Axis)
RETURN NUMBER;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>yearfmt</i>	年ラベルの表示形式を決めます。

年書式プロパティの例

```

/*The following procedure checks the Year
**format.If the current format is Two-Digit

```

```
**format, it resets the value to  
**Four-Digit format, and vice versa.  
*/
```

```
PROCEDURE yearfmt IS  
  template  og_template;  
  axis og_axis;  
  num  number;  
BEGIN  
  template := og_get_template('template0');  
  axis := og_get_axis(template, og_y1_axis);  
  num:= og_get_yearfmt(axis);  
  if num = og_fourdigit_fmt then  
    og_set_yearfmt(axis, og_twodigit_fmt);  
  elsif num = og_twodigit_fmt then  
    og_set_yearfmt(axis, og_fourdigit_fmt);  
  end if;  
END;
```


軸（一般）プロパティ

軸ラベル・プロパティ

軸タイプ・プロパティ

項目ラベル・プロパティ

軸の方向プロパティ

主目盛格子プロパティ

主目盛きざみプロパティ

補助目盛格子プロパティ

補助目盛りきざみプロパティ

補助目盛りきざみ数プロパティ

位置プロパティ

目盛きざみラベル回転プロパティ

目盛きざみラベル・プロパティ

目盛きざみ位置プロパティ

軸ラベル・プロパティ

説明

軸に沿って値を識別するラベルを表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Axislabel  
  (axis      OG_Axis,  
   axislabel BOOLEAN);  
  
FUNCTION OG_Get_Axislabel  
  (axis OG_Axis)  
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>axislabel</i>	軸に沿って値を識別する項目ラベルを表示するかどうかを指定します。

軸ラベル・プロパティの例

```
/*The following procedure determines if
**the Axis Label checkbox is checked.
**If the box is checked, it unchecks
**it, and vice versa.
*/

PROCEDURE GenAxisLbl IS
  template      og_template;
  x_axis        og_axis;
  val           boolean;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);

  val := og_get_axislabel(x_axis);
  if val = true then
    og_set_axislabel(x_axis, false);
  else
    og_set_axislabel(x_axis, true);
  end if;
END;
```

軸タイプ・プロパティ

説明

使用する軸のタイプを指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Continuous_Axistype

OG_Date_Axistype

OG_Discrete_Axistype

構文

```
PROCEDURE OG_Set_Axistype
  (axis
  OG_Axis,
  axistype
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>axistype</i>	使用する軸のタイプを指定します。

軸タイプ・プロパティの例

```
/*The following procedure reads the
**current axis type.If the current type
**is CONTINUOUS, it resets the type
**to DISCRETE, or vice versa.If the
**current type is DATE, it changes the
**year format.
*/

PROCEDURE GenAxisType IS
    template    og_template;
    axis og_axis;
    num    number;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_x_axis);
    num := og_get_axistype(axis);
    if num = og_discrete_axistype then
        og_set_axistype(axis,og_continuous_axistype);
    elsif num = og_continuous_axistype then
        og_set_axistype(axis, og_discrete_axistype);
    elsif num = og_date_axistype then
        og_set_yearfmt(axis, og_twodigit_fmt);
    end if;
END;
```

項目ラベル・プロパティ

説明

軸に沿って表示するラベルのテキストを指定します。

構文

```
PROCEDURE OG_Set_Custlabel
    (axis      OG_Axis,
     custlabel VARCHAR2);

FUNCTION OG_Get_Custlabel
    (axis OG_Axis)
RETURN VARCHAR2;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>custlabel</i>	軸に沿って表示するラベルのテキストを指定します。

項目ラベル・プロパティの例

```
/*The following procedure reads the current
**label of the specific axis, and changes
**the name of that label.
*/

PROCEDURE CustLabel IS
  template      og_template;
  axis og_axis;
  labelvarchar2(20);
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_x_axis);
  label := og_get_custlabel(axis);
  og_set_custlabel(axis, 'Employee Number');
END;
```

軸の方向プロパティ

説明

増分値または項目を軸に沿ってどの方向に設定するかを指定します。このプロパティには、次のビルトイン定数を指定できます。

- OG_Down_Direction**
- OG_Left_Direction**
- OG_Right_Direction**
- OG_Up_Direction**

構文

```
PROCEDURE OG_Set_Direction
  (axis OG_Axis,
   direction NUMBER);

FUNCTION OG_Get_Direction
  (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>direction</i>	増分値や連続した項目を軸に沿ってどの方向に設定するかを指定します。

軸の方向プロパティの例

```

/*The following procedure reads the
**directions of the x and y axis and sets
**them to the opposite directions.
*/

PROCEDURE GenDirection IS
  template    /*The following procedure reads the tick
**position of the x-axis, and sets it to
**a different value.og_axis;
  y_axis      og_axis;
  num         number;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);
  y_axis := og_get_axis(template, og_y1_axis);
  num := og_get_direction(x_axis);
  if num = og_left_direction then
    og_set_direction(x_axis, og_right_direction);
  elsif num = og_right_direction then
    og_set_direction(x_axis, og_left_direction);
  end if;
  num := og_get_direction(y_axis);
  if num = og_up_direction then
    og_set_direction(y_axis, og_down_direction);
  elsif num = og_down_direction then
    og_set_direction(y_axis, og_up_direction);
  end if;
END;
```

主目盛格子プロパティ

説明

主目盛きざみに格子を表示するかどうかを指定します。

構文

```

PROCEDURE OG_Set_Majorgrid
  (axis      OG_Axis,
   majorgrid BOOLEAN);
```

```
FUNCTION OG_Get_Majorgrid
  (axis OG_Axis)
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>majorgrid</i>	それぞれの主目盛きざみにグリッドを表示するかどうかを指定します。

主目盛格子プロパティの例

```
/*The following procedure checks if the
**Major Grid checkbox is checked. If the
**box is checked, it unchecks it, and vice
**versa.
*/

PROCEDURE GenMajorGrids IS
  template /*The following procedure reads the tick
**position of the x-axis, and sets it to
**a different value.og_axis;
  val boolean;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);

  val := og_get_majorgrid(x_axis);
  if val = true then
    og_set_majorgrid(x_axis, false);
  else
    og_set_majorgrid(x_axis, true);
  end if;
END;
```

主目盛きざみプロパティ

説明

主目盛きざみを主目盛間隔で表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Majorticks
  (axis OG_Axis,
  majorticks BOOLEAN);

FUNCTION OG_Get_Majorticks
  (axis OG_Axis)
```

```
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>majorticks</i>	主目盛きざみを主目盛間隔で表示するかどうかを指定します。

主目盛きざみプロパティの例

```
/*The following procedure checks if the
**Major Ticks checkbox is checked. If
**the box is checked, it unchecks it,
**and vice versa.
*/

PROCEDURE GenMajorTicks IS
  template    og_template;
  x_axis      og_axis;
  val         boolean;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);

  val := og_get_majorticks(x_axis);
  if val = true then
    og_set_majorticks(x_axis, false);
  else
    og_set_majorticks(x_axis, true);
  end if;
END;
```

補助目盛格子プロパティ

説明

それぞれの補助目盛りきざみに格子を表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Minorgrid
  (axis      OG_Axis,
   minorgrid BOOLEAN);

FUNCTION OG_Get_Minorgrid
  (axis OG_Axis)
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>minorgrid</i>	それぞれの補助目盛きざみに格子を表示するかどうかを指定します。

補助目盛格子プロパティの例

```
/*The following procedure checks if
**the Minor Grid checkbox is checked.
**If the box is checked, it unchecks it,
**and vice versa.
*/

PROCEDURE GenMinorGrids IS
  template      og_template;
  x_axis        og_axis;
  val           boolean;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);

  val := og_get_minorgrid(x_axis);
  if val = true then
    og_set_minorgrid(x_axis, false);
  else
    og_set_minorgrid(x_axis, true);
  end if;
END;
```

補助目盛きざみプロパティ

説明

補助目盛きざみを表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Minorticks
  (axis      OG_Axis,
   minorticks BOOLEAN);

FUNCTION OG_Get_Minorticks
  (axis OG_Axis)
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>minorticks</i>	補助目盛きざみを表示するかどうかを指定します。

補助目盛きざみプロパティの例

```

/*The following procedure checks if the
**Minor Ticks checkbox is checked. If
**the box is checked, it unchecks it, and
**vice versa.
*/

PROCEDURE GenMinorTicks IS
  template      og_template;
  x_axis        og_axis;
  val           boolean;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);

  val := og_get_minorticks(x_axis);
  if val = true then
    og_set_minorticks(x_axis, false);
  else
    og_set_minorticks(x_axis, true);
  end if;
END;

```

補助目盛りきざみ数プロパティ

説明

主目盛きざみの間に定義される補助目盛きざみ数です。

構文

```

PROCEDURE OG_Set_Minorct
  (axis      OG_Axis,
   minorct   NUMBER);

FUNCTION OG_Get_Minorct
  (axis OG_Axis)
RETURN NUMBER;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>minorct</i>	それぞれの主目盛きざみと主目盛きざみの間に定義される補助目盛きざみ数です。

補助目盛りきざみ数プロパティの例

```

/*The following procedure reads the number

```

```

**of minor ticks per interval and resets
**the value to triple the original value.
*/

```

```

PROCEDURE GenMinorCt IS
  template   og_template;
  x_axis     og_axis;
  num number;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);
  num := og_get_ticklabelrot(x_axis);
  og_set_minorct(x_axis, 3*num);
END;

```

位置プロパティ

説明

どのチャート枠に沿って軸を表示するかを指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Bottom_Position

OG_Left_Position

OG_Right_Position

OG_Top_Position

構文

```

PROCEDURE OG_Set_Position
  (axis OG_Axis,
   position NUMBER);

FUNCTION OG_Get_Position
  (axis OG_Axis)
RETURN NUMBER;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>position</i>	どのチャート枠に沿って軸を表示するかを指定します。

位置プロパティの例

```

/*The following procedure determines
**which edge of the chart the axis

```

```
**appears on, and resets the axis to
**the opposite edge.
*/

PROCEDURE GenPosition IS
    template    og_template;
    axis og_axis;
    num number;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_x_axis);
    num := og_get_position(axis);
    if num = og_bottom_position then
        og_set_position(axis, og_top_position);
    elsif num = og_left_position then
        og_set_position(axis, og_right_position);
    elsif num = og_right_position then
        og_set_position(axis, og_left_position);
    elsif num = og_top_position then
        og_set_position(axis, og_bottom_position);
    end if;
END;
*/
```

目盛きざみラベル回転プロパティ

説明

目盛きざみラベルが回転する方向を指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Ccw_Rotation 逆時計回りに回転します。

OG_Cw_Rotation 時計回りに回転します。

OG_No_Rotation 回転しません。

構文

```
PROCEDURE OG_Set_Ticklabelrot
    (axis    OG_Axis,
     ticklabelrot NUMBER);

FUNCTION OG_Get_Ticklabelrot
    (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>ticklabelrot</i>	目盛きざみラベルが回転する方向を指定します。

目盛きざみラベル回転プロパティの例

```
/*The following procedure reads the
**tick label rotation and changes it
**to a different value.
*/

PROCEDURE GenTickLbl IS
  template      og_template;
  x_axis        og_axis;
  num PROCEDURE GenTickLbl IS
    template og_template;
    x_axis og_axis;
    num number;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);

  num := og_get_ticklabelrot(x_axis);
  if num = og_ccw_rotation then
    og_set_ticklabelrot(x_axis, og_cw_rotation);
  elsif num = og_cw_rotation then
    og_set_ticklabelrot(x_axis, og_no_rotation);
  elsif num = og_no_rotation then
    og_set_ticklabelrot(x_axis, og_ccw_rotation);
  end if;
END;
```

目盛きざみラベル・プロパティ

説明

軸に沿って値を識別するラベルを表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Ticklabels
  (axis OG_Axis,
   ticklabels BOOLEAN);

FUNCTION OG_Get_Ticklabels
  (axis OG_Axis)
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>ticklabels</i>	軸に沿って値を識別する項目ラベルを表示するかどうかを指定します。

目盛きざみラベル・プロパティの例

```
/*The following procedure checks if
**Tick Label checkbox is checked.
**If the box is checked, it unchecks it,
**and vice versa.
*/

PROCEDURE GenTickLbl IS
    template    og_template;
    x_axis      og_axis;
    val         boolean;
BEGIN
    template := og_get_template('template0');
    x_axis := og_get_axis(template, og_x_axis);
    val := og_get_ticklabels(x_axis);
    if val = true then
        og_set_ticklabels(x_axis, false);
    else
        og_set_ticklabels(x_axis, true);
    end if;
END;
```

目盛きざみ位置プロパティ

説明

主目盛きざみと補助目盛きざみをどのように表示するかを指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Cross_Tickpos

OG_Inside_Tickpos

OG_Outside_Tickpos

構文

```
PROCEDURE OG_Set_Tickpos
    (axis OG_Axis,
     tickpos NUMBER);
```

```
FUNCTION OG_Get_Tickpos
  (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

axis	軸オブジェクトのハンドルです。
tickpos	主目盛きざみと補助目盛きざみをどう表示するかを指定します。

目盛きざみ位置プロパティの例

```
/*The following procedre reads the tick
**position of the x-axis, and sets it to
**a different value.
*/

PROCEDURE GenTickPos IS
  template    og_template;
  x_axis      og_axis;
  num         number;
BEGIN
  template := og_get_template('template0');
  x_axis := og_get_axis(template, og_x_axis);
  num := og_get_tickpos(x_axis);
  if num = og_cross_tickpos then
    og_set_tickpos(x_axis, og_inside_tickpos);
  elsif num = og_inside_tickpos then
    og_set_tickpos(x_axis, og_outside_tickpos);
  elsif num = og_outside_tickpos then
    og_set_tickpos(x_axis, og_cross_tickpos);
  end if;
END;
```

軸（項目）プロパティ

- 自動最大値プロパティ
- 自動最小値プロパティ
- 日付書式プロパティ
- 最大項目数プロパティ
- 最小項目数プロパティ
- 数値書式プロパティ

自動最大値プロパティ

説明

軸に表示する最大項目数を「自動」に設定するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Disc_Automax
(axis
OG_Axis,
automax
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>automax</i>	軸に表示する最大項目数を「自動」に設定するかどうかを指定します。
<i>maxcat</i>	軸に表示する最大項目数を指定します（ただし、 <i>automax</i> がFALSEの場合のみ）。

自動最大値プロパティの例

```
/*The following procedure checks if the
**X-axis's maximum is set to auto.If
**true, it resets the value to false with
**default_maxcat; if false, it reads the
**current value and resets it to true.
*/

PROCEDURE datemax IS
```

```
template    og_template;
axis og_axis;
val  If
**true, it resets the value to false with
**default_maxcat; if false, it reads the
**current value and resets
**it to true.number;
default_maxcat    number := 3;
BEGIN
template := og_get_template('template0');
axis := og_get_axis(template, og_x_axis);
val := og_get_disc_automax(axis);
if val = true then
og_set_disc_automax(axis, false, default_maxcat);
elsif val = false then
maxcat := og_get_disc_maxcat(axis);
og_set_disc_automax(axis,true,default_maxcat);
end if;
END;
```

自動最小値プロパティ

説明

軸に表示する最小項目数を「自動」に設定するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Disc_Automin
(axis    OG_Axis,
 automin BOOLEAN,
 mincat  NUMBER);

FUNCTION OG_Get_Disc_Automin
(axis OG_Axis)
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>automin</i>	軸に表示する最小項目数を「自動」に設定するかどうかを指定します。
<i>mincat</i>	軸に表示する最小項目数を指定します（ただし、 <i>automin</i> がFALSEの場合のみ）。

自動最小値プロパティの例

```
/*The following procedure checks if the
**X-axis's minimum is set to auto.If
**true, it resets the value to false with
```



```

*default_mincat; if false, it reads the
**current value and resets the value to
**true.
*/

PROCEDURE datemin IS
    template    og_template;
    axis og_axis;
    val    number;
    default_mincat    number;
    default_mincat    number := 50;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_x_axis);
    val := og_get_disc_automin(axis);
    if val = true then
        og_set_disc_automin(axis,false,default_mincat);
    elsif val = false then
        mincat := og_get_disc_mincat(axis);
        og_set_disc_automin(axis,true,default_mincat);
    end if;
END;

```

日付書式プロパティ

説明

軸目盛きざみラベルの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```

PROCEDURE OG_Set_Disc_Datefmt
    (axis    OG_Axis,
     date_fmt VARCHAR2);

FUNCTION OG_Get_Disc_Datefmt
    (axis OG_Axis)
RETURN VARCHAR2;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>date_fmt</i>	軸目盛きざみラベルの日付書式を指定します。

日付書式プロパティの例

```

/*The following procedure reads the current

```

```

**date format of the axis.If the current
**format is not equal to variable
**'default_date', it resets the value to
**'default_date.'
*/

PROCEDURE datefmt IS
    template    og_template;
    axis og_axis;
    val  varchar2(10);
    default_date    varchar2(10) := 'DD_YY_MM';
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_x_axis);
    val := og_get_disc_datefmt(axis);
    if val != default_date then
        og_set_disc_datefmt(axis, default_date);
    end if;
END;
```

最大項目数プロパティ

説明

軸に表示する最大項目数を指定します（ただし、*automax*がFALSEの場合のみ）。

構文

（前述のOG_Set_Disc_Automaxを参照してください。）

```
FUNCTION OG_Get_Disc_Maxcat
    (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
-------------	-----------------

最大項目数プロパティの例

```

/*
** The following procedure checks if the
**X-axis's maximum is set to auto.If
**true, it resets the value to false with
**default_maxcat; if false, it reads the
**current value and resets
**it to true.
*/
```

```

PROCEDURE datemax IS
    template    og_template;
    axis og_axis;
    val boolean;
    maxcat      number;
    default_maxcat    number := 3;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_x_axis);
    val := og_get_disc_automax(axis);
    if val = true then
        og_set_disc_automax(axis, false, default_maxcat);
    elsif val = false then
        maxcat := og_get_disc_maxcat(axis);
        og_set_disc_automax(axis, true, default_maxcat);
    end if;
END;

```

最小項目数プロパティ

説明

軸に表示する最小項目数を指定します（ただし、*automin*がFALSEの場合のみ）。

構文

（前述のOG_Set_Disc_Autominを参照してください。）

```

FUNCTION OG_Get_Disc_Mincat
    (axis OG_Axis)
RETURN NUMBER;

```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
-------------	-----------------

最小項目数プロパティの例

```

/*The following procedure checks if the
**X-axis's minimum is set to auto.If
**true, it resets the value to false with
**default_mincat; if false, it reads the
**current value and resets the value to
**true.
*/

```

```

PROCEDURE datemin IS

```

```
template    og_template;
axis og_axis;
val  boolean;
mincat      number;
default_mincat      number := 50;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_x_axis);
  val := og_get_disc_automin(axis);
  if val = true then
    og_set_disc_automin(axis,false,default_mincat);
  elsif val = false then
    mincat := og_get_disc_mincat(axis);
    og_set_disc_automin(axis,true,default_mincat);
  end if;
END;
```

数値書式プロパティ

説明

軸目盛りざみラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Disc_Numfmt
  (axis    OG_Axis,
   num_fmt  VARCHAR2);

FUNCTION OG_Get_Disc_Numfmt
  (axis OG_Axis)
RETURN VARCHAR2;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>num_fmt</i>	軸目盛りざみラベルの数値書式を指定します。

数値書式プロパティの例

```
/*The following procedure reads the current
**number format of the axis.If the current
**format is not equal to variable
**'default_format', it resets the value to
**'default_format'
*/
```

```
PROCEDURE discnumfmt IS
  template    og_template;
  axis og_axis;
  val         varchar2(10);
  default_format  varchar2(10);
  default_format varchar2(10) := '9,9,9,9';
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_x_axis);
  val := og_get_disc_numfmt(axis);
  if val != default_format then
    og_set_disc_numfmt(axis, default_format);
  end if;
END;
```

軸（値）プロパティ

- 自動最大値プロパティ
- 自動最小値プロパティ
- 自動主目盛間隔プロパティ
- 最大値プロパティ
- 最小値プロパティ
- 数値書式プロパティ
- パーセント元プロパティ
- パーセント基準プロパティ
- スケール・プロパティ
- ステップ・プロパティ

自動最大値プロパティ

説明

軸最大値を「自動」に設定するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Cont_Automax
  (axis      OG_Axis,
   automax   BOOLEAN,
   maximun   NUMBER);

FUNCTION OG_Get_Cont_Automax
  (axis OG_Axis)
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>automax</i>	軸最大値を「自動」に設定するかどうかを指定します。
<i>maximun</i>	最大軸値を指定します（ただし、 <i>automax</i> がFALSEの場合のみ）。

自動最大値プロパティの例

```
/*The following procedure checks if axis
**Y1's maximum is set to auto.If return
**value is TRUE, resets the value to FALSE
**with default_max; if return value is
**FALSE, it resets the value to TRUE
**after reading the specified maximum
**axis value.
*/

PROCEDURE automin IS
    axis og_axis;
    template    og_template;
    val  boolean;
    num  number;
    default_max number := 3000;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_y1_axis);
    val := og_get_cont_autostep(axis);
    if val = TRUE then
        og_set_cont_autostep(axis, FALSE, default_max);
    else
        num := og_get_cont_step(axis);
        og_set_cont_autostep(axis, TRUE, default_max);
    end if;
END;
```

自動最小値プロパティ

説明

軸最小値を「自動」に設定するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Cont_Automin
(axis      OG_Axis,
 automin  BOOLEAN,
 minimun  NUMBER);

FUNCTION OG_Get_Cont_Automin
(axis OG_Axis)
RETURN BOOLEAN;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>automin</i>	軸最小値を「自動」に設定するかどうかを指定します。

<i>minimum</i>	最小軸値を指定します（ただし、 <i>automin</i> がFALSEの場合のみ）。
----------------	--

自動最小値プロパティの例

```

/*The following procedure checks if axis
**Y1's minimum is set to auto.If the
**value is TRUE, it resets the value to
**FALSE with default_min; if the return
**value is FALSE, it resets the value to
**TRUE after reading the specified minimum
**axis value.
*/

PROCEDURE automin IS
  axis og_axis;
  template og_template;
  val boolean;
  num number;
  default_min number := 500;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val := og_get_cont_automin(axis);
  if val = TRUE then
    og_set_cont_automin(axis, FALSE, default_min);
  elsif val = FALSE then
    num := og_get_cont_minimum(axis);
    og_set_cont_automin(axis, TRUE, default_min);
  end if;
END;
```

自動主目盛間隔プロパティ

説明

主目盛間隔を「自動」に設定するかどうかを指定します。

構文

```

PROCEDURE OG_Set_Cont_Autostep
  (axis OG_Axis,
   autostep BOOLEAN,
   step NUMBER);

FUNCTION OG_Get_Cont_Autostep
  (axis OG_Axis)
RETURN BOOLEAN;
```


パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>autostep</i>	主目盛間隔を「自動」に設定するかどうかを指定します。
<i>step</i>	主目盛間隔を指定します（ただし、 <i>autostep</i> がFALSEの場合のみ）。

自動主目盛間隔プロパティの例

```
/*The following procedure checks if axis
**Y1's step is set to auto.If the return
**value is TRUE, it resets the value to
**FALSE with default step value; if
**return value is FALSE,it resets
**the value to TRUE after reading
**the specified step value.
*/

PROCEDURE autostep IS
    axis og_axis;
    template og_template;
    val boolean;
    num number;
    step number := 500;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_y1_axis);
    val := og_get_cont_autostep(axis);
    if val = TRUE then
        og_set_cont_autostep(axis, FALSE, step);
    else
        num := og_get_cont_step(axis);
        og_set_cont_autostep(axis, TRUE, step);
    end if;
END;
```

最大値プロパティ

説明

最大軸値を指定します（ただし、「自動最大値」がFALSEの場合のみ）。

構文

（前述のOG_Set_Cont_Automaxを参照してください。）

```
FUNCTION OG_Get_Cont_Maximum
    (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
-------------	-----------------

最大値プロパティの例

```
/*The following procedure checks if axis
**Y1's maximum is set to auto.If return
**value is TRUE,it resets the value to
**FALSE with default_max; if return value
**is FALSE, it resets the value to
**TRUE after reading the specified
**maximum axis value.
*/

PROCEDURE automin IS
    axis og_axis;
    template    og_template;
    val  boolean;
    num  number;
    default_max number := 3000;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_y1_axis);
    val := og_get_cont_autostep(axis);
    if val = TRUE then
        og_set_cont_autostep(axis, FALSE, default_max);
    else
        num := og_get_cont_step(axis);
        og_set_cont_autostep(axis, TRUE, default_max);
    end if;
END;
```

最小値プロパティ

説明

最小軸値を指定します（ただし、「自動最小値」がFALSEの場合のみ）。

構文

（前述のOG_Set_Cont_Autominを参照してください。）

```
FUNCTION OG_Get_Cont_Minimum
    (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
-------------	-----------------

最小値プロパティの例

```
/*The following procedure checks if axis
**Y1's minimum is set to auto.If the
**return value is TRUE, it resets the
**value to FALSE with default_min;
**if the return value is FALSE,it resets
**the value to TRUE after reading the
**specified minimum axis value.
*/

PROCEDURE automin IS
    axis og_axis;
    template    og_template;
    val    boolean;
    num    number;
    default_min number := 500;
BEGIN
    template := og_get_template('template0');
    axis := og_get_axis(template, og_y1_axis);
    val := og_get_cont_automin(axis);
    if val = TRUE then
        og_set_cont_automin(axis, FALSE, default_min);
    elsif val = FALSE then
        num := og_get_cont_minimum(axis);
        og_set_cont_automin(axis, TRUE, default_min);
    end if;
END;
```

数値書式プロパティ

説明

軸目盛きざみラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Cont_Numfmt
    (axis    OG_Axis,
     num_fmt  VARCHAR2);

FUNCTION OG_Get_Cont_Numfmt
    (axis OG_Axis)
RETURN VARCHAR2;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>num_fmt</i>	軸目盛きざみラベルの数値書式を指定します。

数値書式プロパティの例

```
/*The following procedure reads the current
**number format of the axis and resets it to
**a different value.
*/

PROCEDURE numFormat IS
  axis og_axis;
  template og_template;
  val varchar2(10);
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val := og_get_cont_numfmt(axis);
  og_set_cont_numfmt(axis, '9,9,9,9,9');
END;
```

パーセント元プロパティ

説明

「パーセント基準」値をどのように計算するかを指定します。このプロパティには、次のビルトイン定数を指定できます。

- OG_Category_Pctby** 各データ値のパーセンテージを、他の項目の同じフィールドのデータ値と相対的に計算します。
- OG_Field_Pctby** 各データ値のパーセンテージを、他のフィールドの同じ項目のデータ値と相対的に計算します。

構文

```
PROCEDURE OG_Set_Pct_By
  (axis OG_Axis,
   pct_of NUMBER);

FUNCTION OG_Get_Pct_By
  (axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>pct_of</i>	「パーセント基準」値をどのように計算するかを指定します。

パーセント元プロパティの例

```

*/The following procedure reads the
**calculating method for the
**Percent Of scaling values
**(with Scale is set to OG_PCT_SCALE)
**from the axis and resets the value to
**the next available value.
/*

PROCEDURE pctby IS
  axis og_axis;
  template   og_template;
  val  number;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val := og_get_pct_by(axis);
  if val = OG_category_pctby then
    og_set_pct_by(axis, og_field_pctby);
  elsif val = og_field_pctby then
    og_set_pct_by(axis, og_category_pctby);
  end if;
END;
```

パーセント基準プロパティ

説明

相対スケール変更係数を指定します（ただし、「スケール」がOG_Pct_Scaleに設定されている場合のみ）。このプロパティには、次のビルトイン定数を指定できます。

- OG_Maximum_Pctof** 各データ値を最大データ値に対するパーセンテージで表示します。
- OG_Minimum_Pctof** 各データ値を最小データ値に対するパーセンテージで表示します。
- OG_Sum_Pctof** 各データ値を全データ値の合計に対するパーセンテージで表示します。

構文

```

PROCEDURE OG_Set_Pct_Of
  (axis   OG_Axis,
   pct_of NUMBER);

FUNCTION OG_Get_Pct_Of
```

```
(axis OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>pct_of</i>	相対スケール変数係数を指定します(ただし、「スケール」がOG_Pct_Scaleに設定されている場合のみ)。

パーセント基準プロパティの例

```
/*The following procedure reads the
**relative scaling factor (with Scale
**set to OG_PCT_SCALE)from the axis
**and resets the value to the next
**available value.
*/

PROCEDURE pctof IS
  axis og_axis;
  template   og_template;
  val  number;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val := og_get_pct_of(axis);
  if val = OG_maximum_pctof then
    og_set_pct_of(axis, og_minimum_pctof);
  elsif val = og_minimum_pctof then
    og_set_pct_of(axis, og_sum_pctof);
  elsif val = og_sum_pctof then
    og_set_pct_of(axis, og_maximum_pctof);
  end if;
END;
```

スケール・プロパティ

説明

軸のスケール設定に使用するアルゴリズムを指定します。このプロパティには、次のビルトイン定数を指定できます。

- OG_Linear_Scale** 最小軸値と最大軸値の間の固定間隔を使用して軸をスケール設定します。
- OG_LOG_Scale** 最小軸値と最大軸値の間の間隔を判別するために対数アルゴリズム(基数は 10のべき) を使用してスケール設定します。
- OG_Pct_Scale** 「パーセント基準」で指定した割合に対して相対的にデータ値を表示するように軸をスケール設定します。

構文

```
PROCEDURE OG_Set_Scale
  (axis  OG_Axis,
   scale NUMBER);

FUNCTION OG_Get_Scale
  (axis  OG_Axis)
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
<i>scale</i>	軸のスケール設定に使用されるアルゴリズムを指定します。

スケール・プロパティの例

```
/*The following procedure reads
**the method used for scaling from
**the axis and resets the value
**to the next available value. PROCEDURE scale IS
*/

PROCEDURE scale IS
  axis og_axis;
  template  og_template;
  val  template := og_get_template('template0'); axis :=
og_get_axis(template, og_y1_axis);
number;
BEGIN
  template := og_get_template('template0');
  axis := og_get_axis(template, og_y1_axis);
  val := og_get_scale(axis);
  if val = OG_linear_scale then
    og_set_scale(axis, og_log_scale);
  elsif val = og_log_scale then
    og_set_scale(axis, og_pct_scale);
  elsif val = og_pct_scale then
    og_set_scale(axis, og_linear_scale);
  end if;
END;
```

ステップ・プロパティ

説明

主目盛間隔を指定します（ただし、「自動ステップ」がFALSEの場合のみ）。

構文

(前述のOG_Set_Cont_Autostepを参照してください。)

```
FUNCTION OG_Get_Cont_Step  
  (axis OG_Axis)  
RETURN NUMBER;
```

パラメータ

<i>axis</i>	軸オブジェクトのハンドルです。
-------------	-----------------

ステップ・プロパティの例

```
/*The following procedure checks if axis  
**Y1's step is set to auto. If the return  
**value is TRUE, it resets the value to  
**FALSE with default step value; if return  
**value is FALSE, it resets the value to  
**TRUE after reading the specified step value.  
*/
```

```
PROCEDURE autostep IS  
  axis og_axis;  
  template og_template;  
  val boolean;  
  num number;  
  step number := 500;  
BEGIN  
  template := og_get_template('template0');  
  axis := og_get_axis(template, og_y1_axis);  
  val := og_get_cont_autostep(axis);  
  if val = TRUE then  
    og_set_cont_autostep(axis, FALSE, step);  
  else  
    num := og_get_cont_step(axis);  
    og_set_cont_autostep(axis, TRUE, step);  
  end if;  
END;
```


チャート要素プロパティ

- ボタン・プロシージャ・プロパティ
- イベント・プロパティ
- 展開プロパティ
- 名前プロパティ

ボタン・プロシージャ・プロパティ

説明

このチャート要素と対応付けるボタン・プロシージャのハンドルです。このプロシージャが必要なマウス・イベントを受け取るには、イベント・プロパティが正しく設定されている必要があることに注意してください。イベント・プロパティには、次のビルトイン定数を指定できます。

OG_No_Events

OG_Mouse_Up

OG_Mouse_Down

OG_Mouse_Move_Down

プロシージャが複数のイベント・タイプを受け取ることができるようにするには、イベントを該当のイベントの定数の合計に設定します。

構文

```
PROCEDURE OG_Set_Button
(
  chart      OG_Object,
  row_num    NUMBER,
  col_name   VARCHAR2,
  button_proc OG_Buttonproc,
  events     NUMBER);
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>row_num</i>	チャート要素で表す問合せ行番号です。
<i>col_name</i>	チャート要素で表す問合せ列です。
<i>button_proc</i>	このチャート要素と対応付けるボタン・プロシージャのハンドルです。
<i>events</i>	ボタン・プロシージャが受け取るマウス・イベントのタイプです。

ボタン・プロシージャ・プロパティの例

```
/*The following procedure assigns
**a button procedure tochart
**element MGR_bars.
*/

PROCEDURE AssignButtonProc IS
    chart og_object;
    mgrbar og_object;
    button og_buttonproc;
BEGIN
    chart := og_get_object('chart');
    mgrbar := og_get_object('MGR_bars');
    button := og_get_buttonproc('button');
    og_set_button(chart, og_get_row(mgrbar), 'MGR', button, og_mouse_down);
END;
```

イベント・プロパティ

説明

ボタン・プロシージャが受け取るマウス・イベントのタイプです。このプロパティには、次のビルトイン定数を指定できます。

OG_No_Events

OG_Mouse_Up

OG_Mouse_Down

OG_Mouse_Move_Down

プロシージャが複数のイベント・タイプを受け取ることができるようにするには、イベントを該当のイベントの定数の合計に設定します。

構文

(OG_Set_Buttonを参照してください。)

パラメータ

なし

イベント・プロパティの例

```
/*The following procedure assigns
```

```

**a button procedure tochart
**element MGR_bars.
*/

PROCEDURE AssignButtonProc IS
    chart og_object;
    mgrbar og_object;
    button og_buttonproc;
BEGIN
    chart := og_get_object('chart');
    mgrbar := og_get_object('MGR_bars');
    button := og_get_buttonproc('button');
    og_set_button(chart, og_get_row(mgrbar), 'MGR', button, og_mouse_down);
END;
```

展開プロパティ

説明

チャート要素（つまり、円グラフ）を展開する距離、つまりチャートのx半径とy半径のパーセンテージ（たとえば、25など）です。このプロパティは、円グラフで使用したときのみ適用されます。また、チャート要素として項目を指定するとすべての円グラフが同じ割合で展開されます。したがって、指定する列名は、項目列ではなく、値列で指定する必要があります。

構文

```

PROCEDURE OG_Set_Explosion
    (chart      OG_Object,
     row_num    NUMBER,
     col_name   VARCHAR2,
     explode_pct NUMBER);
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>row_num</i>	チャート要素で表す問合せ行番号です。
<i>col_name</i>	チャート要素で表す問合せ列です。これは値列の名前で指定する必要があります。
<i>explode_pct</i>	チャート要素（つまり、円グラフ）を展開する距離、つまりチャートのx半径とy半径のパーセンテージ（たとえば、25など）です。

展開プロパティの例

```

/*The following procedure assigns the
**distance the chartelement should be
** exploded to to 50.
*/
```

```
PROCEDURE Explosion IS
    pie og_object;
    mgr_slice og_object;
BEGIN
    pie := og_get_object('pie');
    mgr_slice := og_get_object('MGR_slices');
    og_set_explosion(pie, og_get_row(mgr_slice), 'MGR', 50);
END;
```

名前プロパティ

説明

チャート要素の名前です。

構文

```
PROCEDURE OG_Set_Name
(
    chart      OG_Object,
    row_num    NUMBER,
    col_name   VARCHAR2,
    name       VARCHAR2);
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>row_num</i>	チャート要素で表す問合せ行番号です。
<i>col_name</i>	チャート要素で表す問合せ列です。
<i>name</i>	チャート要素の名前です。

名前プロパティの例

```
/*The following procedure sets
**the name of the chart element.
*/

PROCEDURE Name IS
    chart og_object;
    mgr_bar og_object;
BEGIN
    chart := og_get_object('chart');
    mgr_bar := og_get_object('MgrBars');
    og_set_name(chart, og_get_row(mgr_bar), 'MGR', 'NewName');
END;
```


チャート・プロパティ

- 自動更新プロパティ
- 終了行プロパティ
- フィルタ・プロパティ
- 問合せプロパティ
- データ範囲プロパティ
- サイズと位置プロパティ
- 開始行プロパティ
- テンプレート・プロパティ
- タイトル・プロパティ

自動更新プロパティ

説明

問合せの実行時にチャートを自動的に更新するように指定します。

構文

```
PROCEDURE OG_Set_Autoupdate
  (chart OG_Object,
   autoupdate BOOLEAN);

FUNCTION OG_Get_Autoupdate
  (chart OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>autoupdate</i>	問合せの実行時にチャートを自動的に更新するように指定します。

自動更新プロパティの例

```
/*The following reads the value of
**autoupdate in Chartproperties, and
**resets the value to its opposite value
```

```

*/

PROCEDURE ChartAutoUpdate IS
    chart og_object;
    autoupdate boolean;
BEGIN
    chart := og_get_object('chart');
    autoupdate := og_get_autoupdate(chart);
    if autoupdate = true then
        og_set_autoupdate(chart, false);
    else
        og_set_autoupdate(chart, true);
    end if;
END;

```

終了行プロパティ

説明

チャートに表示するデータの最後の行です。

構文

(OG_Set_Rowを参照してください。)

```

FUNCTION OG_Get_Endrow
    (chart OG_Object)
RETURN NUMBER;

```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
--------------	---------------------

終了行プロパティの例

```

/*The following procedure reads the
**startrow and endrowvalue from chart
**(provided the Plot rows box is checked),
**and resets the range to startrow -1 and
**endrow -1.)
*/

```

```

PROCEDURE ChartStartEnd IS
    chart og_object;
    startrow number;
    endrow number;
BEGIN

```

```
chart := og_get_object('chart');
startrow := og_get_startrow(chart);
endrow := og_get_endrow(chart);
og_set_rows(chart,true, startrow-1, endrow-1);
END;
```

フィルタ・プロパティ

説明

問合せのフィルタ・トリガー・プロシージャの名前です。

構文

```
PROCEDURE OG_Set_Filter
  (chart  OG_Object,
   filter VARCHAR2);

FUNCTION OG_Get_Filter
  (chart  OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>filter</i>	問合せのフィルタ・トリガー・プロシージャの名前です。

フィルタ・プロパティの例

```
/*The following procedure reads
**the name of the current filter trigger
**of the chart, and assigns a different
**filter trigger to the chart
*/

PROCEDURE ChartFilter IS
  chart og_object;
  current_filter varchar2(30);
  new_filter varchar2(30):='MyFilter';
BEGIN
  chart := og_get_object('chart');
  current_filter := og_get_filter(chart);
  og_set_filter(chart, new_filter);
END;
```


問合せプロパティ

説明

チャートに使用する問合せのハンドルです。

構文

```
PROCEDURE OG_Set_Query
  (chart      OG_Object,
   query      OG_Query,
   damage     BOOLEAN   := TRUE,
   update_bbox BOOLEAN   := TRUE);

FUNCTION OG_Get_Query
  (chart OG_Object)
RETURN OG_Query;
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>query</i>	チャートに使用する問合せのハンドルです。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

問合せプロパティの例

```
/*The following procedure reads the
**query handle from the current chart
**(qry0)and resets the handle value to
**qry1.
*/
```

```
PROCEDURE ChartQuery IS
  chart og_object;
  qry0 og_query;
  qry1 og_query;
BEGIN
  chart := og_get_object('chart');
  qry0 := og_get_query(chart);
  qry1 := og_get_query('query1');
  og_set_query(chart, qry1);
END;
```

データ範囲プロパティ

説明

チャートに表示する問合せ行の数を、*startrow*と*endrow*で指定した範囲に制限するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Rows
  (chart      OG_Object,
   rangeflag  BOOLEAN,
   startrow   NUMBER,
   endrow     NUMBER);

FUNCTION OG_Get_Rangeflag
  (chart OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>rangeflag</i>	チャートに表示する問合せ行の数を、 <i>startrow</i> と <i>endrow</i> で指定した範囲に制限するかどうかを指定します。
<i>startrow</i>	チャートに表示するデータの最初の行です。最初のデータ行は0、2番目の行は1となり、以下同様です。
<i>endrow</i>	チャートに表示するデータの最後の行です。

データ範囲プロパティの例

```
/*The following procedure checks if
**the number of queryrows that appear
**on the chart is range restricted.
**If true,it resets the value to false
**(i.e. plots all rows); if false, it
**resets the value to true with a
**restricted range specified by
**startrow and endrow.
*/

PROCEDURE ChartRange IS
  chart og_object;
  rangeflag boolean;
  startrow number := 3;
  endrow number := 9;
BEGIN
  chart := og_get_object('chart');
  rangeflag := og_get_rangeflag(chart);
  if rangeflag = true then
```

```

        og_set_rows(chart,false, startrow, endrow);
    else
        og_set_rows(chart, true, startrow, endrow);
    end if;
END;
```

サイズと位置プロパティ

説明

チャートの枠のx座標とy座標、および高さ、幅です（レイアウト単位）。

構文

```

PROCEDURE OG_Set_Frame
    (chart      OG_Object,
     frame      OG_Rectangle,
     damage     BOOLEAN      := TRUE,
     update_bbox BOOLEAN      := TRUE);

FUNCTION OG_Get_Frame
    (chart OG_Object)
RETURN OG_Rectangle;
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>frame</i>	チャートの枠のx座標とy座標、および高さ、幅です（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

サイズと位置プロパティの例

```

/*The following procedure reads the frame
**size of the chart, and reduces it by half.
*/
```

```

PROCEDURE SizeAndPos IS
    chart og_object;
    rect og_rectangle;
BEGIN
    chart := og_get_object('chart');
    rect := og_get_frame(chart);
    rect.x := rect.x/2;
    rect.y := rect.y/2;
    rect.height := rect.height/2;
    rect.width := rect.width/2;
    og_set_frame(chart, rect);
```

```
END;
```

開始行プロパティ

説明

チャートに表示するデータの最初の行です。最初のデータ行は0、2番目の行は1という順で番号付けされています。

構文

(前述のOG_Set_Rowsを参照してください。)

```
FUNCTION OG_Get_Startrow  
    (chart OG_Object)  
RETURN NUMBER;
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
--------------	---------------------

開始行プロパティの例

```
/*The following procedure reads the  
**startrow and endrowvalue from chart  
**(provided the Plot rows box is checked),  
**and resets the range to startrow -1 and  
**endrow -1.)  
*/  
  
PROCEDURE ChartStartEnd IS  
    chart og_object;  
    startrow number;  
    endrow number;  
BEGIN  
    chart := og_get_object('chart');  
    startrow := og_get_startrow(chart);  
    endrow := og_get_endrow(chart);  
    og_set_rows(chart,true, startrow-1, endrow-1);  
END;
```

テンプレート・プロパティ

説明

チャートに使用するテンプレートのハンドルです。

構文

```
PROCEDURE OG_Set_Template
  (chart      OG_Object,
   template   OG_Template,
   damage     BOOLEAN      := TRUE,
   update_bbox BOOLEAN      := TRUE);

FUNCTION OG_Get_Template
  (chart OG_Object)
RETURN OG_Template;
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>template</i>	チャートに使用するテンプレートのハンドルです。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

テンプレート・プロパティの例

```
/*The following procedure reads the
**template handles template1 and template2
**from chart1 and chart2 respectively, and
**assigns template1 to chart2, template2
**to chart1.
*/
```

```
PROCEDURE ChartTemplate IS
  chart1 og_object;
  chart2 og_object;
  template1 og_template;
  template2 og_template;
BEGIN
  chart1 := og_get_object('chart1');
  chart2 := og_get_object('chart2');
  template1 := og_get_template(chart1);
  template2 := og_get_template(chart2);
  og_set_template(chart1, template2);
  og_set_template(chart2, template1);
END;
```

タイトル・プロパティ

説明

チャートのタイトルです。

構文

```
PROCEDURE OG_Set_Title
    (chart  OG_Object,
     title  VARCHAR2);

FUNCTION OG_Get_Title
    (chart  OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>chart</i>	チャート・オブジェクトのハンドルです。
<i>title</i>	チャートのタイトルです。

タイトル・プロパティの例

```
/*The following procedure reads
**the title of a chart;compare
**the value with new_title.If
**they are not equal,change the
**title to new_title.
*/

PROCEDURE ChartTitle IS
    chart og_object;
    title varchar2(30);
    new_title varchar2(30) := 'New title';
BEGIN
    chart := og_get_object('chart');
    title := og_get_title(chart);
    if title != new_title then
        og_set_title(chart, new_title);
    end if;
END;
```

複数テキスト・プロパティ

単一テキスト・カウント・プロパティ
複数テキスト・カウント・プロパティ

単一テキスト・カウント・プロパティ

説明

複数テキスト要素を構成する単一テキスト要素の数です。

構文

```
FUNCTION OG_Get_Stcount
    (text OG_Object,
     cmptext_index NUMBER)
RETURN NUMBER;
```

パラメータ

<i>text</i>	記述するテキスト要素です。
<i>cmptext_index</i>	定義中の複数テキスト要素の索引番号。

単一テキスト・カウント・プロパティの例

```
*/The following procedure reads the count of
**simple text of the first compound
**text in a text object, and prints the count
**back to the text object.
*/

PROCEDURE simpleText IS
num number;
text og_object;
BEGIN
text := og_get_object('text');
num := og_get_stcount(text,0);
og_set_str(text, num);
END;
```

図表プロパティ

- クローズ・トリガー・プロパティ
- 日付書式プロパティ
- 高さプロパティ
- オープン・トリガー・プロパティ
- 幅プロパティ

クローズ・トリガー・プロパティ

説明

図表のクローズ・トリガーの名前です。

構文

```
PROCEDURE OG_Set_Closetrigger
    (trigger VARCHAR2);

FUNCTION OG_Get_Closetrigger
    RETURN VARCHAR2;
```

パラメータ

<i>trigger</i>	図表のクローズ・トリガーの名前です。
----------------	--------------------

クローズ・トリガー・プロパティの例

```
/*The following procedure reads the name
**of the closetrigger of the current
**display.If the current trigger is not
**new_trigger, it sets new_trigger to be the
**current close trigger procedure.
*/

PROCEDURE CloseTrigger IS
    val varchar2(20);
    new_trigger varchar2(20) := 'CURSORDEFAULT';
BEGIN
    val := og_get_closetrigger;
    if val != new_trigger then
        og_set_closetrigger('CursorDefault');
    end if;
```



```
END;
```

日付書式プロパティ

説明

パラメータの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
OG_Set_Dateformat  
    (dateformat VARCHAR2);  
  
OG_Get_Dateformat  
RETURN VARCHAR2;
```

パラメータ

<i>dateformat</i>	パラメータの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。
-------------------	--

日付書式プロパティの例

```
/*The following procedure reads the date  
**format of display.If the format is not  
**the same as new_datefmt, it sets the current  
**format to new_format.  
*/  
  
PROCEDURE datefmt IS  
    datefmt varchar2(20);  
    new_datefmt varchar2(20) := 'DD/MM/YYYY';  
BEGIN  
    datefmt := og_get_dateformat;  
    if datefmt != new_datefmt then  
        og_set_dateformat('DD/MM/YYYY');  
    end if;  
END;
```

高さプロパティ

説明

レイアウトの高さです（レイアウト単位）。

構文

(OG_Set_Display_Sizeを参照してください)。

```
FUNCTION OG_Get_Display_Height  
RETURN NUMBER;
```

パラメータ

なし

高さプロパティの例

```
/*The following procedure reads the width  
**and height of thecurrent display and  
**reduces the display size by half.  
*/
```

```
PROCEDURE dimension0 IS  
  width number;  
  height number;  
BEGIN  
  width := og_get_display_width;  
  height := og_get_display_height;  
  og_set_display_size(width/2, height/2);  
END;
```

オープン・トリガー・プロパティ

説明

図表のオープン・トリガーの名前です。

構文

```
PROCEDURE OG_Set_Opentrigger  
  (trigger VARCHAR2);  
  
FUNCTION OG_Get_Opentrigger  
RETURN VARCHAR2;
```

パラメータ

<i>trigger</i>	図表のオープン・トリガーの名前です。
----------------	--------------------

オープン・トリガー・プロパティの例

```
/*The following procedure reads the name of  
**the opentrigger of the current display.
```

```

**If the current trigger is not new_trigger,
**it sets new_trigger to be the current open
**trigger procedure.
*/

PROCEDURE OpenTrigger IS
  val varchar2(20);
  new_trigger varchar2(20) := 'TOBLUE';
BEGIN
  val := og_get_openttrigger;
  if val != 'TOBLUE' then
    og_set_openttrigger('tobblue');
  end if;
END;
```

幅プロパティ

説明

レイアウトの幅です（レイアウト単位）。

構文

```

PROCEDURE OG_Set_Display_Size
  (width  NUMBER,
   height NUMBER);

FUNCTION OG_Get_Display_Width
RETURN NUMBER;
```

パラメータ

<i>width</i>	レイアウトの幅です（レイアウト単位）。
<i>height</i>	レイアウトの高さです（レイアウト単位）。

幅プロパティの例

```

/*The following procedure reads the width
**and height of thecurrent display and
**reduces the display size by half.
*/

PROCEDURE dimension0 IS
  width number;
  height number;
BEGIN
  width := og_get_display_width;
  height := og_get_display_height;
```

```
    og_set_display_size(width/2, height/2);  
END;
```

枠（軸チャート）プロパティ

- 基本線の軸プロパティ
- 基本線の値プロパティ
- 項目幅プロパティ
- カスタム日付書式プロパティ
- カスタム数値書式プロパティ
- 参照線カウント・プロパティ
- 第2-Y軸プロパティ

基本線軸プロパティ

説明

基本線の値が対象とする軸を指定します。このプロパティには、次のビルトイン定数を指定できます。

- OG_Template
- OG_Y1_Axis
- OG_Y2_Axis

構文

```
PROCEDURE OG_Set_Baseaxis
  (template OG_Template,
   baseaxis NUMBER);

FUNCTION OG_Get_Baseaxis
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>baseaxis</i>	位置を求めるために基本線の値が比較される軸を指定します。

基本線軸プロパティの例

```
*/The following procedure specifies the
```

```

**date format for the baseline label.
*/

PROCEDURE CusDateFmt IS
  chart og_object;
  template og_template;
  custDate date;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  custDate := og_get_cust_date(template);
  if custDate != '06-DEC-88' then
    og_set_cust_date(template, '06-DEC-96');
  end if;
END;
```

基本線の値プロパティ

説明

フィールドを表示する際に開始点として使用する値です。このプロパティには、次のビルトイン定数を指定できます。

- OG_Custom_Baseline
- OG_Min_Baseline
- OG_Zero_Baseline

構文

```

PROCEDURE OG_Set_Basevalue
  (template OG_Template,
   basevalue NUMBER);

FUNCTION OG_Get_Basevalue
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>basevalue</i>	値軸に沿ってフィールドを表示する際に開始点として使用される値です。

基本線の値プロパティの例

```

/*The following procedure reads
**the baseline value of the field
```

```

**template of a chart.If the current
** baseline value is ZERO,
**the procedure resets the value to
**MAX; If the current baseline value
**is any value other than ZERO, the
**procedure resets the value to ZERO.
*/

PROCEDURE BaseLine IS
  chart og_object;
  template og_template;
  value number;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  value := og_get_basevalue(template);
  if value = og_zero_baseline then
    og_set_basevalue(template, og_max_baseline);
  else
    og_set_basevalue(template, og_zero_baseline);
  end if;
  og_update_chart(chart);
END;
```

項目幅プロパティ

説明

これは、「ストライプ幅」の割合で示した、横棒グラフや縦棒グラフ内の棒の幅です。ストライプ幅は、棒が互いに重なることなくとれる棒の最大の幅で、項目軸の長さを、表示される棒の数で割って決定されます。

構文

```

PROCEDURE OG_Set_Catwidth
  (template OG_Template,
   catwidth NUMBER);

FUNCTION OG_Get_Catwidth
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>catwidth</i>	これは、「ストライプ幅」の割合で示した、横棒グラフや縦棒グラフ内の棒の幅です。ストライプ幅は、棒が互いに重なることなくとれる棒の最大の幅で、項目軸の長さを、表示される棒の数で割って決定されます。

項目幅プロパティの例

```
/* The following procedure reduces the
** category width of the bars by half of
** its original width.
*/

PROCEDURE CatWidth IS
  chart og_object;
  template og_template;
  width number;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  width := og_get_catwidth(template);
  og_set_catwidth(template, width/2);
END;
```

カスタム日付書式プロパティ

説明

日付軸タイプの基本線の値を指定します。この値は、データを表示する際の起点として使用されます。このオプションは、OG_CUSTOM_BASELINEプロパティが選択されている場合のみ有効です。

構文

```
PROCEDURE OG_Set_Cust_Date
(template OG_Template,
 cust_date DATE);

FUNCTION OG_Get_Cust_Date
(template OG_Template)
RETURN DATE;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>cust_date</i>	日付軸タイプの基本線の値を指定します。この値は、値の軸に沿ってデータ・ポイントを描画するための参照として使用されます。

カスタム日付書式プロパティの例

```
/*The following procedure specifies
**the date format for the baseline label.
*/
```



```

PROCEDURE CusDateFmt IS
  chart og_object;
  template og_template;
  custDate date;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  custDate := og_get_cust_date(template);
  if custDate != '06-DEC-88' then
    og_set_cust_date(template, '06-DEC-96');
  end if;
END;
```

カスタム数値書式プロパティ

説明

数値軸タイプの基本線の値を指定します。この値は、自動的にOG_CUSTOM_BASELINEに設定されます。

構文

```

PROCEDURE OG_Set_Cust_Num
  (template OG_Template,
   cust_num NUMBER);

FUNCTION OG_Get_Cust_Num
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>cust_num</i>	数値軸タイプの基本線の値を指定します。この値は、値の軸に沿ってデータ・ポイントを描画するための参照として使用されます。

カスタム数値書式プロパティの例

```

/*The following procedure specifies
**the number format for the baseline label.
*/

PROCEDURE CusNumFmt IS
  chart og_object;
  template og_template;
  num number;
BEGIN
  chart := og_get_object('chart');
```

```
template := og_get_template(chart);
num := og_get_cust_num(template);
og_set_cust_num(template, num/2);
END;
```

参照線カウント・プロパティ

説明

チャート・テンプレートに属する参照線の数です。

構文

```
FUNCTION OG_Get_Reflinect
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

参照線カウント・プロパティの例

```
/*The following procedure reads the
**reference line countand prints the
**number to a text object.
*/

PROCEDURE RefLineCnt IS
  text og_object;
  chart og_object;
  template og_template;
  cnt number;
BEGIN
  text := og_get_object('text object');
  chart := og_get_object('chart');
  template := og_get_template(chart);
  cnt := og_get_reflirect(template);
  og_set_str(text, cnt);
END;
```

第2-Y軸プロパティ

説明

第2-Y軸をチャート内に表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Second_Y
  (template OG_Template,
   second_y BOOLEAN);

FUNCTION OG_Get_Second_Y
  (template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>second_y</i>	第2-Y軸をチャート内に表示するかどうかを指定します。

第2-Y軸プロパティの例

```
/* The following procedure determines if
**a second Y axisappears on the chart.
**If not, it adds a second one.
*/

PROCEDURE SecondY IS
  chart og_object;
  template og_template;
  axis boolean;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  axis := og_get_second_y(template);
  if axis = false then
    og_set_second_y(template, true);
  end if;
  og_update_chart(chart);
END;
```

枠（一般）プロパティ

深さサイズ・プロパティ

フィールド・テンプレート・カウント・プロパティ

枠タイプ・プロパティ

凡例プロパティ

凡例中の列数プロパティ

名前プロパティ

表示枠プロパティ

ルート・プロパティ

シャドウ方向プロパティ

シャドウ・サイズ・プロパティ

深さサイズ・プロパティ

説明

チャート枠と要素を3次元で描画するときの深さのサイズを指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_None_Depthsize

OG_Small_Depthsize

OG_Medium_Depthsize

OG_Large_None_Depthsize

OG_Xlarge_Depthsize

構文

```
PROCEDURE OG_Set_Depthsize
  (template  OG_Template,
   depthsize NUMBER);

FUNCTION OG_Get_Depthsize
```

```
(template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>depthsize</i>	チャート枠と要素を3次元表示にするために、それらが描画される深さのサイズを指定します。

深さサイズ・プロパティの例

```
PROCEDURE FrameDepth IS
  chart og_object;
  template og_template;
  depth number;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  depth := og_get_depthsize(template);
  if depth = og_none_depthsize then
    og_set_depthsize(template, og_small_depthsize);
  elsif depth = og_small_depthsize then
    og_set_depthsize(template, og_medium_depthsize);
  elsif depth = og_medium_depthsize then
    og_set_depthsize(template, og_large_depthsize);
  elsif depth = og_large_depthsize then
    og_set_depthsize(template, og_xlarge_depthsize);
  elsif depth = og_xlarge_depthsize then
    og_set_depthsize(template, og_none_depthsize);
  end if;
END;
```

フィールド・テンプレート・カウント・プロパティ

説明

チャート・テンプレートに属するフィールド・テンプレートの数です。

構文

```
FUNCTION OG_Get_Ftempct
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

フィールド・テンプレート・カウント・プロパティの例

```
/*The following procedure reads the number of
**the field template that belongs to the current
**template,and prints the value to a text object.
*/

PROCEDURE FTempCnt IS
  text og_object;
  chart og_object;
  template og_template;
  num number;
BEGIN
  text := og_get_object('text object');
  chart := og_get_object('chart');
  template := og_get_template(chart);
  num := og_get_ftempct(template);
  og_set_str(text, num);
END;
```

枠タイプ・プロパティ

説明

このテンプレートで表すチャートのタイプです。 このプロパティには、次のビルトイン定数を指定できます。

OG_Axis_Frametype

OG_Pie_Frametype

OG_Table_Frametype

構文

```
FUNCTION OG_Get_Frametype
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

枠タイプ・プロパティの例

```
/*The following reads the frame type
**and prints the value to a text object.
*/
```

```
PROCEDURE FrameType IS
  text og_object;
  chart og_object;
  template og_template;
  num number;
BEGIN
  text := og_get_object('text object');
  chart := og_get_object('chart');
  template := og_get_template(chart);
  num := og_get_frametype(template);
  if num = og_axis_frametype then
    og_set_str(text, 'axis');
  elsif num = og_pie_frametype then
    og_set_str(text, 'pie');
  elsif num = og_table_frametype then
    og_set_str(text, 'table');
  end if;
END;
```

凡例プロパティ

説明

チャートの凡例を表示するかどうかを指定します。（表チャートには適用されません。）

構文

```
PROCEDURE OG_Set_Legend
  (template OG_Template,
   show      BOOLEAN);

FUNCTION OG_Get_Legend
  (template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
表示	チャートの凡例を表示するかどうかを指定します。（表チャートには適用されません。）

凡例プロパティの例

```
/*The following procedure determines
**if a legend is shown.If a legend
**is shown, it hides it; if a legend
**is hidden, itshows it.
*/
```

```
PROCEDURE FrameLegend IS
  chart og_object;
  template og_template;
  val boolean;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  val := og_get_legend(template);
  if val = true then
    og_set_legend(template, false);
  else
    og_set_legend(template, true);
  end if;
END;
```

凡例中の列数プロパティ

説明

凡例内にラベルを表示するために使用する列数です。

構文

```
PROCEDURE OG_Set_Legendcolct
  (template OG_Template,
   colct     NUMBER);

FUNCTION OG_Get_Legendcolct
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>colct</i>	凡例内にラベルを表示するために使用する列数です。

凡例中の列数プロパティの例

```
/*The following procedure reads the number of
**columns in the legend box.If there is more
**than one column in the box, it changes the
**number of the columns to one; if there is
**one column, it changes the number of columns
**to two.
*/

PROCEDURE FrameLegendCol IS
  chart og_object;
```



```

        template og_template;
        num number;
BEGIN
    chart := og_get_object('chart');
    template := og_get_template(chart);
    num := og_get_legendcolct(template);
    if num > 1 then
        og_set_legendcolct(template, 1);
    else
        og_set_legendcolct(template, 2);
    end if;
END;
```

名前プロパティ

説明

チャート・テンプレートの名前です。

構文

```

PROCEDURE OG_Set_Frame_Name
    (template OG_Template,
     name      VARCHAR2);

FUNCTION OG_Get_Frame_Name
    (template OG_Template)
RETURN VARCHAR2;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>name</i>	チャート・テンプレートの名前です。

名前プロパティの例

```

/*The following reads the frame name.
**If the name is not'templatel', it sets
**it to 'templatel'.
*/
```

```

PROCEDURE FrameName IS
    chart og_object;
    template og_template;
    name varchar2(30);
BEGIN
    chart := og_get_object('chart');
    template := og_get_template(chart);
```

```
name := og_get_frame_name(template);
if name != 'template1' then
  og_set_frame_name(template, 'template1');
end if;
END;
```

表示枠プロパティ

説明

チャートの周囲に四角形を表示するかどうかを指定します。（円グラフ・チャートには適用されません。）

構文

```
PROCEDURE OG_Set_Plotframe
(template OG_Template,
  show      BOOLEAN);

FUNCTION OG_Get_Plotframe
(template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
表示	チャートの周囲に四角形を表示するかどうかを指定します。

表示枠プロパティの例

```
/*The following procedure determines
**whether a plot frame is drawn.If
**true, it removes the plot frame;
**if false, it adds a plot frame to
**the current chart.
*/

PROCEDURE FramePlot IS
  chart og_object;
  template og_template;
  val boolean;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  val := og_get_plotframe(template);
  if val = true then
    og_set_plotframe(template, false);
  else
```

```

        og_set_plotframe(template, true);
    end if;
END;
```

ルート・プロパティ

説明

チャート・テンプレートのハンドル（OG_Template型）です。

構文

```

FUNCTION OG_Get_Root
    (template OG_Template)
RETURN OG_Object;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

ルート・プロパティの例

```

/*The procedure gets the handle
**(root)of the chart object.
*/

PROCEDURE FrameRoot IS
    chart og_object;
    template og_template;
    root og_object;
BEGIN
    chart := og_get_object('chart');
    template := og_get_template(chart);
    root := og_get_root(template);
END;
```

シャドウ方向プロパティ

説明

チャート枠と要素を描画するときのシャドウの方向を指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Upperright_Shadowdir

OG_Upperleft_Shadowdir

OG_Lowerright_Shadowdir

OG_Lowerleft_Shadowdir

構文

```
PROCEDURE OG_Set_Shadowdir
  (template OG_Template,
   shadowdir NUMBER);

FUNCTION OG_Get_Shadowdir
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>shadowdir</i>	チャート枠と要素が描画されるシャドウの方向を指定します。

シャドウ方向プロパティの例

```
PROCEDURE FrameShadowDir IS
  chart og_object;
  template og_template;
  shadow number;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  shadow := og_get_shadowdir(template);
  if shadow = og_upperright_shadowdir then
    og_set_shadowdir(template, og_lowerleft_shadowdir);
  elsif shadow = og_lowerleft_shadowdir then
    og_set_shadowdir(template, og_upperleft_shadowdir);
  elsif shadow = og_upperleft_shadowdir then
    og_set_shadowdir(template, og_lowerright_shadowdir);
  elsif shadow = og_lowerright_shadowdir then
    og_set_shadowdir(template, og_upperright_shadowdir);
  end if;
END;
```

シャドウ・サイズ・プロパティ

説明

チャート枠と要素を描画するときのシャドウのサイズを指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_None_Shadowsize

OG_Small_Shadowsize

OG_Medium_Shadowsize

OG_Large_Shadowsize

OG_Xlarge_Shadowsize

構文

```
PROCEDURE OG_Set_Shadowsize
    (template    OG_Template,
     shadowsize  NUMBER);

FUNCTION OG_Get_Shadowsize
    (template    OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>shadowsize</i>	チャート枠と要素が描画されるシャドウのサイズを指定します。

シャドウ・サイズ・プロパティの例

```
/*The following procedure reads the shadow size
**of the chart, and changes the size to a
**different value.
*/

PROCEDURE FrameShadow IS
    chart og_object;
    template og_template;
    shadow number;
BEGIN
    chart := og_get_object('chart');
    template := og_get_template(chart);
    shadow := og_get_shadowsize(template);
    if shadow = og_none_shadowsize then
        og_set_shadowsize(template, og_small_shadowsize);
    elsif shadow = og_small_shadowsize then
        og_set_shadowsize(template, og_medium_shadowsize);
    elsif shadow = og_medium_shadowsize then
        og_set_shadowsize(template, og_large_shadowsize);
    elsif shadow = og_large_shadowsize then
        og_set_shadowsize(template, og_xlarge_shadowsize);
    elsif shadow = og_xlarge_shadowsize then
        og_set_shadowsize(template, og_none_shadowsize);
    end if;
END;
```

枠（円グラフ）プロパティ

項目ラベル・プロパティ
項目日付書式プロパティ
項目数値書式プロパティ
データ値プロパティ
オーバーラップなしプロパティ
その他プロパティ
パーセント書式プロパティ
パーセント値プロパティ
表示順序プロパティ
目盛きざみプロパティ
表示プロパティ
合計値プロパティ
値書式プロパティ

項目ラベル・プロパティ

説明

円グラフの各円弧に、項目の名前のラベルを付けるかどうかを指定します。

構文

```
PROCEDURE OG_Set_Categs  
    (template OG_Template,  
     categs BOOLEAN);  
  
FUNCTION OG_Get_Categs  
    (template OG_Template)  
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>categs</i>	円グラフの各円弧を、示されている項目の名前でラベル表示するかどうかを指定します。

項目ラベル・プロパティの例

```
/*The following procedure gets
**information about the relationship
**between individual pie slices and
**the complete chart.If the current
**relationship is TOTALVALUE, the
**procedure resets the relationship
** to PERCENTAGE with a value of 50;
**If the current relationship is
**PERCENTAGE, the procedure resets
**the relationship to TOTALVALUE with
**a value of 400000.
*/

PROCEDURE PieUsage IS
  pie og_object;
  template og_template;
  usage number;
  usagevalue number;
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  usage := og_get_usage(template);
  if usage = og_totalvalue_usage then
    usagevalue := og_get_usagevalue(template);
    og_set_usage(template, og_pct_usage, 50);
  elsif usage = og_pct_usage then
    usagevalue := og_get_usagevalue(template);
    og_set_usage(template, og_totalvalue_usage, 400000);
  end if;
  og_update_chart(pie);
END;
```

項目日付書式プロパティ

説明

項目ラベルの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Catdatefmt
  (template  OG_Template,
   catdatefmt VARCHAR2);

FUNCTION OG_Get_Catdatefmt
  (template OG_Template)
RETURN VARCHAR2;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>catdatefmt</i>	項目ラベルの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。

項目日付書式プロパティの例

```
/*The following procedure changes the
**pie slice label's date format if the
**format is not currently
**'DD-MM-YY'.
*/

PROCEDURE CatDateFmt IS
  pie og_object;
  template og_template;
  format varchar2(20);
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  format := og_get_catdatefmt(template);
  if format != 'MM-DD-YY' then
    og_set_catdatefmt(template, 'MM-DD-YY');
  end if;
  og_update_chart(pie);
END;
```

項目数値書式プロパティ

説明

項目ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Catnumfmt
```



```
(template OG_Template,
 catnumfmt VARCHAR2);

FUNCTION OG_Get_Catnumfmt
(template OG_Template)
RETURN VARCHAR2;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>catnumfmt</i>	項目ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。

項目数値書式プロパティの例

```
/*The following procedure changes the
**pie slice label's number format if
**the format is not currently
**'9,9,9,9'.
*/

PROCEDURE CatNumFmt IS
  pie og_object;
  template og_template;
  format varchar2(20);
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  format := og_get_catnumfmt(template);
  if format != '9,9,9,9' then
    og_set_catnumfmt(template, '9,9,9,9');
  end if;
  og_update_chart(pie);
END;
```

データ値プロパティ

説明

円グラフの各円弧にそのデータ値のラベルを付けるかどうかを指定します。

構文

```
PROCEDURE OG_Set_Datavals
(template OG_Template,
 datavals BOOLEAN);

FUNCTION OG_Get_Datavals
(template OG_Template)
```

RETURN BOOLEAN;

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>datavals</i>	円グラフの各円弧を、そのデータ値でラベル表示するかどうかを指定します。

データ値プロパティの例

```
/* The following procedure hides/shows
** the data value for each pie slice.
*/

PROCEDURE DataVals IS
    pie og_object;
    template og_template;
    val boolean;
BEGIN
    pie := og_get_object('pie');
    template := og_get_template(pie);
    val := og_get_datavals(template);
    if val = true then
        og_set_datavals(template, false);
    elsif val = false then
        og_set_datavals(template, true);
    end if;
    og_update_chart(pie);
END;
```

オーバーラップなしプロパティ

説明

円グラフの各円弧のラベルが互いにオーバーラップしないように指定します。

構文

```
PROCEDURE OG_Set_Nooverlap
    (template OG_Template,
     nooverlap BOOLEAN);

FUNCTION OG_Get_Nooverlap
    (template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>nooverlap</i>	円グラフの各円弧のラベルが互いにオーバーラップしないように指定します。

オーバーラップなしプロパティの例

```

/*The following procedure determines if
**pie slice labels are allowed to overlap.
**If overlapping is allowed, the procedure
**disallows it.
*/

```

```

PROCEDURE NoOverlap IS
  pie og_object;
  template og_template;
  val boolean;
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  val := og_get_nooverlap(template);
  if val = false then
    og_set_nooverlap(template, true);
  end if;
  og_update_chart(pie);
END;

```

その他プロパティ

説明

データ値が円グラフの中で個々の円弧として表示されるように、データ値が表すチャートの最小パーセンテージを指定します。この数値より少ないパーセンテージを表すデータ値は、円グラフの中で「その他」として1つの円弧にまとめられます。

構文

```

PROCEDURE OG_Set_Other
  (template OG_Template,
   other NUMBER);

FUNCTION OG_Get_Other
  (template OG_Template)
RETURN NUMBER;

```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

<i>other</i>	データ値が円グラフの中で個々の円弧として表示されるように、データ値が表すチャートの最小パーセンテージを指定します。この数値より少ないパーセンテージを表すデータ値は、円グラフの中で「その他」として1つの円弧にまとめられます。
--------------	---

その他プロパティの例

```
/*The following procedure doubles
**the percentage value for which
**any chart slice with a value
**less than or equal to the
**percentage value will be labeled
**"other."
*/

PROCEDURE Other IS
  pie og_object;
  template og_template;
  num number;
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  num := og_get_other(template);
  og_set_other(template, num*2);
  og_update_chart(pie);
END;
```

パーセント書式プロパティ

説明

パーセント値ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Pctfmt
  (template OG_Template,
   pctfmt   VARCHAR2);

FUNCTION OG_Get_Pctfmt
  (template OG_Template)
RETURN VARCHAR2;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

<i>pctfmt</i>	パーセント値ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。
---------------	--

パーセント書式プロパティの例

```

/*The following procedure hides/shows the
**percent value for each pie slice.
*/

```

```

PROCEDURE PctVals IS
  pie og_object;
  template og_template;
  val boolean;
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  val := og_get_pctvalues(template);
  if val = true then
    og_set_pctvalues(template, false);
  elsif val = false then
    og_set_pctvalues(template, true);
  end if;
  og_update_chart(pie);
END;

```

パーセント値プロパティ

説明

円グラフの各円弧に、チャート全体のパーセンテージのラベルを付けるかどうかを指定します。

構文

```

PROCEDURE OG_Set_Pctvalues
  (template OG_Template,
   pctvalues BOOLEAN);

FUNCTION OG_Get_Pctvalues
  (template OG_Template)
RETURN BOOLEAN;

```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>pctvalues</i>	円グラフの各円弧を、示されている完成したチャートの割合でラベル表示するかどうかを指定します。

パーセント値プロパティの例

```
/* The following procedure hides/shows
**the percent value for each pie slice.
*/

PROCEDURE PctVals IS
  pie og_object;
  template og_template;
  val boolean;
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  val := og_get_pctvalues(template);
  if val = true then
    og_set_pctvalues(template, false);
  elsif val = false then
    og_set_pctvalues(template, true);
  end if;
  og_update_chart(pie);
END;
```

表示順序プロパティ

説明

データ値が表示される方向を指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Ccw_Plotorder 値が逆時計回りの方向で表示されます。

OG_Cw_Plotorder 値が時計回りの方向で表示されます。

構文

```
PROCEDURE OG_Set_Plotorder
  (template OG_Template,
   plotorder NUMBER);

FUNCTION OG_Get_Plotorder
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>plotorder</i>	データ値が表示される方向を指定します。

表示順序プロパティの例

```
/*The following procedure reads the
**direction in which the data values
**are plotted, and reverses the
**plotting direction.
*/

PROCEDURE plotOrder IS
  pie og_object;
  template og_template;
  porder number;
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  porder := og_get_plotorder(template);
  if porder = og_cw_plotorder then
    og_set_plotorder(template, og_ccw_plotorder);
  else
    og_set_plotorder(template, og_cw_plotorder);
  end if;
  og_update_chart(pie);
END;
```

目盛きざみプロパティ

説明

円グラフの各円弧とそのラベルを結ぶ目盛きざみを表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Ticks
  (template OG_Template,
   ticks     BOOLEAN);

FUNCTION OG_Get_Ticks
  (template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>ticks</i>	円グラフの各円弧とそのラベルを結ぶ目盛きざみを表示するかどうかを指定します。

目盛きざみプロパティの例

```
/*The following procedure hides/  
**shows the ticks for each pie  
**slice.  
*/  
  
PROCEDURE ticks IS  
  pie og_object;  
  template og_template;  
  val boolean;  
BEGIN  
  pie := og_get_object('pie');  
  template := og_get_template(pie);  
  val := og_get_ticks(template);  
  if val = true then  
    og_set_ticks(template, false);  
  else  
    og_set_ticks(template, true);  
  end if;  
  og_update_chart(pie);  
END;
```

表示プロパティ

説明

円グラフの各円弧とチャート全体との関係を指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Totalvalue_Usage

OG_Pct_Usage

構文

```
PROCEDURE OG_Set_Usage  
  (template OG_Template,  
   usage     NUMBER,  
   usagevalue NUMBER);  
  
FUNCTION OG_Get_Usage  
  (template OG_Template)  
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

<i>usage</i>	円グラフの各円弧と完成したチャートとの関係を指定します。
<i>usagevalue</i>	円グラフの各円弧は、そのデータ値におけるここで指定した値に対するパーセンテージとして表示されます。（ <i>usage</i> がOG_TOTALVALUE_USAGEに設定されている場合に限り有効です。）

表示プロパティの例

```
/*The following procedure gets
**information about the relationship
**between individual pie slices and
**the complete chart.If the current
**relationship is TOTALVALUE, the procedure
**resets the relationship to PERCENTAGE
**with a value of 50.If the current
**relationship is PERCENTAGE, the procedure
**resets the relationship to TOTALVALUE
**with a value of 400000.
*/

PROCEDURE PieUsage IS
  pie og_object;
  template og_template;
  usage number;
  usagevalue number;
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  usage := og_get_usage(template);
  if usage = og_totalvalue_usage then
    usagevalue := og_get_usagevalue(template);
    og_set_usage(template, og_pct_usage, 50);
  elsif usage = og_pct_usage then
    usagevalue := og_get_usagevalue(template);
    og_set_usage(template, og_totalvalue_usage, 400000);
  end if;
  og_update_chart(pie);
END;
```

合計値プロパティ

説明

円グラフの各円弧は、そのデータ値におけるここで指定した値に対するパーセンテージとして表示されます。

構文

(前述のOG_Set_Usageを参照してください。)

```
FUNCTION OG_Get_Usagevalue  
  (template OG_Template)  
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

合計値プロパティの例

```
/*The following procedure gets  
**information about the relationship  
**between individual pie slices and  
**the complete chart.If the current  
**relationship is TOTALVALUE, the procedure  
**resets the relationship to PERCENTAGE  
**with a value of 50.If the current  
**relationship is PERCENTAGE, the procedure  
**resets the relationship to TOTALVALUE  
**with a value of 400000.  
*/  
  
PROCEDURE PieUsage IS  
  pie og_object;  
  template og_template;  
  usage number;  
  usagevalue number;  
BEGIN  
  pie := og_get_object('pie');  
  template := og_get_template(pie);  
  usage := og_get_usage(template);  
  if usage = og_totalvalue_usage then  
    usagevalue := og_get_usagevalue(template);  
    og_set_usage(template, og_pct_usage, 50);  
  elsif usage = og_pct_usage then  
    usagevalue := og_get_usagevalue(template);  
    og_set_usage(template, og_totalvalue_usage, 400000);  
  end if;  
  og_update_chart(pie);  
END;
```

値書式プロパティ

説明

データ値ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Valuefmt
  (template      OG_Template,
   valuenumfmt   VARCHAR2);

FUNCTION OG_Get_Valuefmt
  (template      OG_Template)
RETURN VARCHAR2;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>valuenumfmt</i>	データ値ラベルの数値書式を指定します。

値書式プロパティの例

```
/*The following procedure changes the pie
**slice label's value format if the format
**is not currently '0999'.
*/

PROCEDURE ValFmt IS
  pie og_object;
  template og_template;
  format varchar2(20);
BEGIN
  pie := og_get_object('pie');
  template := og_get_template(pie);
  format := og_get_valuefmt(template);
  if format != '0999' then
    og_set_valuefmt(template, '0999');
  end if;
  og_update_chart(pie);
END;
```

枠（表チャート）プロパティ

- 自動最大値プロパティ
- 自動最小値プロパティ
- 列名プロパティ
- 格子カウント・プロパティ
- 水平格子プロパティ
- 最大行数プロパティ
- 最小行数プロパティ
- 垂直格子プロパティ

自動最大値プロパティ

説明

チャートに表示される最大行数を「自動」に設定するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Automax
  (template OG_Template,
   automax  BOOLEAN,
   maxrows  NUMBER);

FUNCTION OG_Get_Automax
  (template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>automax</i>	チャートに表示される最大行数を「自動」に設定するかどうかを指定します。
<i>maxrows</i>	チャートに表示される最大行数を指定します(ただし、 <i>automax</i> がFALSEの場合のみ)。

自動最大値プロパティの例

```
/*The following procedure determines if
**there is a maximum number of rows to
```

```

**be displayed in the table or if the
**number of rows is automatically
**determined.If the number of
**rows is not automatically determined,
**the procedure reads the number of rows
**the table displays currently and resets
**it to be automatically determined.
*/

PROCEDURE AutoMax IS
  table1 og_object;
  template og_template;
  val boolean;
  maxrows number := 2;
BEGIN
  table1 := og_get_object('table');
  template := og_get_template(table1);
  val := og_get_automax(template);
  if val = false then
    minrows := og_get_maxrows(template);
    og_set_automax(template, false, maxrows/2);
  end if;
  og_update_chart(table1);
END;
```

自動最小値プロパティ

説明

チャートに表示される最小行数を「自動」に設定するかどうかを指定します。

構文

```

PROCEDURE OG_Set_Automin
  (template OG_Template,
   automin  BOOLEAN,
   minrows  NUMBER);

FUNCTION OG_Get_Automin
  (template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>automin</i>	チャートに表示される最小行数を「自動」に設定するかどうかを指定します。
<i>minrows</i>	チャートに表示される最小行数を指定します（ただし、 <i>automin</i> がFALSEの場合のみ）。

自動最小値プロパティの例

```
/*"The following procedure
**determines if there is a
**minimum number of rows that
**must be displayed in the
**table or whether the number of
**rows is automatically determined.
**If the number of rows is not
**automatically determined, the procedure
**reads the number of rows the table
**currently displays and resets it to
**be automatically determined.
*/

PROCEDURE AutoMax IS
    table1 og_object;
    template og_template;
    val boolean;
    minrows number := 2;
BEGIN
    table1 := og_get_object('table');
    template := og_get_template(table1);
    val := og_get_automin(template);
    if val = false then
        minrows := og_get_minrows(template);
        og_set_automin(template, false, minrows/2);
    end if;
    og_update_chart(table1);
END;
```

列名プロパティ

説明

列の名前をチャート内の最初の行として表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Cname
    (template OG_Template,
     cname     BOOLEAN);

FUNCTION OG_Get_Cname
    (template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>cname</i>	列の名前をチャート内の最初の行として表示するかどうかを指定します。

列名プロパティの例

```

/*The following procedure hides/shows the
**table's column names.
*/

PROCEDURE ColNames IS
  table1 og_object;
  template og_template;
  val boolean;
BEGIN
  table1 := og_get_object('table');
  template := og_get_template(table1);
  val := og_get_cname(template);
  if val = true then
    og_set_cname(template, false);
  elsif val = false then
    og_set_cname(template, true);
  end if;
  og_update_chart(table1);
END;
```

格子カウント・プロパティ

説明

各水平格子間に表示されるデータの行数です（ただし、「水平格子」をTRUEに設定した場合のみ）。

構文

```

PROCEDURE OG_Set_Gridct
  (template OG_Template,
   gridct   NUMBER);

FUNCTION OG_Get_Gridct
  (template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>gridct</i>	各水平格子間に表示されるデータの行数です（ただし、「水平格子」をTRUEに設定した場合のみ）。

格子カウント・プロパティの例

```
/*The following procedure doubles
**the grid count of thetable.
*/

PROCEDURE gridcnt IS
  table1 og_object;
  template og_template;
  cnt number;
BEGIN
  table1 := og_get_object('table');
  template := og_get_template(table1);
  cnt := og_get_gridct(template);
  og_set_gridct(template, cnt*2);
  og_update_chart(table1);
END;
```

水平格子プロパティ

説明

行と行の間に水平格子を表示するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Hgrid
  (template OG_Template,
   hgrid     BOOLEAN);

FUNCTION OG_Get_Hgrid
  (template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>hgrid</i>	行と行の間に水平グリッドを表示するかどうかを指定します。

水平格子プロパティの例

```
/*The following procedure hides/shows
**horizontal gridlines.
*/

PROCEDURE HoriGrid IS
  table1 og_object;
```



```

template og_template;
val boolean;
BEGIN
table1 := og_get_object('table');
template := og_get_template(table1);
val := og_get_hgrid(template);
if val = true then
    og_set_hgrid(template, false);
elsif val = false then
    og_set_hgrid(template, true);
end if;
og_update_chart(table1);
END;
```

最大行数プロパティ

説明

チャートに表示する最大行数を指定します（「自動最大値」がFALSEの場合のみ）。

構文

（前述のOG_Set_Automaxを参照してください。）

```

FUNCTION OG_Get_Maxrows
(template OG_Template)
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

最大行数プロパティの例

```

/*The following procedure determines
**if there is a maximum number of rows to
**be displayed in the table or if the
**number of rows is automatically
**determined.If the number of rows is
**automatically determined, the procedure
**reads the number of rows the table
**displays currently and resets it to be
**automatically determined.
*/

PROCEDURE AutoMax IS
table1 og_object;
template og_template;
```

```
val boolean;  
maxrows number := 2;  
BEGIN  
  table1 := og_get_object('table');  
  template := og_get_template(table1);  
  val := og_get_automax(template);  
  if val = false then  
    maxrows := og_get_maxrows(template);  
    og_set_automax(template, false, maxrows/2);  
  end if;  
  og_update_chart(table1);  
END;
```

最小行数プロパティ

説明

チャートに表示する最小行数を指定します（「自動最小値」がFALSEの場合のみ）。

構文

（前述のOG_Set_Autominを参照してください。）

```
FUNCTION OG_Get_Minrows  
  (template OG_Template)  
RETURN NUMBER;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
-----------------	----------------

最小行数プロパティの例

```
/*"The following procedure  
**determines if there is a  
**minimum number of rows that  
**must be displayed in the  
**table or whether the number of  
**rows is automatically determined.  
**If the number of rows is not  
**automatically determined, the procedure  
**reads the number of rows the table  
**currently displays and resets it to  
**be automatically determined.  
**/  
  
PROCEDURE AutoMax IS  
  table1 og_object;
```

```

template og_template;
val boolean;
minrows number := 2;
BEGIN
table1 := og_get_object('table');
template := og_get_template(table1);
val := og_get_automin(template);
if val = false then
    minrows := og_get_minrows(template);
    og_set_automin(template, false, minrows/2);
end if;
og_update_chart(table1);
END;
```

垂直格子プロパティ

説明

列と列の間に垂直格子を表示するかどうかを指定します。

構文

```

PROCEDURE OG_Set_Vgrid
(
    template OG_Template,
    vgrid     BOOLEAN);

FUNCTION OG_Get_Vgrid
(
    template OG_Template)
RETURN BOOLEAN;
```

パラメータ

<i>template</i>	チャート・テンプレートです。
<i>vgrid</i>	列と列の間に垂直グリッドを表示するかどうかを指定します。

垂直格子プロパティの例

```

/* The following procedure hides/shows
**vertical gridlines.
*/

PROCEDURE VertGrid IS
table1 og_object;
template og_template;
val boolean;
BEGIN
```

```
table1 := og_get_object('table');  
template := og_get_template(table1);  
val := og_get_vgrid(template);  
if val = true then  
  og_set_vgrid(template, false);  
elsif val = false then  
  og_set_vgrid(template, true);  
end if;  
og_update_chart(table1);  
END.
```

フィールド・テンプレート（一般）プロパティ

- カラー回転プロパティ
- 日付書式プロパティ
- 名前プロパティ
- 数値書式プロパティ
- ルート・プロパティ

カラー回転プロパティ

説明

Graphics Builderによって、カラー・パレットやパターン・パレットを自動的に選択させ、このフィールド・テンプレートを使用する各フィールドに他と重複しないようにパレットを割り当てるかどうかを指定します。このプロパティには、次のビルトイン定数を指定できます。

- OG_None_Colorrot
- OG_Auto_Colorrot
- OG_Color_Colorrot
- OG_Pattern_Colorrot
- OG_Both_Colorrot

構文

```
PROCEDURE OG_Set_Colorrot
  (ftemp      OG_Ftemp,
   colorrot   NUMBER);

FUNCTION OG_Get_Colorrot
  (ftemp      OG_Ftemp)
RETURN NUMBER;
```

パラメータ

ftemp	フィールド・テンプレートのハンドルです。
colorrot	Graphics Builderがカラー・パレットやパターン・パレットを自動的に回転させて、このフィールド・テンプレートを使用する各フィールドに固有のパレット設定を選択するかどうかを指定します。

カラー回転プロパティの例

```
/*The following procedure reads if any
**color rotation is applied to the chart.
**If none has been applied, it applies
**AUTO color rotation.If another method
**of color rotation is currently applied,
**it changes the rotation to NONE.
*/

PROCEDURE fieldColRot IS
    ftemp og_ftemp;
    color number;
BEGIN
    ftemp := og_get_ftemp(og_get_template(og_get_object('chart')),0);
    color := og_get_colorrot(ftemp);
    if color = og_none_colorrot then
        og_set_colorrot(ftemp, og_auto_colorrot);
    else
        og_set_colorrot(ftemp, og_none_colorrot);
    end if;
    og_update_chart(og_get_object('chart'));
END;
```

日付書式プロパティ

説明

フィールド・ラベルの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Datefmt
    (ftemp      OG_Ftemp,
     date_fmt   VARCHAR2);

FUNCTION OG_Get_Datefmt
    (ftemp      OG_Ftemp)
RETURN VARCHAR2;
```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>date_fmt</i>	フィールド・ラベルの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。

日付書式プロパティの例

```

    */The following procedure
    **determines if label number
    **formats are all '9,9,9,9'.
    **If not, it changes them all
    **to '9,9,9,9'.
    */

PROCEDURE fieldDateFmt IS
    ftemp og_ftemp;
    datefmt varchar2(20);
BEGIN
    ftemp := og_get_ftemp(og_get_template(og_get_object('chart'),0);
    datefmt := og_get_datefmt(ftemp);
    if datefmt != 'DD-MM-YYYY' then
        og_set_datefmt(ftemp, 'DD-MM-YYYY');
    end if;
END;

```

名前プロパティ

説明

フィールド・テンプレートの名前です。

構文

```

PROCEDURE OG_Set_Ftemp_Name
    (ftemp OG_Ftemp,
     name  VARCHAR2);

FUNCTION OG_Get_Ftemp_Name
    (ftemp OG_Ftemp)
RETURN VARCHAR2;

```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>name</i>	フィールド・テンプレートの名前です。

名前プロパティの例

```

/*The following button procedure
**appends a '1' tothe current
**field template's name.
*/

```

```
PROCEDURE fieldname IS
    ftemp og_ftemp;
    chart og_object;
    name varchar2(20);
BEGIN
    ftemp := og_get_ftemp(og_get_template(og_get_object('chart')),0);
    name := og_get_ftemp_name(ftemp);
    og_set_ftemp_name(ftemp, name||'1');
END;
```

数値書式プロパティ

説明

フィールド・ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

構文

```
PROCEDURE OG_Set_Numfmt
    (ftemp    OG_Ftemp,
     num_fmt  VARCHAR2);

FUNCTION OG_Get_Numfmt
    (ftemp    OG_Ftemp)
RETURN VARCHAR2;
```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>num_fmt</i>	フィールド・ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。

数値書式プロパティの例

```
*/The following procedure
**determines if the labels'number
**format is '9,9,9,9'.If
**not, it changes the format
**to '9,9,9,9'.
*/

PROCEDURE fieldNumFmt IS
    ftemp og_ftemp;
    numfmt varchar2(20);
BEGIN
    ftemp := og_get_ftemp(og_get_template(og_get_object('chart')),0);
    numfmt := og_get_numfmt(ftemp);
    if numfmt != '9,9,9,9' then
        og_set_numfmt(ftemp, '9,9,9,9');
```



```
    end if;  
END;
```

ルート・プロパティ

説明

フィールド・テンプレートが属するチャート・テンプレートのハンドルです。

構文

```
FUNCTION OG_Get_Root  
    (ftemp OG_Ftemp)  
RETURN OG_Object;
```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
--------------	----------------------

ルート・プロパティの例

```
/*The following procedure gets  
**a chart's field template handles.  
*/  
  
PROCEDURE fieldname IS  
    ftemp og_ftemp;  
    root og_object;  
BEGIN  
    ftemp := og_get_ftemp(og_get_template(og_get_object('chart')),0);  
    root := og_get_root(ftemp);  
END;
```

フィールド・テンプレート（軸チャート）プロパティ

- 軸プロパティ
- カーブ調整プロパティ
- ラベル回転プロパティ
- 線の形式プロパティ
- オーバーラップ・プロパティ
- 表示位置プロパティ
- 表示形式プロパティ

軸プロパティ

説明

フィールドの値が対象とする軸を指定します。このプロパティには、次のビルトイン定数を指定できます。

- OG_Y1_Axis**
- OG_Y2_Axis**

構文

```
PROCEDURE OG_Set_Axis
  (ftemp OG_Ftemp,
   axis  NUMBER);

FUNCTION OG_Get_Axis
  (ftemp OG_Ftemp)
RETURN NUMBER;
```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>axis</i>	フィールドの表示方法を決定するのに必要なデータ値が比較される軸を指定します。

軸プロパティの例

```
/*The following procedure
```

```
**rotates the main Y axis the
**chart currently refers to
**(if there is more than one
**Y axis)and switches the main
**Y axis to a different Y axis.
*/

PROCEDURE axis IS
  axis number;
  ftemp og_ftemp;
  chart og_object;
BEGIN
  chart := og_get_object('chart');
  ftemp := og_get_ftemp(og_get_template(chart),0);
  axis := og_get_axis(ftemp);
  if axis = og_y1_axis then
    og_set_axis(ftemp, og_y2_axis);
  elsif axis = og_y2_axis then
    og_set_axis(ftemp, og_y1_axis);
  end if;
  og_update_chart(chart);
END;
```

カーブ調整プロパティ

説明

カーブ調整をチャートに適用するかどうか、また適用する場合にはどのアルゴリズムを使用するかを指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_No_Curvefit

OG_Linear_Curvefit

OG_LOG_Curvefit

OG_Exp_Curvefit

OG_Power_Curvefit

構文

```
PROCEDURE OG_Set_Curvefit
  (ftemp      OG_Ftemp,
   curvefit   NUMBER);

FUNCTION OG_Get_Curvefit
  (ftemp      OG_Ftemp)
```

RETURN NUMBER;

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>curvefit</i>	カーブ調整をチャートに適用するかどうか、また適用する場合にはどのアルゴリズムを使用するかを指定します。

カーブ調整プロパティの例

```
/*The following procedure determines
**if a curve fit is applied to the chart.
**If not, it applies a Linear CurveFit
**to the chart.If a curve fit is currently
**applied to the chart, it removes it.
*/

PROCEDURE CurveFit IS
  curve number;
  ftemp og_ftemp;
  chart og_object;
BEGIN
  chart := og_get_object('chart');
  ftemp := og_get_ftemp(og_get_template(chart),0);
  curve := og_get_curvefit(ftemp);
  if curve = og_no_curvefit then
    og_set_curvefit(ftemp,og_linear_curvefit);
  else
    og_set_curvefit(ftemp,og_no_curvefit);
  end if;
  og_update_chart(chart);
END;
```

ラベル回転プロパティ

説明

ラベル表示形式を含む各フィールドのラベルの回転角度を指定します。このプロパティには、次のビルトイン定数を指定できます。

- OG_Ccw_Rotation** 逆時計回りに回転します。
- OG_Cw_Rotation** 時計回りに回転します。
- OG_No_Rotation**

構文

```

PROCEDURE OG_Set_Labelrot
  (ftemp  OG_Ftemp,
   linestyle NUMBER);

FUNCTION OG_Get_Labelrot
  (ftemp  OG_Ftemp)
RETURN NUMBER;

```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>linesty</i>	ラベル表示形式を含む各フィールドのラベルの回転角度を指定します。

ラベル回転プロパティの例

```

/* The following procedure rotates a
**chart's rotation labels.
*/

PROCEDURE lblrot IS
  rot number;
  ftemp og_ftemp;
  chart og_object;
BEGIN
  chart := og_get_object('chart');
  ftemp := og_get_ftemp(og_get_template(chart),0);
  rot := og_get_labelrot(ftemp);
  if rot = og_no_rotation then
    og_set_labelrot(ftemp,og_cw_rotation);
  elsif rot = og_cw_rotation then
    og_set_labelrot(ftemp, og_ccw_rotation);
  elsif rot = og_ccw_rotation then
    og_set_labelrot(ftemp, og_no_rotation);
  end if;
  og_update_chart(chart);
END;

```

線の形式プロパティ

説明

線表示形式を含むフィールドのデータ・ポイント間を結ぶために使用する線の形式を指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Spline_Linestyle

OG_Step_Linestyle

OG_Straight_Linestyle

構文

```
PROCEDURE OG_Set_Linesty
  (ftemp    OG_Ftemp,
   linestyle NUMBER);

FUNCTION OG_Get_Linesty
  (ftemp OG_Ftemp)
RETURN NUMBER;
```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>linesty</i>	線表示形式を含む各フィールドのデータ・ポイント間を結ぶために使用する線の形式を指定します。

線の形式プロパティの例

```
/*The following procedure rotates
**the line style of a chart.
*/

PROCEDURE linestyle IS
  style number;
  ftemp og_ftemp;
  chart og_object;
BEGIN
  chart := og_get_object('chart');
  ftemp := og_get_ftemp(og_get_template(chart),0);
  style := og_get_linesty(ftemp);
  if style = og_spline_linestyle then
    og_set_linesty(ftemp, og_step_linestyle);
  elsif style = og_step_linestyle then
    og_set_linesty(ftemp, og_straight_linestyle);
  elsif style = og_straight_linestyle then
    og_set_linesty(ftemp, og_spline_linestyle);
  end if;
  og_update_chart(chart);
END;
```

オーバーラップ・プロパティ

説明

複数のフィールドからのデータ値を横棒グラフや縦棒グラフで表示する棒どうしがそれぞれオーバーラップする割合をパーセンテージを指定します。

構文

```
PROCEDURE OG_Set_Overlap
    (ftemp    OG_Ftemp,
     overlap  NUMBER);

FUNCTION OG_Get_Overlap
    (ftemp    OG_Ftemp)
RETURN NUMBER;
```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>overlap</i>	複数のフィールドからのデータ値を横棒チャートや縦棒チャートで表示する棒どうしがそれぞれオーバーラップする割合をパーセントで指定します。

オーバーラップ・プロパティの例

```
/*The following procedure reads
**the overlap percentage that has
**been specified.If the specified
**percentage is between 0 to 50,
**it redraws the column using
**90% overlap,if the percentage is
**over 90%,it redraws the columns
**with 0% overlap.
*/

PROCEDURE overlap IS
    percent number;
    ftemp og_ftemp;
    chart og_object;
BEGIN
    chart := og_get_object('chart');
    ftemp := og_get_ftemp(og_get_template(chart),0);
    percent := og_get_overlap(ftemp);
    if percent between 0 and 50 then
        og_set_overlap(ftemp, 90);
    else
        og_set_overlap(ftemp, 0);
    end if;
END;
```

表示位置プロパティ

説明

各項目において2つ以上のフィールドのデータ値間の関係を指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_Normal_Plotpos

OG_Fromprev_Plotpos

OG_Stacked_Plotpos

構文

```
PROCEDURE OG_Set_Plotpos
  (ftemp    OG_Ftemp,
   plotpos  NUMBER);

FUNCTION OG_Get_Plotpos
  (ftemp  OG_Ftemp)
RETURN NUMBER;
```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>plotpos</i>	各項目において2つ以上のフィールドのデータ値間の関係を指定します。

使用部分位置プロパティの例

```
/*The following button procedure rotates
**the plot position of columns in a chart.
*/

PROCEDURE plotpos IS
  pos number;
  ftemp og_ftemp;
  chart og_object;
BEGIN
  chart := og_get_object('chart');
  ftemp := og_get_ftemp(og_get_template(chart),0);
  pos := og_get_plotpos(ftemp);
  if pos = og_normal_plotpos then
    og_set_plotpos(ftemp,og_fromprev_plotpos);
  elsif pos = og_fromprev_plotpos then
    og_set_plotpos(ftemp, og_stacked_plotpos);
  elsif pos = og_stacked_plotpos then
    og_set_plotpos(ftemp, og_normal_plotpos);
  end if;
```



```
og_update_chart (chart);  
END;
```

表示形式プロパティ

説明

このフィールドをチャートに表示するために使用する要素を指定します。このプロパティには、次のビルトイン定数を指定できます。

OG_None_Ploptype

OG_Bar_Ploptype

OG_Line_Ploptype

OG_Symbol_Ploptype

OG_Fill_Ploptype

OG_Spike_Ploptype

OG_Label_Ploptype

構文

```
PROCEDURE OG_Set_Ploptype  
  (ftemp    OG_Ftemp,  
   ploptype NUMBER);  
  
FUNCTION OG_Get_Ploptype  
  (ftemp OG_Ftemp)  
RETURN NUMBER;
```

パラメータ

<i>ftemp</i>	フィールド・テンプレートのハンドルです。
<i>ploptype</i>	このフィールドをチャート上に表示するために使用する要素を指定します。

表示形式プロパティの例

```
/*On a mouse click, the following  
**procedure rotates the plot type  
**of a chart.  
*/
```

```
PROCEDURE Plottype (buttonobj IN og_object,  
                   hitobj IN og_object,  
                   win IN og_window,
```

```
                                eventinfo IN og_event) IS
chart og_object;
template og_template;
ftemp og_ftemp;
num number;
BEGIN
chart := og_get_object('chart');
template := og_get_template(chart);
ftemp := og_get_ftemp(template, 0);
num := og_get_plotttype(ftemp);
if num = og_none_plotttype then
    og_set_plotttype(ftemp, og_bar_plotttype);
elsif num = og_bar_plotttype then
    og_set_plotttype(ftemp, og_line_plotttype);
elsif num = og_line_plotttype then
    og_set_plotttype(ftemp, og_symbol_plotttype);
elsif num = og_symbol_plotttype then
    og_set_plotttype(ftemp, og_fill_plotttype);
elsif num = og_fill_plotttype then
    og_set_plotttype(ftemp, og_spike_plotttype);
elsif num = og_spike_plotttype then
    og_set_plotttype(ftemp, og_label_plotttype);
elsif num = og_label_plotttype then
    og_set_plotttype(ftemp, og_none_plotttype);
end if;
og_update_chart(chart);
END;
```

一般プロパティ

ボタン・プロシージャ・プロパティ

列・プロパティ

イベント・プロパティ

問合せ実行プロパティ

フォーマット・トリガー・プロパティ

オブジェクトを隠すプロパティ

内側枠ボックス・プロパティ

名前プロパティ

オブジェクト・タイプ・プロパティ

外側枠ボックス・プロパティ

親プロパティ

パラメータ設定プロパティ

ボタン・プロシージャ・プロパティ

説明

ボタン・プロシージャをオブジェクトと対応付けるハンドルです。このプロシージャが必要なマウス・イベントを確実に受け取るために、イベント・プロパティも適切に設定する必要があることに注意してください。

構文

```
PROCEDURE OG_Set_Button
  (object      OG_Object,
   buttonproc  OG_Buttonproc,
   damage      BOOLEAN      := TRUE,
   update_bbox BOOLEAN      := TRUE);

FUNCTION OG_Get_Button
  (object OG_Object)
RETURN OG_Buttonproc;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>buttonproc</i>	オブジェクトと対応付けるボタン・プロシージャのハンドル。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

ボタン・プロシージャ・プロパティの例

```
/*The following procedure reads the button
**procedure from a rectangle object, and adds
**the same procedure to a circle object.
*/

PROCEDURE transfer (buttonobj IN og_object,
                    hitobj IN og_object,
                    win IN og_window,
                    eventinfo IN og_event) IS
    rect og_object;
    circle og_object;
    proc og_buttonproc;
BEGIN
    rect := og_get_object('rect');
    circle := og_get_object('circle');
    proc := og_get_button(rect);
    og_set_button(circle, proc);
END;
```

列プロパティ

説明

オブジェクトを選択したときに、パラメータに設定される列値です。このプロパティは、チャート要素のみに適用されます。

構文

```
PROCEDURE OG_Set_Keycol
    (object OG_Object,
     keycol VARCHAR2);

FUNCTION OG_Get_Keycol
    (object OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
---------------	-------------

<i>keycol</i>	オブジェクトを選択した時にパラメータに設定される列値。
---------------	-----------------------------

列プロパティの例

```

/*The following procedure reads
**the column value of a parameter
**and assigns a different value to it
*/

```

```

PROCEDURE GenColumn IS
  rect og_object;
  param varchar2(20);
BEGIN
  rect := og_get_object('rect');
  param := og_get_keycol(rect);
  og_set_keycol(rect, 'init');
END;

```

イベント・プロパティ

説明

ボタン・プロパティが指定したプロシージャが受け取るマウス・イベントのタイプです。このプロパティには、下記のビルトイン定数を指定できます。プロシージャで複数のイベント・タイプを受け取ることができるようにするには、このプロパティを該当のイベントの定数の合計に設定します。OG_Mouse_Move_UpとOG_Mouse_Move_Downは、図表レイヤーに対してのみ使用されることに注意してください。

OG_No_Events

OG_Mouse_Down

OG_Mouse_Up

OG_Mouse_Move_Down

構文

```

PROCEDURE OG_Set_Events
  (object      OG_Object,
   events      NUMBER,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Events
  (object OG_Object)
RETURN NUMBER;

```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>events</i>	ボタン・プロパティで指定したプロシージャによって受け取られるマウス・イベントのタイプ。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

イベント・プロパティの例

```
/*The following procedure reads
**the current mouse event in an object,
**and assigns a different event to it.
*/

PROCEDURE Events IS
  rect og_object;
  events number;
BEGIN
  rect := og_get_object('rect');
  events := og_get_events(rect);
  if events = og_no_events then
    og_set_events(rect, og_mouse_down);
  elsif events = og_mouse_down then
    og_set_events(rect, og_mouse_up);
  elsif events = og_mouse_up then
    og_set_events(rect, og_mouse_move_down);
  elsif events = og_mouse_move_down then
    og_set_events(rect, og_no_events);
  end if;
END;
```

問合せ実行プロパティ

説明

オブジェクトを選択したときに、実行する問合せを指定します。

構文

```
PROCEDURE OG_Set_Execquery
  (object      OG_Object,
   execquery   OG_Query);

FUNCTION OG_Get_Execquery
  (object      OG_Object)
RETURN OG_Query;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>execquery</i>	オブジェクトを選択したときに実行される問合せを指定します。

問合せ実行プロパティの例

```
/*The following procedure reads the specified
**query of the object, and assigns a different
**query to it.
*/
```

```
PROCEDURE GenQuery IS
  rect og_object;
  query og_query;
  query1 og_query;
BEGIN
  rect := og_get_object('rect');
  query := og_get_execquery(rect);
  query1 := og_get_query('query1');
  og_set_execquery(rect, query1);
END;
```

フォーマット・トリガー・プロパティ

説明

オブジェクトのフォーマット・トリガーです。このプロパティは、チャート要素だけに適用されます。

構文

```
PROCEDURE OG_Set_Fmttrig
  (object OG_Object,
   fmttrig VARCHAR2);

FUNCTION OG_Get_Fmttrig
  (object OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>fmttrig</i>	オブジェクトのフォーマット・トリガーです。このプロパティは、チャート要素だけに適用されます。

フォーマット・トリガー・プロパティの例

```
/*The following procedure reads the specified
```

```

**format trigger from an object, and assigns a
**different format trigger to it.
*/

*/PROCEDURE GenFmtTrigger IS
  rect og_object;
  fmttrig varchar2(20);
BEGIN
  rect := og_get_object('rect');
  fmttrig := og_get_fmttrig(rect);
  og_set_fmttrig(rect, 'fmttrig1');
END;
```

オブジェクトを隠すプロパティ

説明

オブジェクトを隠します。

構文

```

PROCEDURE OG_Set_Hide
  (object OG_Object)
  hide BOOLEAN);

FUNCTION OG_Get_Hide
  (object OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>hide</i>	オブジェクトを隠す。

オブジェクトを隠すプロパティの例

```

/*The following button
**procedure hides or
**shows an object as it
**is selected.
*/

PROCEDURE OGBUTTONPROC0 (buttonobj IN og_object,
hitobj IN og_object,
win IN og_window,
```



```

eventinfo IN og_event) IS
val boolean;
BEGIN
val := og_get_hide(hitobj);
if val then
og_set_hide(hitobj, false);
og_set_bfcolor(hitobj, 'red');
else
og_set_hide(hitobj, true);
og_set_bfcolor(hitobj, 'red');
end if;
END;
```

内側枠ボックス・プロパティ

説明

オブジェクトの内側枠ボックスです。これは、枠の太さや他のプロパティ設定とは無関係に、オブジェクトの形を構成する（つまり、オブジェクトの4つの制御ポイントを接続する）四角形です。

構文

```

FUNCTION OG_Get_Ibbox
(object OG_Object)
RETURN OG_Rectangle;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
---------------	-------------

内側枠ボックス・プロパティの例

```

/*The following reads the dimensions
**of the inner bounding and outer
**bounding boxes and calculates
**the size of the actual bounding box.
*/
```

```

PROCEDURE GenIOBox IS
obj og_object;
ibox og_rectangle;
```

```

    obox og_rectangle;
    num number;
BEGIN
    obj := og_get_object('rect');
    ibox := og_get_ibbox(obj);
    obox := og_get_obbox(obj);
    num := (obox.height * obox.width) - (ibox.height*ibox.width);
END;
```

名前プロパティ

説明

オブジェクト名です。

構文

```

PROCEDURE OG_Set_Name
    (object      OG_Object,
     name        VARCHAR2
     damage      BOOLEAN      := TRUE,
     update_bbox BOOLEAN      := TRUE);

FUNCTION OG_Get_Name
    (object OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>name</i>	オブジェクト名。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

名前プロパティの例

```

/*The following procedure reads
**the name of the object and assigns
**another name to it.
*/

PROCEDURE GenName IS
    obj og_object;
    name varchar2(20);
BEGIN
    obj := og_get_object('circle');
    name := og_get_name(obj);
    og_set_name(obj, 'teresa');
```

END;

オブジェクト・タイプ・プロパティ

説明

オブジェクト・タイプです。このプロパティには、次のビルトイン定数を指定できます。

OG_Arc_Objtype

OG_Chart_Objtype

OG_Group_Objtype

OG_Image_Objtype

OG_Line_Objtype

OG_Poly_Objtype

OG_Rect_Objtype

OG_Rrect_Objtype

OG_Symbol_Objtype

OG_Text_Objtype

構文

```
FUNCTION OG_Get_Objtype
  (object OG_Object)
RETURN NUMBER;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
---------------	-------------

オブジェクト・タイプ・プロパティの例

```
/*The following button procedure checks
**the type of object being selected on by
**the mouse, and prints the type name to
**a text object.
*/

PROCEDURE GenObjType2 (buttonobj IN og_object,
                      hitobj IN og_object,
                      win IN og_window,
```

```
                                eventinfo IN og_event) IS
text og_object;
objtype number;

BEGIN
text := og_get_object('text object');
objtype := og_get_objtype(hitobj);
if objtype = og_arc_objtype then
    og_set_str(text, 'arc');
elsif objtype = og_chart_objtype then
    og_set_str(text, 'chart');
elsif objtype = og_group_objtype then
    og_set_str(text, 'group');
elsif objtype = og_image_objtype then
    og_set_str(text, 'image');
elsif objtype = og_line_objtype then
    og_set_str(text, 'line');
elsif objtype = og_poly_objtype then
    og_set_str(text, 'poly');
elsif objtype = og_rect_objtype then
    og_set_str(text, 'rect');
elsif objtype = og_rrect_objtype then
    og_set_str(text, 'rrect');
elsif objtype = og_symbol_objtype then
    og_set_str(text, 'symbol');
elsif objtype = og_text_objtype then
    og_set_str(text, 'text');
end if;

END;
```

外側枠ボックス・プロパティ

説明

オブジェクトの外側枠ボックスです。これは、オブジェクトを完全に囲める最小サイズの四角形です。外側枠ボックスは、オブジェクトの枠が太い場合には内側枠ボックスと異なる場合があります。内側枠ボックスではオブジェクトの本来あるべき形を辿っているのに対し、外側枠ボックスではオブジェクト全体を囲みます。

構文

```
FUNCTION OG_Get_Obbox
(object OG_Object)
RETURN OG_Rectangle;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
---------------	-------------

外側枠ボックス・プロパティの例

```

/*The following reads the dimensions of the
**inner bounding and outer bounding boxes and
**calculates the size of the actual bounding box.
*/

PROCEDURE GenIOBox IS
  obj og_object;
  ibox og_rectangle;
  obox og_rectangle;
  num number;
BEGIN
  obj := og_get_object('rect');
  ibox := og_get_ibbox(obj);
  obox := og_get_obbox(obj);
  num := (obox.height * obox.width) - (ibox.height*ibox.width);
END;

```

親プロパティ

説明

オブジェクトの親オブジェクトに対するハンドルです。

構文

```

FUNCTION OG_Get_Parent
  (object OG_Object)
RETURN OG_Object;

```

パラメータ

<i>object</i>	定義中のオブジェクト。
---------------	-------------

親プロパティの例

```

/*The following procedure gets the
**parent of the current object, and
**prints the name of the parent object
**to a text object.
*/

PROCEDURE GenParent IS
  text og_object;
  obj og_object;
  parent og_object;
  name varchar2(20);
BEGIN

```

```
text := og_get_object('text object');
obj := og_get_object('circle');
parent := og_get_parent(obj);
name := og_get_name(parent);
og_set_str(text, name);
END;
```

パラメータ設定プロパティ

説明

オブジェクトを選択したときに、値が設定されるパラメータです。

構文

```
PROCEDURE OG_Set_Setparam
  (object OG_Object,
   setparam VARCHAR2);

FUNCTION OG_Get_Setparam
  (object OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>setparam</i>	オブジェクトを選択すると値が設定されるパラメータです。

パラメータ設定プロパティの例

```
/*The following procedure reads the
**parameter of a rectangle object, and
**assigns a new parameter to it.
*/

PROCEDURE SetParam IS
  rect og_object;
  param varchar2(20);
BEGIN
  rect := og_get_object('rect');
  param := og_get_setparam(rect);
  og_set_setparam(rect, 'PARAM1');
END;
```

図形プロパティ

バックグラウンド線カラー・プロパティ

バックグラウンド塗りカラー・プロパティ

凹凸スタイル・プロパティ

端形式プロパティ

線種スタイル・プロパティ

枠パターン・プロパティ

枠幅プロパティ

塗りパターン・プロパティ

フォアグラウンド線カラー・プロパティ

フォアグラウンド塗りカラー・プロパティ

結合形式プロパティ

回転角度プロパティ

転送モード・プロパティ

バックグラウンド線カラー・プロパティ

説明

オブジェクトのバックグラウンド線カラーです。

構文

```
PROCEDURE OG_Set_Becolor
  (object      OG_Object,
   becolor     VARCHAR2,
   damage      BOOLEAN   := TRUE,
   update_bbox BOOLEAN   := TRUE);

FUNCTION OG_Get_Becolor
  (object OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>becolor</i>	オブジェクトのバックグラウンド線カラー。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

バックグラウンド線カラー・プロパティの例

```
/* /*The following procedure swaps the foreground
**and background edge colors.
*/

PROCEDURE FBEdgeColor IS
  obj og_object;
  fcolor varchar2(20);
  bcolor varchar2(20);
BEGIN
  obj := og_get_object('rect');
  fcolor := og_get_fecolor(obj);
  bcolor := og_get_becolor(obj);
  og_set_fecolor(obj, bcolor);
  og_set_becolor(obj, fcolor);
END;
```

バックグラウンド塗りカラー・プロパティ

説明

オブジェクトのバックグラウンド塗りカラーです。

構文

```
PROCEDURE OG_Set_Bfcolor
(object      OG_Object,
 bfcolor     VARCHAR2,
 damage      BOOLEAN    := TRUE,
 update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Bfcolor
(object      OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>bfcolor</i>	オブジェクトのバックグラウンド塗りカラー。
<i>damage</i>	ダメージ・フラグ。

<code>update_bbox</code>	バウンディング・ボックス更新フラグ。
--------------------------	--------------------

バックグラウンド塗りカラー・プロパティの例

```
/*The following procedure swaps the foreground
**and background fill colors.
*/
```

```
PROCEDURE FBFillColor IS
  obj og_object;
  fcolor varchar2(20);
  bcolor varchar2(20);
BEGIN
  obj := og_get_object('rect');
  fcolor := og_get_ffcolor(obj);
  bcolor := og_get_bfcolor(obj);
  og_set_ffcolor(obj, bcolor);
  og_set_bfcolor(obj, fcolor);
END;
```

凹凸スタイル・プロパティ

説明

オブジェクトの凹凸スタイルです。このプロパティには、次のビルトイン定数を指定できます。

OG_Inset_Bstyle

OG_Lowered_Bstyle

OG_Outset_Bstyle

OG_Plain_Bstyle

OG_Raised_Bstyle

構文

```
PROCEDURE OG_Set_Bevelstyle
  (object      OG_Object,
   bevelstyle  NUMBER,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Bevelstyle
  (object OG_Object)
RETURN NUMBER;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>bevelstyle</i>	オブジェクトの凹凸スタイル。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

凹凸スタイル・プロパティの例

```
/*The following button procedure rotates
**the bevel style of a selected object.
*/

PROCEDURE bevel (buttonobj IN og_object,
                 hitobj IN og_object,
                 win IN og_window,
                 eventinfo IN og_event) IS
    obj og_object;
    num number;
BEGIN
    obj := og_get_object('rect');
    num := og_get_bevelstyle(obj);
    if num = og_inset_bstyle then
        og_set_bevelstyle(obj, og_lowered_bstyle);
    elsif num = og_lowered_bstyle then
        og_set_bevelstyle(obj, og_outset_bstyle);
    elsif num = og_outset_bstyle then
        og_set_bevelstyle(obj, og_plain_bstyle);
    elsif num = og_plain_bstyle then
        og_set_bevelstyle(obj, og_raised_bstyle);
    elsif num = og_raised_bstyle then
        og_set_bevelstyle(obj, og_inset_bstyle);
    end if;
END;
```

端形式プロパティ

説明

オブジェクトの枠の端形式です。このプロパティには、次のビルトイン定数を指定できます。

OG_Butt_Cstyle

OG_Projecting_Cstyle

OG_Round_Cstyle

構文

```

PROCEDURE OG_Set_Capstyle
  (object      OG_Object,
   capstyle    NUMBER,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Capstyle
  (object OG_Object)
RETURN NUMBER;

```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>capstyle</i>	オブジェクトの枠の端形式。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

端形式プロパティの例

```

/*The following button procedure rotates
**the cap style of an object's edge.
*/

PROCEDURE CapStyle (buttonobj IN og_object,
                   hitobj IN og_object,
                   win IN og_window,
                   eventinfo IN og_event) IS
  num number;
BEGIN
  num := og_get_capstyle(hitobj);
  if num = og_butt_cstyle then
    og_set_capstyle(hitobj, og_projecting_cstyle);
  elsif num = og_projecting_cstyle then
    og_set_capstyle(hitobj, og_round_cstyle);
  elsif num = og_round_cstyle then
    og_set_capstyle(hitobj, og_butt_cstyle);
  end if;
END;

```

線種スタイル・プロパティ

説明

オブジェクトの枠の線種スタイルです。このプロパティには、次のビルトイン定数を指定できます。

OG_Solid_Dstyle
OG_Dot_Dstyle
OG_Long_Dstyle
OG_Dashdot_Dstyle
OG_Dotdot_Dstyle
OG_Short_Dstyle
OG_Dashdotdot_Dstyle

構文

```
PROCEDURE OG_Set_Dashstyle
  (object      OG_Object,
   dashstyle   NUMBER,
   damage      BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Dashstyle
  (object OG_Object)
RETURN NUMBER;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>dashstyle</i>	オブジェクトの枠の線種スタイル。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

線種スタイル・プロパティの例

```
*/The following button procedure rotates
**the dash style on hit object.
*/

PROCEDURE DashStyle (buttonobj IN og_object,
                    hitobj IN og_object,
                    win IN og_window,
                    eventinfo IN og_event) IS
  num number;
BEGIN
  num := og_get_dashstyle(hitobj);
  if num = og_solid_dstyle then
    og_set_dashstyle(hitobj, og_dot_dstyle);
  elsif num = og_dot_dstyle then
    og_set_dashstyle(hitobj,og_long_dstyle);
```

```

elseif num = og_long_dstyle then
  og_set_dashstyle(hitobj,og_dashdot_dstyle);
elseif num = og_dashdot_dstyle then
  og_set_dashstyle(hitobj,og_dotdot_dstyle);
elseif num = og_dotdot_dstyle then
  og_set_dashstyle(hitobj,og_short_dstyle);
elseif num = og_short_dstyle then
  og_set_dashstyle(hitobj,og_dashdotdot_dstyle);
elseif num = og_dashdotdot_dstyle then
  og_set_dashstyle(hitobj,og_solid_dstyle);
end if;
END;

```

枠パターン・プロパティ

説明

オブジェクトの枠パターンです。

構文

```

PROCEDURE OG_Set_Edgepatt
  (object      OG_Object,
   edgepatt    VARCHAR2,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Edgepatt
  (object OG_Object)
RETURN VARCHAR2;

```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>edgepatt</i>	オブジェクトの枠パターン。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

枠パターン・プロパティの例

```

/*The following procedure swaps the edge
**and fill patterns of an object.
*/

```

```

PROCEDURE EdgePattern IS
  obj og_object;
  edgepatt varchar2(20);
  fillpatt varchar2(20);

```

```
BEGIN
  obj := og_get_object('rect');
  edgepatt := og_get_edgepatt(obj);
  fillpatt := og_get_fillpatt(obj);
  og_set_edgepatt(obj, fillpatt);
  og_set_fillpatt(obj, edgepatt);
END;
```

枠幅プロパティの例

説明

オブジェクトの枠の幅です（レイアウト単位）。

構文

```
PROCEDURE OG_Set_Ewidth
  (object      OG_Object,
   ewidth      NUMBER,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Ewidth
  (object OG_Object)
RETURN NUMBER;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>ewidth</i>	オブジェクトの枠の幅（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

枠幅プロパティの例

```
/*The following procedure reads the edge
**width of a selected object.If the width
**is 0, it resets the width to value800.

PROCEDURE EdgeWidth IS
  obj og_object;
  width number;
BEGIN
  obj := og_get_object('rect');
  width := og_get_ewidth(obj);
  if width = 0 then
    og_set_ewidth(obj, 800);
  end if;
END;
```

塗りパターン・プロパティ

説明

オブジェクトの塗りパターンです。

構文

```
PROCEDURE OG_Set_Fillpatt
  (object      OG_Object,
   fillpatt    VARCHAR2,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Fillpatt
  (object OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>fillpatt</i>	オブジェクトの塗りパターン。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

塗りパターン・プロパティの例

```
/*The following procedure swaps the edge
**and fill patterns of an object.
*/
```

```
PROCEDURE EdgePattern IS
  obj og_object;
  edgepatt varchar2(20);
  fillpatt varchar2(20);
BEGIN
  obj := og_get_object('rect');
  edgepatt := og_get_edgepatt(obj);
  fillpatt := og_get_fillpatt(obj);
  og_set_edgepatt(obj, fillpatt);
  og_set_fillpatt(obj, edgepatt);
END;
```

フォアグラウンド線カラー・プロパティ

説明

オブジェクトのフォアグラウンド線カラーです。

構文

```
PROCEDURE OG_Set_Fecolor
(object          OG_Object,
 fecolor        VARCHAR2,
 damage         BOOLEAN    :=  TRUE,
 update_bbox    BOOLEAN    :=  TRUE);

FUNCTION OG_Get_Fecolor
(object OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>fecolor</i>	オブジェクトのフォアグラウンド線カラー。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

フォアグラウンド線カラー・プロパティの例

```
/*The following procedure swaps the foreground
**and background edge colors.
*/

PROCEDURE FBEdgeColor IS
  obj og_object;
  fcolor varchar2(20);
  bcolor varchar2(20);
BEGIN
  obj := og_get_object('rect');
  fcolor := og_get_fecolor(obj);
  bcolor := og_get_becolor(obj);
  og_set_fecolor(obj, bcolor);
  og_set_becolor(obj, fcolor);
END;
```


フォアグラウンド塗りカラー・プロパティ

説明

オブジェクトのフォアグラウンド塗りカラーです。

構文

```
PROCEDURE OG_Set_Ffcolor
  (object      OG_Object,
   ffcolor     VARCHAR2,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Ffcolor
  (object OG_Object)
RETURN VARCHAR2;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>ffcolor</i>	オブジェクトのフォアグラウンド塗りカラー。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

フォアグラウンド塗りカラー・プロパティの例

```
/*The following procedure swaps the foreground
**and background fill colors.
*/
```

```
PROCEDURE FBFillColor IS
  obj og_object;
  fcolor varchar2(20);
  bcolor varchar2(20);
BEGIN
  obj := og_get_object('rect');
  fcolor := og_get_ffcolor(obj);
  bcolor := og_get_bfcolor(obj);
  og_set_ffcolor(obj, bcolor);
  og_set_bfcolor(obj, fcolor);
END;
```

結合形式プロパティ

説明

オブジェクトの枠の結合形式です。このプロパティには、次のビルトイン定数を指定できます。

OG_Mitre_Jstyle

OG_Bevel_Jstyle

OG_Round_Jstyle

構文

```
PROCEDURE OG_Set_Joinstyle
(object          OG_Object,
 joinstyle      NUMBER,
 damage         BOOLEAN    := TRUE,
 update_bbox    BOOLEAN    := TRUE);

FUNCTION OG_Get_Joinstyle
(object OG_Object)
RETURN NUMBER;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>joinstyle</i>	オブジェクトの枠の結合形式。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

結合形式プロパティの例

```
/*The following button procedure rotates
**the join style of an object's edge.
*/

PROCEDURE JoinStyle (mitreonobj IN og_object,
                    hitobj IN og_object,
                    win IN og_window,
                    eventinfo IN og_event) IS
    num number;
BEGIN
    num := og_get_joinstyle(hitobj);
    if num = og_mitre_jstyle then
        og_set_joinstyle(hitobj, og_bevel_jstyle);
    elsif num = og_bevel_jstyle then
        og_set_joinstyle(hitobj,og_round_jstyle);
    elsif num = og_round_jstyle then
```

```

        og_set_joinstyle(hitobj,og_mitre_jstyle);
    end if;
END;
```

回転角度プロパティ

説明

オブジェクトの回転角度です。オブジェクトが最初に作成された時の角度が0と見なされ、現在の角度と最初の角度との時計回りで測った差がこのプロパティです。このプロパティを設定すると、オブジェクトを絶対角度で回転させることができ、またOG_Rotateプロシージャを使用すると、オブジェクトを相対角度で回転させることができます。（OG_Rotateを使用してオブジェクトを回転させると、回転角度プロパティが自動的に更新され、新しい絶対角度が反映されます。）

構文

```

PROCEDURE OG_Set_Rotang
(
    object      OG_Object,
    rotang      NUMBER,
    damage      BOOLEAN    := TRUE,
    update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Rotang
(
    object OG_Object)
RETURN NUMBER;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>rotang</i>	オブジェクトの回転角度。オブジェクトが最初に作成された時の角度が0と見なされ、現在の角度と最初の角度との時計回りで測った差がこのプロパティです。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

回転角度プロパティの例

```

/*The following procedure reads the rotation
**angle from a selected object, and rotates
**the object another 45 degrees to the right.
*/
```

```

PROCEDURE RotAngle IS
    obj og_object;
    rotang number;
BEGIN
    obj := og_get_object('rect');
```

```
    rotang := og_get_rotang(obj);
    og_set_rotang(obj, rotang+45);
END;
```

転送モード・プロパティ

説明

オブジェクトの転送モードです。このプロパティには、次のビルトイン定数を指定できます。

- OG_Copy_Transfer
- OG_Revcopy_Transfer
- OG_Or_Transfer
- OG_Revor_Transfer
- OG_Clear_Transfer
- OG_Revclear_Transfer
- OG_Invert_Transfer
- OG_Backinvert_Transfer

構文

```
PROCEDURE OG_Set_Transfer
  (object      OG_Object,
   transfer    NUMBER,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Transfer
  (object OG_Object)
RETURN NUMBER;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>transfer</i>	オブジェクトの転送モード。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

転送モード・プロパティの例

```
/*The following button procedure rotates the
**transfer mode of a selected object.
*/
```

```
PROCEDURE transher (copyonobj IN og_object,  
                    hitobj IN og_object,  
                    win IN og_window,  
                    eventinfo IN og_event) IS  
  
    num number;  
BEGIN  
    num := og_get_transfer(hitobj);  
    if num = og_copy_transfer then  
        og_set_transfer(hitobj, og_revcopy_transfer);  
    elsif num = og_revcopy_transfer then  
        og_set_transfer(hitobj, og_or_transfer);  
    elsif num = og_or_transfer then  
        og_set_transfer(hitobj, og_revor_transfer);  
    elsif num = og_revor_transfer then  
        og_set_transfer(hitobj, og_clear_transfer);  
    elsif num = og_clear_transfer then  
        og_set_transfer(hitobj, og_revclear_transfer);  
    elsif num = og_revclear_transfer then  
        og_set_transfer(hitobj, og_invert_transfer);  
    elsif num = og_invert_transfer then  
        og_set_transfer(hitobj, og_backinvert_transfer);  
    elsif num = og_backinvert_transfer then  
        og_set_transfer(hitobj, og_copy_transfer);  
    end if;  
END;
```

グループ・プロパティ

- 子の数プロパティ
- クリップ・フラグ・プロパティ

子の数プロパティ

説明

グループ・オブジェクトに属する子の数です。別のグループ・オブジェクトが、対象としているグループの子である場合、そのオブジェクトは単に1つのオブジェクトとしてカウントされます。

構文

```
FUNCTION OG_Get_Childcount
  (object OG_Object)
RETURN NUMBER;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
---------------	-------------

子の数プロパティの例

```
/*The following procedure gets the number
**of children in a group object.
*/

PROCEDURE GrpChildCnt IS
  grp og_object;
  cnt number;
BEGIN
  grp := og_get_object('group');
  cnt := og_get_childcount(grp);
END;
```

クリップ・フラグ・プロパティ

説明

グループ内の最初のオブジェクトの表示領域を、グループを表示する枠にするかどうかを指定します。TRUEに設定すると、レイアウトに表示されるオブジェクトは、枠内に含まれるグループ・

オブジェクト、またはグループ・オブジェクトの一部となります。枠になっているオブジェクト自体も表示されます。

構文

```
PROCEDURE OG_Set_Clipflag
  (object      OG_Object,
   clipflag    BOOLEAN,
   damage      BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Clipflag
  (object OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>object</i>	定義中のオブジェクト。
<i>clipflag</i>	グループ内の最初のオブジェクトの表示領域を、グループを表示する枠にするかどうかを指定する。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

クリップ・フラグ・プロパティの例

```
/*The following procedure checks if
**clipflag is true.Ifnot, it sets the
**flag to true.
*/
```

```
PROCEDURE GrpClipFlg IS
  grp og_object;
  flag boolean;
BEGIN
  grp := og_get_object('group');
  flag := og_get_clipflag(grp);
  if flag = false then
    og_set_clipflag(grp, true);
  end if;
END;
```

イメージ・プロパティ

- クリップする四角形プロパティ
- 粗さプロパティ
- 高さプロパティ
- 位置プロパティ
- 品質プロパティ
- 幅プロパティ

クリップする四角形プロパティ

説明

イメージを表示する枠のx座標とy座標、および高さ、幅です（レイアウト単位）。この枠の中に含まれるイメージのみが表示されます。このプロパティを指定しないと、表示する枠がイメージと同じ大きさになります。

構文

```
PROCEDURE OG_Set_Cliprect
  (image      OG_Object,
   cliprect   OG_Rectangle,
   damage     BOOLEAN      := TRUE,
   update_bbox BOOLEAN      := TRUE);

FUNCTION OG_Get_Cliprect
  (image OG_Object)
RETURN OG_Rectangle;
```

パラメータ

<i>image</i>	定義中のイメージ・オブジェクト。
<i>cliprect</i>	イメージを表示する枠のx座標とy座標、および高さ、幅（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

クリップする四角形プロパティの例

```
/*The following procedure reduces the
**size of the clipping rectangle by half.
*/
```



```
PROCEDURE ClipRect IS
    image og_object;
    rect og_rectangle;
BEGIN
    image := og_get_object('image');
    rect := og_get_cliprect(image);
    rect.height := rect.height/2;
    rect.width := rect.width/2;
    og_set_cliprect(image, rect);
    og_set_clipflag(image, true);
END;
```

粗さプロパティ

説明

Graphics Builderでイメージを表示するときに、そのイメージをディザリングするかどうかを指定します。このプロパティには、次のいずれかを指定できます。

構文

```
PROCEDURE OG_Set_Image_Dither
    (image OG_Object,
     dither BOOLEAN);

FUNCTION OG_Get_Image_Dither
    (image OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>image</i>	定義中のイメージ・オブジェクト。
<i>dither</i>	Graphics Builderでイメージを表示するときに、そのイメージをディザリングするかどうかを指定する。

粗さプロパティの例

```
/*The following button procedure
**dithers an image or removes
**dithering.
*/

PROCEDURE SetDither (buttonobj IN og_object,
                    hitobj IN og_object,
                    win IN og_window,
                    eventinfo IN og_event) IS

    val boolean;
```

```
image og_object;  
BEGIN  
  image := og_get_object('image');  
  val := og_get_image_dither(image);  
  if val then  
    og_set_image_dither(og_get_object('image'), false);  
  else  
    og_set_image_dither(og_get_object('image'), true);  
  end if;  
END;
```

高さプロパティ

説明

イメージの高さです(レイアウト単位)。このプロパティをイメージのデフォルトの高さ以外の値に設定すると、イメージは新しい高さに収まるようにスケーリングされます。

構文

(前述のOG_Set_Image_Sizeを参照してください。)

```
FUNCTION OG_Get_Image_Height  
  (image OG_Object)  
RETURN NUMBER;
```

パラメータ

<i>image</i>	定義中のイメージ・オブジェクト。
--------------	------------------

高さプロパティの例

```
/* The following procedure reduces  
**an image's size by half.  
*/  
  
PROCEDURE SizeWidthHeight IS  
  image og_object;  
  height number;  
  width number;  
BEGIN  
  image := og_get_object('image');  
  width := og_get_image_width(image);  
  height := og_get_image_height(image);  
  og_set_image_size(image, width/2, height/2);  
END;
```

位置プロパティ

説明

イメージの左上角のx座標とy座標です（レイアウト単位）。

構文

```
PROCEDURE OG_Set_Upperleft
  (image      OG_Object,
   upperleft  OG_Point,
   damage     BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Upperleft
  (image OG_Object)
RETURN OG_Point;
```

パラメータ

<i>image</i>	定義中のイメージ・オブジェクト。
<i>upperleft</i>	イメージの左上角のx座標とy座標（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

位置プロパティの例

```
/*The following procedure reads the
**(x,y) coordinates of the image's
**upper-left corner.If the coordinate
**is not(0,0), the procedure
**moves the image's upper-left
**corner to the (0,0) coordinate.
*/
```

```
PROCEDURE Position IS
  image og_object;
  pos og_point;
BEGIN
  image := og_get_object('image');
  pos := og_get_upperleft(image);
  if pos.x != 0 and pos.y != 0 then
    pos.x := 0;
    pos.y := 0;
    og_set_upperleft(image, pos);
  end if;
END;
```

品質プロパティ

説明

イメージを描画するときの品質を指定します。イメージの品質を高くすると、見栄えはよくなりますが、操作（たとえば、描画、移動、スケーリングなど）の処理時間が長くなります。このプロパティには、次のビルトイン定数を指定できます。

OG_High_Iquality

OG_Medium_Iquality

OG_Low_Iquality

構文

```
PROCEDURE OG_Set_Image_Quality
  (image    OG_Object,
   quality  NUMBER);

FUNCTION OG_Get_Image_Quality
  (image OG_Object)
RETURN NUMBER;
```

パラメータ

<i>image</i>	定義中のイメージ・オブジェクト。
<i>quality</i>	イメージを描画するときの品質を指定する。

品質プロパティの例

```
/*The following procedure checks image
**quality.If the image is currently drawn
**with high quality, the procedure redraws
**it with low quality.
*/

PROCEDURE GetQuality (buttonobj IN og_object,
                     hitobj IN og_object,
                     win IN og_window,
                     eventinfo IN og_event) IS
  image og_object;
  qty number;
BEGIN
  image := og_get_object('image');
  qty := og_get_image_quality(image);
  if qty = og_high_iquality then
    og_set_image_quality(image, og_low_iquality);
```

```

    end if;
END;
```

幅プロパティ

説明

イメージの幅です（レイアウト単位）。このプロパティをイメージのデフォルトの幅以外の値に設定すると、イメージは新しい幅に収まるようにスケーリングされます。

構文

```

PROCEDURE OG_Set_Image_Size
  (image      OG_Object,
   width      NUMBER,
   height     NUMBER,
   damage     BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Image_Width
  (image OG_Object)
RETURN NUMBER;
```

パラメータ

<i>image</i>	定義中のイメージ・オブジェクト。
<i>width</i>	イメージの幅（レイアウト単位）。
<i>height</i>	イメージの高さ（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

幅プロパティの例

```

/* The following procedure reduces
**an image's size by half.
*/

PROCEDURE SizeWidthHeight IS
  image og_object;
  height number;
  width number;
BEGIN
  image := og_get_object('image');
  width := og_get_image_width(image);
  height := og_get_image_height(image);
  og_set_image_size(image, width/2, height/2);
END;
```

線プロパティ

- 矢印スタイル・プロパティ
- 終点プロパティ
- 始点プロパティ

矢印スタイル・プロパティ

説明

線の矢印のスタイルです。このプロパティには、次のビルトイン定数を指定できます。

- OG_Noarrow_Astyle** 線に矢印を付けません。
- OG_Start_Astyle** 線の始点に矢印を付けます。
- OG_End_Astyle** 線の終点に矢印を付けます。
- OG_Both_Astyle** 線の両端に矢印を付けます。
- OG_Midtoend_Astyle** 線の中央に、始点の方を指す矢印を付けます。
- OG_Midtoend_Astyle** 線の中央に、終点の方を指す矢印を付けます。

構文

```
PROCEDURE OG_Set_Arrowstyle
  (line      OG_Object,
   arrowstyle  NUMBER,
   damage     BOOLEAN   :=  TRUE,
   update_bbox BOOLEAN   :=  TRUE);

FUNCTION OG_Get_Arrowstyle
  (line OG_Object)
RETURN NUMBER; パラメータ
```

パラメータ

<i>line</i>	定義中の線オブジェクト。
<i>arrowstyle</i>	線の矢印のスタイル。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

矢印スタイル・プロパティの例

```
/*The following procedure determines a
*line's current arrow style.If the line
**does not include arrows, the procedure adds
**arrows to both ends of the line.If the
*line does include arrows, the
**procedure removes them.
*/

PROCEDURE Arrow (buttonobj IN og_object,
                 hitobj IN og_object,
                 win IN og_window,
                 eventinfo IN og_event) IS
    arrow og_object;
    num number;
BEGIN
    arrow := og_get_object('arrow');
    num := og_get_arrowstyle(arrow);
    if num = og_noarrow_astyle then
        og_set_arrowstyle(arrow, og_both_astyle);
    else
        og_set_arrowstyle(arrow, og_noarrow_astyle);
    end if;
END;
```

終点プロパティ

説明

線の終点のx座標とy座標です（レイアウト単位）。

構文

```
PROCEDURE OG_Set_Endpt
    (line      OG_Object,
     endpt     OG_Point,
     damage    BOOLEAN    := TRUE,
     update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Endpt
    (line OG_Object)
RETURN OG_Point;
```

パラメータ

<i>line</i>	定義中の線オブジェクト。
<i>endpt</i>	線の終点のx座標とy座標（レイアウト単位）。

<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

終点プロパティの例

```
/*The following procedure reads the
**coordinates of the line's ending point.
**If the line does not end at the upper-left
**corner of the display, the procedure resets
**the end point to (0,0).
*/

PROCEDURE OGBUTTONPROC0 (buttonobj IN og_object,
                        hitobj IN og_object,
                        win IN og_window,
                        eventinfo IN og_event) IS
    arrow og_object;
    pos og_point;
BEGIN
    arrow := og_get_object('a');
    pos := og_get_endpt(arrow);
    if pos.x != 0 and pos.y != 0 then
        pos.x := 0;
        pos.y := 0;
        og_set_endpt(arrow, pos);
    end if;
END;
```

始点プロパティ

説明

線の始点のx座標とy座標です（レイアウト単位）。

構文

```
PROCEDURE OG_Set_Startpt
    (line      OG_Object,
     startpt   OG_Point,
     damage    BOOLEAN    := TRUE,
     update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Startpt(line OG_Object)
RETURN OG_Point;
```

パラメータ

<i>line</i>	定義中の線オブジェクト。
-------------	--------------

<i>startpt</i>	線の始点のx座標とy座標（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

始点プロパティの例

```
/* /*The following procedure reads
**the coordinates of a line's
**starting point.If the line does
**not start from the upper-left corner
**of the display, the procedure resets
**the start point to (0,0).
*/

PROCEDURE StartPt (buttonobj IN og_object,
                  hitobj IN og_object,
                  win IN og_window,
                  eventinfo IN og_event) IS
    arrow og_object;
    pos og_point;
BEGIN
    arrow := og_get_object('a');
    pos := og_get_startpt(arrow);
    if pos.x != 0 and pos.y != 0 then
        pos.x := 0;
        pos.y := 0;
        og_set_startpt(arrow, pos);
    end if;
END;
```

多角形プロパティ

- 閉じ形状プロパティ
- ポイント数プロパティ

閉じ形状プロパティ

説明

多角形の閉じ形状です。このプロパティには、次のいずれかを指定できます。

- TRUE** 多角形を閉じる。
- FALSE** 多角形を開く。

構文

```
PROCEDURE OG_Set_Poly_Closed
  (poly      OG_Object,
   closed     BOOLEAN,
   damage     BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Poly_Closed
  (poly OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>poly</i>	定義中の多角形。
<i>closed</i>	多角形の閉じ形状。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

閉じ形状プロパティの例

```
/*The following procedure determines
**whether a polygon is closed.
**If the polygon is open, the procedure
**closes it.
*/

PROCEDURE closure IS
  polygon og_object;
```

```
    val boolean;  
BEGIN  
    polygon := og_get_object('polygon');  
    val := og_get_poly_closed(polygon);  
    if val = false then  
        og_set_poly_closed(polygon, true);  
    end if;  
END;
```

ポイント数プロパティ

説明

多角形オブジェクトを構成するポイントの数です。

構文

```
FUNCTION OG_Get_Pointct  
    (poly OG_Object)  
RETURN NUMBER;
```

パラメータ

<i>poly</i>	定義中の多角形。
-------------	----------

ポイント数プロパティの例

```
/*The following procedure reads the  
**number of points that compose the  
**polygon object and prints the number  
**to a text object.  
*/
```

```
PROCEDURE PntCnt IS  
    text og_object;  
    polygon og_object;  
    cnt number;  
BEGIN  
    text := og_get_object('text object');  
    polygon := og_get_object('polygon');  
    cnt := og_get_pointct(polygon);  
    og_set_str(text, cnt);  
END;
```

印刷プロパティ

部数プロパティ

終了ページ・プロパティ

Landscapeプロパティ

名前プロパティ

ページ・サイズ・プロパティ

出力ファイル・プロパティ

開始ページ・プロパティ

部数プロパティ

説明

印刷部数です。

構文

```
PROCEDURE OG_Set_Copies
  (copies NUMBER);

FUNCTION OG_Get_Copies
  RETURN NUMBER;
```

パラメータ

<i>copies</i>	印刷部数。
---------------	-------

部数プロパティの例

```
/*The following procedure reads
**the number of copies and adds two more
**copies to print.
*/
```

```
PROCEDURE PrinterCopies IS
  copies number;
BEGIN
  copies := og_get_copies;
```

```
    og_set_copies(copis+2);  
END;
```

終了ページ・プロパティ

説明

印刷する最終ページです。

構文

```
PROCEDURE OG_Set_Endpage  
    (endpage NUMBER);  
  
FUNCTION OG_Get_Endpage  
    RETURN NUMBER;
```

パラメータ

<i>endpage</i>	印刷する最終ページ。
----------------	------------

終了ページ・プロパティの例

```
/*The following procedure reads the  
**end page number and resets it to the  
**original number plus two.  
*/
```

```
PROCEDURE PrinterEndPage IS  
    ep number;  
BEGIN  
    ep := og_get_endpage;  
    og_set_endpage(ep+2);  
END;
```

Landscapeプロパティ

説明

図表をLandscapeモードとPortraitモードのどちらで印刷するかを指定します。

構文

```
PROCEDURE OG_Set_Landscape  
    (landscape BOOLEAN);  
  
FUNCTION OG_Get_Landscape
```

```
RETURN BOOLEAN;
```

パラメータ

<i>landscape</i>	図表をLandscapeモードとPortraitモードのどちらで印刷するかを指定します。
------------------	--

Landscapeプロパティの例

```
/*The following procedure determines
**if the display is printed in landscape
**or portrait mode, and prints the mode
**type to a text object.
*/

PROCEDURE PrinterLandscape IS
landscape boolean;

BEGIN
landscape := og_get_landscape;
  if landscape then
    og_set_str(og_get_object('text object'), 'landscape');
  else
    og_set_str(og_get_object('text object'), 'portrait');
  end if;
END;
```

名前プロパティ

説明

現在のプリンタの名前です。

構文

```
PROCEDURE OG_Set_Printer_Name
  (name VARCHAR2);

FUNCTION OG_Get_Printer_Name
RETURN VARCHAR2;
```

パラメータ

<i>name</i>	現在のプリンタの名前。
-------------	-------------

名前プロパティの例

```
/*The following procedure sets the
**printer name and prints the name to
```

```
**a text object.  
*/
```

```
PROCEDURE PrinterName IS  
    name varchar2(30);  
BEGIN  
    name := og_get_printer_name;  
    og_set_str(og_get_object('text object'), name);  
END;
```

ページ・サイズ・プロパティ

説明

ページ・サイズです（インチ単位）。

構文

```
PROCEDURE OG_Set_Pagesize  
    (width    NUMBER,  
     height   NUMBER);
```

パラメータ

<i>width</i>	ページの幅（インチ単位）。
<i>height</i>	ページの高さ（インチ単位）。

ページ・サイズ・プロパティの例

```
/*The following procedure sets the  
**page size.  
*/
```

```
PROCEDURE PrinterPageSize IS  
    height number := 10*og_inch;  
    width number := 10*og_inch;  
    printfile varchar2(20);  
BEGIN  
    og_set_pagesize(height, width);  
END;
```

出力ファイル・プロパティ

説明

出力先のPostScriptファイルの名前です。このプロパティをNULLにすると、出力がプリンタへ送られます。

構文

```
PROCEDURE OG_Set_Printfile
  (filename VARCHAR2);

FUNCTION OG_Get_Printfile
  RETURN VARCHAR2;
```

パラメータ

<i>filename</i>	出力先のPostScriptファイルの名前。このプロパティをNULLにすると、出力がプリンタへ送られます。
-----------------	---

出力ファイル・プロパティの例

```
/*The following procedure sets the
**PostScript file name and prints it
**to a text object.
*/

PROCEDURE PrinterPrintFile IS
  printfile varchar2(20);
BEGIN
  og_set_printfile('myfile');
  printfile := og_get_printfile;
  og_set_str(og_get_object('text object'), printfile);
END;
```

開始ページ・プロパティ

説明

印刷する第1ページです。

構文

```
PROCEDURE OG_Set_Startpage
  (startpage NUMBER);

FUNCTION OG_Get_Startpage
  RETURN NUMBER;
```


パラメータ

<i>startpage</i>	印刷する第1ページ。
------------------	------------

開始ページ・プロパティの例

```
PROCEDURE PrinterStartPage IS
    sp number;
BEGIN
    sp := og_get_startpage;
    og_set_startpage(sp+2);
END;
```

問合せプロパティ

- 「キャッシュ・タイプ」プロパティ
- 「カスタム問合せプロシージャ」プロパティ
- 「日付書式」プロパティ
- 「オープン時実行」プロパティ
- 「タイマーで実行」プロパティ
- 「最大行」プロパティ
- 名前プロパティ
- 「問合せ後トリガー・プロシージャ」プロパティ
- 「問合せソース」プロパティ
- 「問合せタイプ」プロパティ

「キャッシュ・タイプ」プロパティ

説明

問合せの実行により新しく取り出されたデータを処理する方法を決定します。このプロパティには、次のビルトイン定数を指定できます。

- | | |
|----------------------------|---|
| OG_Append_Cachetype | データのすべての既存行が保持され、データの新規行が既存データ・セットの一番下に追加されます。 |
| OG_Copy_Cachetype | 前回の実行によるすべてのデータが特別バッファにコピーされ、新しく取り出されたデータに置換されます。 |
| OG_None_Cachetype | 前回の実行によるすべてのデータが破棄され、新しく取り出されたデータに置換されます。 |

構文

```
PROCEDURE OG_Set_Cachetype
  (query      OG_Query,
   cachetype  NUMBER);

FUNCTION OG_Get_Cachetype
```

```
(query OG_Query)
RETURN NUMBER;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>cachetype</i>	問合せの実行により新しく取り出されたデータを処理する方法を決定します。

「キャッシュ・タイプ」プロパティの例

```
/*The following procedure rotates the query
**cache type of a query.
*/

PROCEDURE QryCacheType (buttonobj IN og_object,
                        hitobj IN og_object,
                        win IN og_window,
                        eventinfo IN og_event) IS

    qry og_query;
    num number;
BEGIN
    qry := og_get_query('query0');
    num := og_get_cachetype(qry);
    if num = og_append_cachetype then
        og_set_cachetype(qry, og_copy_cachetype);
    elsif num = og_copy_cachetype then
        og_set_cachetype(qry, og_none_cachetype);
    elsif num = og_none_cachetype then
        og_set_cachetype(qry, og_append_cachetype);
    end if;
END;
```

「カスタム問合せプロシージャ」プロパティ

説明

カスタム問合せが実行されるときに起動するPL/SQLプロシージャです。

構文

```
PROCEDURE OG_Set_Customproc
    (query      OG_Query,
     customproc VARCHAR2);

FUNCTION OG_Get_Customproc
    (query OG_Query)
RETURN VARCHAR2;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>customproc</i>	カスタムの問合せが実行されるときに起動するPL/SQLプロシージャです。

「カスタム問合せプロシージャ」プロパティの例

```
/*The following button procedure swaps the two PL/SQL
**procedures which are invoked when a custom query is
**executed.
*/

PROCEDURE CustQry (buttonobj IN og_object,
                   hitobj IN og_object,
                   win IN og_window,
                   eventinfo IN og_event) IS
    proc varchar2(20);
    qry og_query;
BEGIN
    qry := og_get_query('query0');
    proc := og_get_customproc(qry);
    if proc = 'CUSTQRY1' then
        og_set_customproc(qry, 'CUSTQRY2');
    elsif proc = 'CUSTQRY2' then
        og_set_customproc(qry, 'CUSTQRY1');
    end if;
    og_execute_query(qry);
END;
```

「日付書式」プロパティ

説明

問合せの日付書式マスクです。

構文

```
PROCEDURE OG_Set_Dateformat
    (query      OG_Query,
     dateformat VARCHAR2);

FUNCTION OG_Get_Dateformat
    (query OG_Query)
RETURN VARCHAR2;
```

パラメータ

<i>query</i>	定義中の問合せ。
--------------	----------

<i>dateformat</i>	問合せの日付書式マスク。
-------------------	--------------

「日付書式」プロパティの例

```
/*The following procedure reads and sets
**the Date Format mask for the query.
*/
```

```
PROCEDURE QueryDateFmt IS
    qry og_query;
    DateFmt varchar2(20);
BEGIN
    qry := og_get_query('query0');
    DateFmt := og_get_dateformat(qry);
    og_set_dateformat(qry, 'DD-MM-YYYY');
    DateFmt := og_get_dateformat(qry);
END;
```

「オープン時実行」プロパティ

説明

図表を実行時にオープンしたときに、問合せを自動的に実行するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Execopen
    (query    OG_Query,
     execopen BOOLEAN);

FUNCTION OG_Get_Execopen
    (query OG_Query)
RETURN BOOLEAN;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>execopen</i>	図表を実行中にオープンしたときに問合せを自動的に実行するかどうかを指定します。

「オープン時実行」プロパティの例

```
/*The following procedure checks if the Execute
**on Open checkbox is checked.If it is checked,
**it unchecks it, or vice versa.
*/
```

```
PROCEDURE ExecOpen IS
    execOpen boolean;
    qry og_query;
BEGIN
    qry := og_get_query('query0');
    execOpen := og_get_execopen(qry);
    if execOpen then
        og_set_execopen(qry, false);
    else
        og_set_execopen(qry, true);
    end if;
END;
```

「タイマーで実行」プロパティ

説明

問合せを実行するときに使用するタイマーの名前です。NULLの場合は、タイマーを使用して問合せを実行しません。

構文

```
PROCEDURE OG_Set_ExecTimer
    (query      OG_Query,
     exectimer  VARCHAR2);

FUNCTION OG_Get_ExecTimer
    (query OG_Query)
RETURN VARCHAR2;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>exectimer</i>	問合せを実行するときに使用するタイマーの名前。NULLの場合は、タイマーを使用して問合せを実行しません。

「タイマーで実行」プロパティの例

```
/*The following procedure reads the name of
**the timer on which the query executes and
**assigns a new timer to the query.
*/

PROCEDURE ExecTimer IS
    exectimer varchar2(20);
    qry og_query;
BEGIN
    qry := og_get_query('query0');
```

```
exectimer := og_get_exectimer(qry);
og_set_exectimer(qry, 'timer1');
END;
```

「最大行」プロパティ

説明

問合せデータ・セットに保持されるデータの最大行数を指定します。NULLの場合は、すべての行が保持されます。

構文

```
PROCEDURE OG_Set_Maxrows
  (query    OG_Query,
   maxrows  NUMBER);

FUNCTION OG_Get_Maxrows
  (query    OG_Query)
RETURN NUMBER;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>maxrows</i>	問合せデータ・セットに保持されるデータの最大行数を指定します。NULLの場合は、すべての行が保持されます。

「最大行」プロパティの例

```
/*The following procedure reads the maximum
**number of rows of data that are retained
**in the query's data set, and adds two rows to
**the original number.
*/
```

```
PROCEDURE QueryMaxRow IS
  qry og_query;
  num number;
BEGIN
  qry := og_get_query('query0');
  num := og_get_maxrows(qry);
  og_set_maxrows(qry, num+2);
END;
```

「最大行フラグ」プロパティ

説明

データ・セットに含まれる行数に制限を設定するかどうかを指定します。このプロパティは、キャッシュ・タイプがOG_APPEND_CACHETYPEタイプのときに限り使用されます。

構文

```
PROCEDURE OG_Set_Maxflag
  (query    OG_Query,
   maxflag  BOOLEAN);

FUNCTION OG_Get_Maxflag
  (query OG_Query)
RETURN BOOLEAN;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>maxflag</i>	問合せデータ・セットに含むことができるデータの最大行数を指定します。

「最大行フラグ」プロパティの例

```
/*The following procedure reads the maximum
**number of rows of data that are retained
**in the query's data set, and adds two rows to
**the original number.If the incremented number
**is greater than 1024, then it disables the
**maximum rows flag, thus allowing the query to get
**all the rows of data.
*/

PROCEDURE MaxFlagToggle IS
  qry og_query;
  num number;
BEGIN
  qry := og_get_query('query0');
  num := og_get_maxrows(qry);

  num := num+2;
  og_set_maxrows(qry, num);

  IF ((num > 1024) AND (og_get_maxflag(qry)=TRUE)) THEN
    og_set_maxflag(qry,FALSE);
  END IF;
END;
```


名前プロパティ

説明

問合せの名前です。

構文

```
PROCEDURE OG_Set_Name
  (query  OG_Query,
   name   VARCHAR2);

FUNCTION OG_Get_Name
  (query  OG_Query)
RETURN VARCHAR2;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>name</i>	問合せの名前です。

名前プロパティの例

```
/*The following procedure swaps
** the name of two queries.
*/

PROCEDURE QueryName IS
  qry0 og_query;
  qry1 og_query;
  name0 varchar2(30);
  name1 varchar2(30);
BEGIN
  qry0 := og_get_query('query0');
  qry1 := og_get_query('query1');
  name0 := og_get_name(qry0);
  name1 := og_get_name(qry1);
  og_set_name(qry1, 'tmp');
  og_set_name(qry0, name1);
  og_set_name(qry1, name0);
END;
```

「問合せ後トリガー・プロシージャ」プロパティ

説明

問合せが実行された後で起動するPL/SQLプロシージャです。

構文

```
PROCEDURE OG_Set_Postproc
  (query      OG_Query,
   postproc   VARCHAR2);

FUNCTION OG_Get_Postproc
  (query OG_Query)
RETURN VARCHAR2;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>postproc</i>	問合せが実行された後で起動するPL/SQLプロシージャです。

「問合せ後トリガー・プロシージャ」プロパティの例

```
/*The following button procedure swaps the two PL/SQL
**procedures which are invoked after the query is
**executed.
*/

PROCEDURE PostTrigger (buttonobj IN og_object,
                       hitobj IN og_object,
                       win IN og_window,
                       eventinfo IN og_event) IS
  proc varchar2(20);
  gry og_query;
BEGIN
  gry := og_get_query('query0');
  proc := og_get_postproc(gry);
  if proc = 'POST1' then
    og_set_postproc(gry, 'POST2');
  elsif proc = 'POST2' then
    og_set_postproc(gry, 'POST1');
  end if;
  og_execute_query(gry);
END;
```

「問合せソース」プロパティ

説明

問合せデータのソースです。データベースからのデータの場合は、このプロパティに問合せのSQL SELECT文のテキストが含まれている必要があります。データがファイル・システムに格納されている場合は、このプロパティにデータ・ファイルのパスと名前を指定する必要があります。

構文

```
PROCEDURE OG_Set_Querysource
  (query      OG_Query,
   querysource VARCHAR2);

FUNCTION OG_Get_Querysource
  (query OG_Query)
RETURN VARCHAR2;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>querysource</i>	問合せデータのソース。

「問合せソース」プロパティの例

```
/*The following procedure swaps the source
**of two queries.
*/
```

```
PROCEDURE QuerySource IS
  qry0 og_query;
  qry1 og_query;
  source0 varchar2(50);
  source1 varchar2(50);
BEGIN
  qry0 := og_get_query('query0');
  qry1 := og_get_query('query1');
  source0:= og_get_querysource(qry0);
  source1:= og_get_querysource(qry1);
  og_set_querysource(qry0, source1);
  og_set_querysource(qry1, source0);
END;
```

「問合せタイプ」プロパティ

説明

問合せのタイプです。このプロパティには、次のビルトイン定数を指定できます。

OG_Custom_Qtype	問合せはカスタム問合せです。
OG_Exsql_Qtype	問合せによりSQL SELECT文を含むテキスト・ファイルからデータが取り出されます。
OG_Prn_Qtype	PRNファイルに基づいて問合せが行われます。

- OG_Sql_Qtype** 問合せはSQL SELECT文です。
- OG_Sylk_Qtype** SYLKファイルに基づいて問合せが行われます。
- OG_Wks_Qtype** WKSファイルに基づいて問合せが行われます。

構文

```
PROCEDURE OG_Set_Querytype
  (query      OG_Query,
   querytype  NUMBER);

FUNCTION OG_Get_Querytype
  (query  OG_Query)
RETURN NUMBER;
```

パラメータ

<i>query</i>	定義中の問合せ。
<i>querytype</i>	問合せのタイプです。

「問合せタイプ」プロパティの例

```
/*The following procedure rotates the
**query type of a query.
*/

PROCEDURE QryType (buttonobj IN og_object,
hitobj IN og_object,
               win IN og_window,
               eventinfo IN og_event) IS
  qry og_query;
  num number;
BEGIN
  qry := og_get_query('query0');
  num := og_get_querytype(qry);

  if num = og_custom_qtype then
    og_set_querytype(qry, og_exsql_qtype);
  elsif num = og_exsql_qtype then
    og_set_querytype(qry, og_prn_qtype);
  elsif num = og_prn_qtype then
    og_set_querytype(qry, og_sql_qtype);
  elsif num = og_sql_qtype then
    og_set_querytype(qry, og_sylk_qtype);
  elsif num = og_sylk_qtype then
    og_set_querytype(qry, og_wks_qtype);
  elsif num = og_wks_qtype then
    og_set_querytype(qry, og_custom_qtype);
  end if;
```

END;

四角形プロパティ

基本四角形プロパティ

基本四角形プロパティ

説明

四角形オブジェクトの基礎として使用する四角形の左上角のx座標とy座標、および高さと幅です（レイアウト単位）。

構文

```
PROCEDURE OG_Set_Rect_Baserect
  (rect      OG_Object,
   baserect  OG_Rectangle,
   damage    BOOLEAN      := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Rect_Baserect
  (rect OG_Object)
RETURN OG_Rectangle;
```

パラメータ

<i>rect</i>	定義中の四角形オブジェクト。
<i>baserect</i>	四角形オブジェクトの基礎として使用する四角形の左上角のx座標とy座標、および高さと幅（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

基本四角形プロパティの例

```
/*The following procedure determines the size
**of therectangle base and doubles it.
*/

PROCEDURE baseRect IS
  rect og_rectangle;
  obj og_object;
BEGIN
  obj := og_get_object('rect');
  rect := og_get_rect_baserect(obj);
  rect.x := rect.x * 2;
  rect.y := rect.y * 2;
```

```
rect.height := rect.height * 2;  
rect.width := rect.width * 2;  
og_set_rect_baserect(obj, rect);  
END;
```

参照線プロパティ

- 軸プロパティ
- 日付値プロパティ
- ラベル・プロパティ
- 数値プロパティ

軸プロパティ

説明

参照値の基準となる軸を指定します。このプロパティには、次のビルトイン定数を指定できます。

- OG_X_Axis
- OG_Y1_Axis
- OG_Y2_Axis

構文

```
PROCEDURE OG_Set_Axis
  (refline  OG_Refline,
   axis     NUMBER);

FUNCTION OG_Get_Axis
  (refline  OG_Refline)
RETURN NUMBER;
```

パラメータ

<i>refline</i>	定義中の参照線。
<i>axis</i>	参照値の基準となる軸を指定します。

軸プロパティの例

```
/*The following procedure maps
**the reference line against the
**Y1-axis if the line is currently
**mapped againstY2-axis.
*/

PROCEDURE Axis IS
```



```
chart og_object;  
axis number;  
refline og_refline;  
template og_template;  
BEGIN  
  chart := og_get_object('chart');  
  template := og_get_template(chart);  
  refline := og_get_refline(template, 0);  
  axis := og_get_axis(refline);  
  if axis = og_y2_axis then  
    og_set_axis(refline, og_y1_axis);  
  end if;  
  og_update_chart(chart);  
END;
```

日付値プロパティ

説明

参照線の日付値です。

構文

```
PROCEDURE OG_Set_Datevalue  
  (refline OG_Refline,  
   datevalue DATE);  
  
FUNCTION OG_Get_Datevalue  
  (refline OG_Refline)  
RETURN DATE;
```

パラメータ

refline	定義中の参照線。
datevalue	参照線の日付値。

日付値プロパティの例

```
/*The following procedure increases  
**reference line value by30 days.  
*/  
  
PROCEDURE DateVal IS  
  chart og_object;  
  dateval date;  
  refline og_refline;  
  template og_template;  
BEGIN
```

```
chart := og_get_object('chart');
template := og_get_template(chart);
refline := og_get_refline(template, 0);
dateval := og_get_datevalue(refline);
og_set_datevalue(refline, dateval+30);
og_update_chart(chart);
END;
```

ラベル・プロパティ

説明

凡例に表示する参照線を識別するテキスト・ラベルです。

構文

```
PROCEDURE OG_Set_Label
  (refline OG_Refline,
   label   VARCHAR2);

FUNCTION OG_Get_Label
  (refline OG_Refline)
RETURN VARCHAR2;
```

パラメータ

<i>refline</i>	定義中の参照線。
<i>label</i>	凡例に表示する参照線を識別するテキスト・ラベル。

ラベル・プロパティの例

```
/*The following procedure changes
**the reference line name to 'New Label'
**if this is not the current name of the
**label.
*/

PROCEDURE label IS
  chart og_object;
  label varchar2(20);
  refline og_refline;
  template og_template;
BEGIN
  chart := og_get_object('chart');
  template := og_get_template(chart);
  refline := og_get_refline(template, 0);
  label := og_get_label(refline);
  if label != 'New Label' then
```

```
        og_set_label(refline, 'New label');
    end if;
    og_update_chart(chart);
END;
```

数値プロパティ

説明

参照線の数値です。

構文

```
PROCEDURE OG_Set_Numvalue
    (refline  OG_Refline,
     numvalue NUMBER);

FUNCTION OG_Get_Numvalue
    (refline OG_Refline)
RETURN NUMBER;
```

パラメータ

<i>refline</i>	定義中の参照線。
<i>numvalue</i>	参照線の数値。

数値プロパティの例

```
/*The following procedure increases reference
**line value by500.
*/

PROCEDURE NumVal IS
    chart og_object;
    num number;
    refline og_refline;
    template og_template;
BEGIN
    chart := og_get_object('chart');
    template := og_get_template(chart);
    refline := og_get_refline(template, 0);
    num := og_get_numvalue(refline);
    og_set_numvalue(refline, num+500);
    og_update_chart(chart);
END;
```

丸い四角形プロパティ

- 基本四角形プロパティ
- 丸め半径プロパティ

基本四角形プロパティ

説明

四角形オブジェクトの基礎として使用する四角形の左上角のx座標とy座標、および高さと幅です（レイアウト単位）。

構文

```
PROCEDURE OG_Set_Rrect_Baserect
  (rrect      OG_Object,
   baserect   OG_Rectangle,
   damage     BOOLEAN      := TRUE,
   update_bbox BOOLEAN      := TRUE);

FUNCTION OG_Get_Rrect_Baserect
  (rrect OG_Object)
RETURN OG_Rectangle;
```

パラメータ

<i>rrect</i>	定義中の丸い四角形。
<i>baserect</i>	四角形オブジェクトの基礎として使用する四角形の左上角のx座標とy座標、および高さと幅（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

基本四角形プロパティの例

```
/*The following button procedure reduces
**the size of thebase rectangle or the
**rounded rectangle.
*/

PROCEDURE baserect (buttonobj IN og_object,
                    hitobj IN og_object,
                    win IN og_window,
                    eventinfo IN og_event) IS
  brect og_rectangle;
```

```

    rrect og_object;
BEGIN
    rrect := og_get_object('rrect');
    brect := og_get_rrect_baserect(rrect);
    brect.x := brect.x/2;
    brect.y := brect.y/2;
    brect.height := brect.height/2;
    brect.width := brect.width/2;
    og_set_rrect_baserect(rrect, brect);
END;
```

丸め半径プロパティ

説明

角を形成する弧を360度になるまで延長してできる楕円のx半径とy半径(レイアウト単位)です。

構文

```

PROCEDURE OG_Set_Corner
(
    rrect      OG_Object,
    corner     OG_Point,
    damage     BOOLEAN      := TRUE,
    update_bbox BOOLEAN      := TRUE);

FUNCTION OG_Get_Corner
(
    rrect OG_Object)
RETURN OG_Point;
```

パラメータ

<i>rrect</i>	定義中の丸い四角形。
<i>corner</i>	丸める角を形成する弧を360度になるまで延長してできる楕円のx半径とy半径(レイアウト単位)。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

丸め半径プロパティの例

```

/*The following button procedure doubles
**the x and y radiiof the ellipse used to
**create the corners of a rounded rectangle
*/

PROCEDURE corner (buttonobj IN og_object,
                  hitobj IN og_object,
                  win IN og_window,
                  eventinfo IN og_event) IS
```

```
    pt og_point;  
    rrect og_object;  
BEGIN  
    rrect := og_get_object('rrect');  
    pt := og_get_corner(rrect);  
    pt.x := pt.x*2;  
    pt.y := pt.y*2;  
    og_set_corner(rrect, pt);  
END;
```

単一テキスト・プロパティ

- カラー・プロパティ
- フォント・プロパティ
- テキスト文字列プロパティ

カラー・プロパティ

説明

文字列のテキストの表示カラー。これはテキスト自体のカラーであることに注意してください。テキスト・オブジェクトの枠を設定したり、カラーを塗る場合には、テキスト・オブジェクトのグラフィック・プロパティを変更します。

構文

```
FUNCTION OG_Get_Color
  (text      OG_Object,
   cmptext_index  NUMBER,
   smptext_index  NUMBER)
RETURN VARCHAR2;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>cmptext_index</i>	定義中の複数テキスト要素の索引番号。
<i>Smptext_index</i>	定義中の単一テキスト要素の索引番号。

カラー・プロパティの例

```
/* The following procedure reads the color
**of the current text object.If the
**current color is not red, it changes
**it to red.
*/

PROCEDURE color (buttonobj IN og_object,
                hitobj IN og_object,
                win IN og_window,
                eventinfo IN og_event) IS
color varchar2(20);
BEGIN
```

```
color := og_get_color(hitobj, 0, 0);
if color != 'red' then
  og_set_gcolor(hitobj, 'red');
end if;
END;
```

フォント・プロパティ

説明

文字列のテキストの表示フォント。

構文

```
FUNCTION OG_Get_Font_Typeface
(text          OG_Object,
  cmptext_index NUMBER,
  smptext_index NUMBER)
RETURN VARCHAR2;

FUNCTION OG_Get_Font_Ptsize
(text          OG_Object,
  cmptext_index NUMBER,
  smptext_index NUMBER)
RETURN NUMBER;

FUNCTION OG_Get_Font_Style
(text          OG_Object,
  cmptext_index NUMBER,
  smptext_index NUMBER)
RETURN NUMBER;

FUNCTION OG_Get_Font_Weight
(text          OG_Object,
  cmptext_index NUMBER,
  smptext_index NUMBER)
RETURN NUMBER;

FUNCTION OG_Get_Font_Width
(text          OG_Object,
  cmptext_index NUMBER,
  smptext_index NUMBER)
RETURN NUMBER;

FUNCTION OG_Get_Font_Kerning
(text          OG_Object,
  cmptext_index NUMBER,
  smptext_index NUMBER)
RETURN BOOLEAN;

FUNCTION OG_Get_Font_Charset
```



```
(text          OG_Object,  
  cmptext_index  NUMBER,  
  smptext_index  NUMBER)  
RETURN NUMBER;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>cmptext_index</i>	定義中の複数テキスト要素の索引番号。
<i>smptext_index</i>	定義中の単一テキスト要素の索引番号。

フォント・プロパティの例

```
/*The following procedure reads the  
**current typeface from selected text  
**object.If the current style is not  
**the same as the typeface from argument,  
**it assigns a new typeface to the  
**text object.  
*/  
  
PROCEDURE fonttypeface (text og_object, typeface varchar2) IS  
  style varchar2(10);  
BEGIN  
  style := og_get_font_typeface(text, 0,0);  
  if style != typeface then  
    og_set_font_typeface(text, typeface);  
  end if;  
END;
```

テキスト文字列プロパティ

説明

単一テキスト要素の実際のテキストを含む文字列。

構文

```
PROCEDURE OG_Set_Str  
  (text          OG_Object,  
   str           VARCHAR2,  
   damage        BOOLEAN   := TRUE,
```

```
update_bbox BOOLEAN := TRUE);  
  
FUNCTION OG_Get_Str  
  (text          OG_Object,  
   cmptext_index NUMBER,  
   smptext_index NUMBER)  
RETURN VARCHAR2;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>str</i>	単一テキスト要素の実際のテキストが含まれている文字列。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。
<i>cmptext_index</i>	定義中の複数テキスト要素の索引番号。
<i>smptext_index</i>	定義中の単一テキスト要素の索引番号。

テキスト文字列プロパティの例

```
/*The following procedure reads a text string from  
**a display and appends numbers to it.  
*/  
  
PROCEDURE TextString IS  
  text og_object;  
  
  BEGIN  
    text := og_get_object('text object');  
    og_set_str(text,og_get_str(text,0,0)||'123');  
  END;
```

サウンド・プロパティ

名前プロパティ

名前プロパティ

説明

サウンドの名前です。

構文

```
PROCEDURE OG_Set_Name
  (sound OG_Sound,
   name  VARCHAR2);

FUNCTION OG_Get_Name
  (sound OG_Sound)
RETURN VARCHAR2;
```

パラメータ

<i>sound</i>	定義中のサウンド・オブジェクト。
<i>name</i>	サウンドの名前。

名前プロパティの例

```
/*The following procedure gets the
**name of sound from the sound handler
**and assigns a new name to it.
*/

PROCEDURE SoundName (sound in og_sound) IS
name varchar2(10);
BEGIN
name := og_get_name(sound);
og_set_name(sound, name || '2');
END;
```

記号プロパティ

- 中央プロパティ
- 索引プロパティ
- 記号サイズ・プロパティ

中央プロパティ

説明

記号の中央のx座標とy座標です（レイアウト単位）。

構文

```
PROCEDURE OG_Set_Center
  (symbol      OG_Object,
   center      OG_Point,
   damage      BOOLEAN    :=  TRUE,
   update_bbox BOOLEAN    :=  TRUE);

FUNCTION OG_Get_Center
  (symbol OG_Object)
RETURN OG_Point;
```

パラメータ

<i>symbol</i>	定義中の記号オブジェクト。
<i>center</i>	記号の中央のx座標とy座標（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

中央プロパティの例

```
/*The following procedure moves
**the symbol from its original
**coordinate (x,y) to (x/2, y/2).
*/

PROCEDURE Center IS
  center og_point;
  symbol og_object;
BEGIN
  symbol := og_get_object('symbol');
```

```
center := og_get_center(symbol);
center.x := center.x/2;
center.y := center.y/2;
og_set_center(symbol, center);

END;
```

索引プロパティ

説明

記号パレットに表示するときの記号の位置の索引（または数値）です。

構文

```
PROCEDURE OG_Set_Indx
  (symbol      OG_Object,
   indx        NUMBER,
   damage       BOOLEAN    := TRUE,
   update_bbox  BOOLEAN    := TRUE);

FUNCTION OG_Get_Indx
  (symbol OG_Object)
RETURN NUMBER;
```

パラメータ

<i>symbol</i>	定義中の記号オブジェクト。
<i>indx</i>	記号パレットに表示するときの記号の位置の索引（または数値）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

索引プロパティの例

```
/*The following procedure gets the
**index of an object's symbol position
**in the symbol palette, and replaces
**the current symbol with the symbol
**which has an index value equal to the
**current index value + 1.
```

```
PROCEDURE get_index IS
  sym_index number;
  symbol og_object;
BEGIN
  symbol := og_get_object('symbol');
```

```
sym_index := og_get_indx(symbol);
og_set_indx(symbol, sym_index+1);
END;
```

記号サイズ・プロパティ

説明

記号のサイズです。このプロパティには、次のビルトイン定数を指定できます。

- OG_Large_Symsize
- OG_Medium_Symsize
- OG_Small_Symsize

構文

```
PROCEDURE OG_Set_Symsize
(symbol      OG_Object,
 symsize     NUMBER,
 damage      BOOLEAN    := TRUE,
 update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Symsize
(symbol OG_Object)
RETURN NUMBER;
```

パラメータ

<i>symbol</i>	定義中の記号オブジェクト。
<i>symsize</i>	記号のサイズ。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

記号サイズ・プロパティの例

```
/*The following procedure reads a symbol's
**size.If the symbol's size is not LARGE,
**the procedure changes it to LARGE;
**if a symbol's size is LARGE, the procedure
**changes it to small.
*/

PROCEDURE get_size IS
sym_size number;
```

```
    symbol og_object;  
BEGIN  
    symbol := og_get_object('symbol');  
    sym_size := og_get_symsize(symbol);  
    if sym_size != og_large_symsize then  
        og_set_symsize(symbol, og_large_symsize);  
    else  
        og_set_symsize(symbol, og_small_symsize);  
    end if;  
END;
```

テキスト・プロパティ

バウンディング・ボックスの高さプロパティ

バウンディング・ボックスの幅プロパティ

キャラクタ・セット・プロパティ

カラー・プロパティ

複数テキスト数プロパティ

カスタム間隔プロパティ

固定バウンディング・ボックス・プロパティ

水平線文字揃えプロパティ

水平原点プロパティ

非表示プロパティ

カーニング・プロパティ

近似プロパティ

原点プロパティ

ポイント・サイズ・プロパティ

スケーラブルな境界線プロパティ

スケーラブル・フォント・プロパティ

間隔プロパティ

スタイル・プロパティ

合成プロパティ

書体プロパティ

垂直文字揃えプロパティ

垂直原点プロパティ

フォントの太さプロパティ

フォントの幅プロパティ

丸めプロパティ

バウンディング・ボックスの高さプロパティ

説明

バウンディング・ボックスの高さです（レイアウト単位）。バウンディング・ボックスを変更すると常に、このプロパティが新しい高さを反映して自動的に更新されます。このプロパティは、「固定バウンディング・ボックス」プロパティがTRUEの場合にのみ、高さを設定するために使用します。

構文

（OG_Set_Text_Sizeを参照してください。）

```
FUNCTION OG_Get_Text_Height  
    (text OG_Object)  
RETURN NUMBER;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
-------------	------------------

バウンディング・ボックスの高さプロパティの例

```
/* The following procedure doubles the size of the  
**text object's bounding box.  
*/
```

```
PROCEDURE BBoxSize IS  
width number;  
height number;  
text og_object;  
BEGIN  
text := og_get_object('text object');  
width := og_get_text_width(text);  
height := og_get_text_height(text);  
og_set_text_size(text, width*2, height*2);  
END;
```

バウンディング・ボックスの幅プロパティ

説明

バウンディング・ボックスの幅です（レイアウト単位）。バウンディング・ボックスを変更すると常に、このプロパティが新しい幅を反映して自動的に更新されます。このプロパティは、「固定バウンディング・ボックス」プロパティがTRUEの場合にのみ、幅を設定するために使用します。

構文

```
PROCEDURE OG_Set_Text_Size
  (text      OG_Object,
   width     NUMBER,
   height    NUMBER,
   damage    BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Text_Width
  (text OG_Object)
RETURN NUMBER;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>width</i>	バウンディング・ボックスの幅（レイアウト単位）。
<i>height</i>	バウンディング・ボックスの高さ（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

バウンディング・ボックスの幅プロパティの例

```
/* The following procedure doubles the size of the
**text object's bounding box.
*/

PROCEDURE BBoxSize IS
width number;
height number;
text og_object;
BEGIN
  text := og_get_object('text object');
  width := og_get_text_width(text);
  height := og_get_text_height(text);
  og_set_text_size(text, width*2, height*2);
END;
```

キャラクタ・セット・プロパティ

説明

フォントのキャラクタ・セットです。このフィールドの値には、U.S. ASCII、漢字、アラビア文字などのキャラクタ・セットを指定します。すべてのキャラクタ・セットをすべてのシステムで利用できるわけではありません。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。このフィールドには、次のビルトイン定数を指定できます。

OG_Us7ascii_Charset

OG_We8dec_Charset

OG_We8hp_Charset

OG_Us8pc437_Charset

OG_We8ebcdic37_Charset

OG_We8ebcdic500_Charset

OG_We8pc850_Charset

OG_D7dec_Charset

OG_F7dec_Charset

OG_S7dec_Charset

OG_E7dec_Charset

OG_Sf7ascii_Charset

OG_Ndk7dec_Charset

OG_I7dec_Charset

OG_Nl7dec_Charset

OG_Ch7dec_Charset

OG_Sf7dec_Charset

OG_We8iso8859p1_Charset

OG_Ee8iso8859p2_Charset

OG_Se8iso8859p3_Charset

OG_Nee8iso8859p4_Charset

OG_Cl8iso8859p5_Charset
OG_Ar8iso8859p6_Charset
OG_El8iso8859p7_Charset
OG_Iw8iso8859p8_Charset
OG_We8iso8859p9_Charset
OG_Ar8asmo708plus_Charset
OG_Ar7asmo449plus_Charset
OG_We8macroman8_Charset
OG_Jvms_Charset
OG_Jeuc_Charset
OG_Jdec_Charset
OG_Sjis_Charset
OG_Jdbcs_Charset
OG_Jhp_Charset
OG_Ksc5601_Charset
OG_Kibm5540_Charset
OG_Kdbcs_Charset
OG_Cgb231380_Charset
OG_Cdbcs_Charset
OG_Big5_Charset
OG_Cns1164386_Charset

構文

```
PROCEDURE OG_Set_Font_Charset
(
  text      OG_Object,
  charset   NUMBER,
  damage    BOOLEAN := TRUE,
  update_bbox BOOLEAN := TRUE);
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
-------------	------------------

<i>charset</i>	フォントのキャラクタ・セット。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

キャラクタ・セット・プロパティの例

```
/*The following button procedure checks
**if the selected text object's font set
**is in US ASCII Character Set.If not,
**it assigns ASCII Character Set to the object.
*/

PROCEDURE CharSet (buttonobj IN og_object,
                   hitobj IN og_object,
                   win IN og_window,
                   eventinfo IN og_event) IS

setNo number;
BEGIN

    SetNo := OG_get_Font_charset(hitobj,0,0);
    if SetNo != og_US7ASCII_Charset then
        og_set_font_charset(hitobj, og_US7ASCII_charset);
    end if;
END;
```

カラー・プロパティ

説明

テキスト・オブジェクトのカラーです。

構文

```
PROCEDURE OG_Set_Gcolor
(text      OG_Object,
gcolor     VARCHAR2,
damage     BOOLEAN    := TRUE,
update_bbox BOOLEAN    := TRUE);
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>gcolor</i>	テキスト・オブジェクトのカラー。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

カラー・プロパティの例

```
/*The following procedure reads
** the color of the current text object.
**If the current color is not red,
**it changes it to red.
*/

PROCEDURE color (buttonobj IN og_object,
                 hitobj IN og_object,
                 win IN og_window,
                 eventinfo IN og_event) IS
color varchar2(20);
BEGIN
    color := og_get_color(hitobj, 0, 0);
    if color != 'red' then
        og_set_gcolor(hitobj, 'red');
    end if;
END;
```

複数テキスト数プロパティ

説明

テキスト・オブジェクトを構成するテキスト要素の数です。

構文

```
FUNCTION OG_Get_Ctcount
(text OG_Object)
RETURN NUMBER;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
-------------	------------------

複数テキスト数プロパティの例

```
/*The following procedure counts the number of
**compound text elements in a text object.
*/

PROCEDURE CompoundTextCnt IS
    num number;
    text og_object;
```

```

BEGIN
    text := og_get_object;
    num := og_get_ctcount(text);
END;

```

カスタム間隔プロパティ

説明

テキスト・オブジェクトのカスタム間隔です（レイアウト単位）。このプロパティは、「間隔」プロパティがカスタム間隔に設定されている場合にのみ、間隔を指定するために使用します。

構文

```

PROCEDURE OG_Set_Custom
    (text      OG_Object,
     custom    NUMBER,
     damage    BOOLEAN    := TRUE,
     update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Custom
    (text OG_Object)
RETURN NUMBER;

```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>custom</i>	テキスト・オブジェクトのカスタム間隔（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

カスタム間隔プロパティの例

```

/*The following procedure resets the
**custom spacing to twice its original setting.
*/

PROCEDURE CustomSpacing (buttonobj IN og_object,
                        hitobj IN og_object,
                        win IN og_window,
                        eventinfo IN og_event) IS

num number;
BEGIN
    num := og_get_custom(hitobj);
    og_set_str(hitobj, 'abc'| |num);
    og_set_custom(hitobj, num*2);

```

END;

固定バウンディング・ボックス・プロパティ

説明

テキスト・オブジェクトのバウンディング・ボックスを固定サイズのままにするかどうかを指定します。このプロパティをTRUEに設定した場合は、「幅」プロパティと「高さ」プロパティの値によってバウンディング・ボックスのサイズが変更されます。

構文

```
PROCEDURE OG_Set_Fixed
  (text      OG_Object,
   fixed     BOOLEAN,
   damage    BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Fixed
  (text OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>fixed</i>	テキスト・オブジェクトのバウンディング・ボックスを固定サイズのままにするかどうかを指定します。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

固定バウンディング・ボックス・プロパティの例

```
/*The following procedure checks if the text object's bounding box
**remains a fixed size.
*/

PROCEDURE FixBBox IS
  val boolean;
  text og_object;
BEGIN
  text := og_get_object('text object');
  val := og_get_fixed(text);
  if val then
    og_set_fixed(text, false);
  else
    og_set_fixed(text, true);
```



```

        end if;
    END;

```

水平文字揃えプロパティ

説明

テキスト・オブジェクトの水平文字揃えです。このプロパティには、次のビルトイン定数を指定できます。

OG_Left_Halign

OG_Center_Halign

OG_Right_Halign

構文

```

PROCEDURE OG_Set_Halign
(
    text      OG_Object,
    halign    NUMBER,
    damage    BOOLEAN    := TRUE,
    update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Halign
(
    text OG_Object)
RETURN NUMBER;

```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>halign</i>	テキスト・オブジェクトの水平文字揃え。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

水平文字揃えプロパティの例

水平文字揃え

```

/*The following procedure reads the horizontal
**alignment and readjusts it.
*/

```

```

PROCEDURE Halign IS
    num number:=og_right_halign;
    text og_object;

```

```
BEGIN
  text := og_get_object('text object');
  num := og_get_halign(text);
  if num = og_left_halign then
    og_set_halign(text, og_center_halign);
  elsif num = og_center_halign then
    og_set_halign(text, og_right_halign);
  elsif num = og_right_halign then
    og_set_halign(text, og_left_halign);
  end if;
END;
```

水平原点プロパティ

説明

テキスト・オブジェクトの原点に対する水平位置です。このプロパティには、次のビルトイン定数を指定できます。

- OG_Left_Horigin** バウンディング・ボックスの左枠を原点とします。
- OG_Center_Horigin** バウンディング・ボックスの左枠と右枠の中央を原点とします。
- OG_Right_Horigin** バウンディング・ボックスの右枠を原点とします。

構文

```
PROCEDURE OG_Set_Horigin
  (text      OG_Object,
   horigin    NUMBER,
   damage     BOOLEAN   := TRUE,
   update_bbox BOOLEAN   := TRUE);

FUNCTION OG_Get_Horigin
  (text OG_Object)
RETURN NUMBER;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>horigin</i>	原点に対するテキスト・オブジェクトの水平位置。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

水平原点プロパティの例

```
/*The following procedure reads the horizontal
```

```
**origin and readjusts it.
*/

PROCEDURE Horigin IS
num number;
text og_object;
BEGIN
text := og_get_object('text object');
num := og_get_horigin(text);
if num = og_left_horigin then
og_set_horigin(text, og_center_horigin);
elsif num = og_center_horigin then
og_set_horigin(text, og_rightr_horigin);
elsif num = og_right_horigin then
og_set_horigin(text, og_left_horigin);
end if;
END;
```

非表示プロパティ

説明

テキスト・オブジェクト内のテキストを表示しないようにするかどうかを指定します。これは、パスワードを見せたくない場合に、パスワードを入力するテキスト・フィールドに指定すると便利です。

構文

```
PROCEDURE OG_Set_Invisible
(text      OG_Object,
invisible  BOOLEAN,
damage     BOOLEAN := TRUE,
update_bbox  BOOLEAN := TRUE);

FUNCTION OG_Get_Invisible
(text OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>invisible</i>	テキスト・オブジェクト内のテキストを表示しないようにするかどうかを指定します。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

非表示プロパティの例

```
/*The following procedure determines if
** text in a text object is invisible.If it
** is invisible it makes it visible; if it is
** visible it makes it invisible.
*/

PROCEDURE Invisible IS
  val boolean;
  text og_object;
BEGIN
  text := og_get_object('text object');
  val := og_get_invisible(text);
  if val then
    og_set_invisible(text, false);
  else
    og_set_invisible(text, true);
  end if;
END;
```

カーニング・プロパティ

説明

フォントをカーニングするかどうかを指定します。カーニングとは、テキストを読みやすくするために、隣接する文字間の空白を調整することです。

構文

```
PROCEDURE OG_Set_Font_Kerning
(text      OG_Object,
kerning    BOOLEAN,
damage     BOOLEAN := TRUE,
update_bbox BOOLEAN := TRUE);
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>kerning</i>	フォントをカーニングするかどうかを指定します。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

カーニング・プロパティの例

```
/*The following button procedure turns font
**kerning on and off for a selected text object.

PROCEDURE kerning (buttonobj IN og_object,
                   hitobj IN og_object,
                   win IN og_window,
                   eventinfo IN og_event) IS
    val boolean;
BEGIN
    val := og_get_font_kerning(hitobj,0,0);
    if val then
        og_set_font_kerning(hitobj, false);
    else
        og_set_font_kerning(hitobj, true);
    end if;
END;
```

近似プロパティ

説明

指定したフォントが見つからない場合に、Graphics Builderによってそれに最も近いフォントが代用されるようにするかどうかを指定します。近似フォントを見つけるときの優先順位は、書体、サイズ、スタイル、フォントの太さ、幅の順です（つまり、Graphics Builderによって、まず指定された書体が検索され、次にサイズが検索され、という順番で検索されていきます）。

構文

```
PROCEDURE OG_Set_Font_Nearest
(text          OG_Object,
nearest       BOOLEAN,
damage        BOOLEAN    := TRUE,
update_bbox   BOOLEAN    := TRUE);
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>nearest</i>	指定したフォントが見つからない場合に、Graphics Builderによってそれに最も近いフォントが代用されるようにするかどうかを指定します。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

近似プロパティの例

```
/*The following button procedure sets nearest
**properties to true.
*/

/*The following button procedure sets nearest
**properties to true. PROCEDURE nearest (buttonobj IN og_object,
                                hitobj IN og_object,
                                win IN og_window,
                                eventinfo IN og_event) IS

BEGIN
    OG_Set_Font_Nearest(hitobj, true);
END;
```

原点プロパティ

説明

テキスト・オブジェクトの左上角のx座標とy座標です（レイアウト単位）。

構文

```
PROCEDURE OG_Set-Origin
(
    text      OG_Object,
    origin     OG_Point,
    damage     BOOLEAN    := TRUE,
    update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get-Origin
(
    text OG_Object)
RETURN OG_Point;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>origin</i>	テキスト・オブジェクトの左上角のx座標とy座標です（レイアウト単位）。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

ポイント・サイズ・プロパティ

説明

フォントのサイズです。このフィールドの値はシステムによって異なります。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。

構文

```
PROCEDURE OG_Set_Font_Ptsize
  (text      OG_Object,
   psize     NUMBER,
   damage    BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>psize</i>	フォントのサイズ。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

ポイント・サイズ・プロパティの例

```
/*The following procedure reads the point size of the
**current object and enlarges the text object to double
** its original size.
*/
```

```
PROCEDURE psize (text og_object) IS
  num number;
BEGIN
  num := og_get_font_ptsize(text,0,0);
  og_set_font_ptsize(text, num*2);
END;
```

スケーラブルな境界線プロパティ

説明

テキスト・オブジェクトをスケーリングするときにテキスト・オブジェクトの境界線もスケーリングするかどうかを指定します。

構文

```
PROCEDURE OG_Set_Bbscale
  (text      OG_Object,
   bbscale   BOOLEAN,
   damage    BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Bbscale
  (text OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>bbscale</i>	テキスト・オブジェクトをスケーリングするときにテキスト・オブジェクトの境界線もスケーリングするかどうかを指定します。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

スケーラブルな境界線プロパティの例

```
/*The following procedure checks if the text
**object's bounding box is scaled when the
**text object is scaled.
*/

PROCEDURE Scalebox IS
  val boolean;
  text og_object;
BEGIN
  text := og_get_object('text object');
  val := og_get_bbscale(text);
  if val then
    og_set_bbscale(text, false);
  else
    og_set_bbscale(text, true);
  end if;
END;
```

スケーラブル・フォント・プロパティ

説明

テキスト・オブジェクトをスケーリングするときにフォントのサイズもスケーリングするかどうかを指定します。このプロパティには、次のいずれかを指定できます。

構文

```
PROCEDURE OG_Set_Fontscale
  (text      OG_Object,
   fontsize  BOOLEAN,
   damage     BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Fontscale
  (text OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>fontsize</i>	テキスト・オブジェクトをスケーリングするときにフォントのサイズもスケーリングするかどうかを指定します。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

スケーラブル・フォント・プロパティの例

```
/* The following procedure checks if the point size is
**scaled when the text object is scaled.
*/

PROCEDURE Scalable IS
  val boolean;
  text og_object;
BEGIN
  text := og_get_object('text object');
  val := og_get_fontscale(text);
  if val then
    og_set_fontscale(text, false);
  else
    og_set_fontscale(text, true);
  end if;
END;
```

間隔プロパティ

説明

テキスト・オブジェクトの行間隔です。このプロパティには、下記のビルトイン定数を指定できます。カスタム間隔を設定する場合は、「カスタム間隔」プロパティの値で間隔を指定する必要があります。

OG_Single_Space

OG_Onehalf_Space

OG_Double_Space

OG_Custom_Space テキストにカスタム行間隔が使用されます。実際に使用する間隔は、「カスタム間隔」プロパティで定義します。

構文

```
PROCEDURE OG_Set_Spacing
  (text      OG_Object,
   spacing    NUMBER,
   damage     BOOLEAN   :=  TRUE,
   update_bbox BOOLEAN   :=  TRUE);

FUNCTION OG_Get_Spacing
  (text OG_Object)
RETURN NUMBER;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>spacing</i>	テキスト・オブジェクトの行間隔。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

間隔プロパティの例

```
/* The following button procedure
**rotates the spacing setting of a
**text object.
*/

/* The following button procedure
**rotates the spacing setting of a
**text object. PROCEDURE Spacing (buttonobj IN og_object,
                                hitobj IN og_object,
                                win IN og_window,
                                eventinfo IN og_event) IS

num number;
BEGIN
  num := og_get_spacing(hitobj);
  if num = og_single_space then
    og_set_spacing(hitobj, og_onehalf_space);
  elsif num = og_onehalf_space then
    og_set_spacing(hitobj, og_double_space);
```

```

elseif num = og_double_space then
og_set_spacing(hitobj, og_custom_space);
elseif num = og_custom_space then
og_set_spacing(hitobj, og_single_space);
end if;
END;

```

スタイル・プロパティ

説明

フォントのスタイルです。すべてのスタイルをすべてのシステムで利用できるわけではありません。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。このフィールドには、次のビルトイン定数を指定できます。

OG_Blink_Fontstyle

OG_Inverted_Fontstyle

OG_Italic_Fontstyle

OG_Oblique_Fontstyle

OG_Outline_Fontstyle

OG_Overstrike_Fontstyle

OG_Plain_Fontstyle

OG_Shadow_Fontstyle

OG_Underline_Fontstyle

OG_Unknown_Fontstyle スタイルが不明であることを意味します。スタイルをこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってそのフォント・スタイルが判別されない場合は、この値が戻されます。

構文

```

PROCEDURE OG_Set_Font_Style
(
  text      OG_Object,
  style     NUMBER,
  damage    BOOLEAN    := TRUE,
  update_bbox BOOLEAN  := TRUE);

```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>style</i>	フォントのスタイル。

<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

スタイル・プロパティの例

```
/*The following button procedure
**rotates the text style
*/

/*The following button procedure
**rotates the text style
*/ PROCEDURE style (buttonobj IN og_object,
                    hitobj IN og_object,
                    win IN og_window,
                    eventinfo IN og_event) IS

text og_object;
num number;
BEGIN
    text := og_get_object('text object');
    num := og_get_font_style(text,0,0);
    if num = og_blink_fontstyle then
        og_set_font_style(text, og_inverted_fontstyle);
    elsif num = og_inverted_fontstyle then
        og_set_font_style(text, og_italic_fontstyle);
    elsif num = og_italic_fontstyle then
        og_set_font_style(text, og_oblique_fontstyle);
    elsif num = og_oblique_fontstyle then
        og_set_font_style(text, og_outline_fontstyle);
    elsif num = og_outline_fontstyle then
        og_set_font_style(text, og_overstrike_fontstyle);
    elsif num = og_overstrike_fontstyle then
        og_set_font_style(text, og_plain_fontstyle);
    elsif num = og_plain_fontstyle then
        og_set_font_style(text, og_shadow_fontstyle);
    elsif num = og_shadow_fontstyle then
        og_set_font_style(text, og_underline_fontstyle);
    elsif num = og_underline_fontstyle then
        og_set_font_style(text, og_unknown_fontstyle);
    elsif num = og_unknown_fontstyle then
        og_set_font_style(text, og_blink_fontstyle);
    end if;
END;
```

合成プロパティ

説明

指定したフォントが見つからない場合、Graphics Builderによって近似フォントを変形させて指定フォントを合成するかどうかを指定します。

構文

```
PROCEDURE OG_Set_Font_Synthesize
  (text      OG_Object,
   synthesize BOOLEAN,
   damage     BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>synthesize</i>	指定したフォントが見つからない場合、Graphics Builderによって近似フォントを変形させて指定フォントを合成するかどうかを指定します。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

合成プロパティの例

```
/*The following button procedure sets synthesize
** properties to true.
*/
```

```
/*The following button procedure sets synthesize
**properties to true. PROCEDURE synthesize (buttonobj IN og_object,
      hitobj IN og_object,
      win IN og_window,
      eventinfo IN og_event) IS
```

```
BEGIN
  OG_Set_Font_Synthesize(hitobj, true);
END;
```

書体プロパティ

説明

フォントの書体(フォント名)です。このフィールドの値はシステムによって異なり、times、courier およびhelveticaなどの書体が含まれていることがあります。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。

構文

```
PROCEDURE OG_Set_Font_Typeface
  (text          OG_Object,
   typeface      VARCHAR2,
   damage        BOOLEAN    :=  TRUE,
   update_bbox   BOOLEAN    :=  TRUE);
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>typeface</i>	フォントの書体 (フォント名)。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

書体プロパティの例

```
/*The following procedure reads the
**current typeface from selected text
**object.If the current style is not
**the same as the typeface from argument,
**it assigns a new typeface to the text
**object.

PROCEDURE fonttypeface (text og_object, typeface varchar2)IS
style varchar2(10);
BEGIN
  style := og_get_font_typeface(text, 0,0);
  if style != typeface then
    og_set_font_typeface(text, typeface);
  end if;
END;
```

垂直文字揃えプロパティ

説明

テキスト・オブジェクトの垂直文字揃えです。このプロパティには、次のビルトイン定数を指定できます。

OG_Top_Valign

OG_Middle_Valign

OG_Bottom_Valign

構文

```
PROCEDURE OG_Set_Valign
  (text      OG_Object,
   valign    NUMBER,
   damage    BOOLEAN    := TRUE,
   update_bbox BOOLEAN    := TRUE);

FUNCTION OG_Get_Valign
  (text OG_Object)
RETURN NUMBER;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>valign</i>	テキスト・オブジェクトの垂直文字揃え。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

垂直文字揃えプロパティの例

```
/* The following procedure reads the vertical
** alignment and readjusts it.
*/
```

```
PROCEDURE Valign IS
  num number;
  text og_object;
BEGIN
  text := og_get_object('text object');
  num := og_get_valign(text);
  if num = og_top_valign then
    og_set_valign(text, og_middle_valign);
  elsif num = og_middle_valign then
```

```
        og_set_valign(text, og_bottom_valign);
    elsif num = og_bottom_valign then
        og_set_valign(text, og_top_valign);
    end if;
END;
```

垂直原点プロパティ

説明

原点に対するテキスト・オブジェクトの垂直位置です。このプロパティには、次のビルトイン定数を指定できます。

- OG_Top_Vorigin** バウンディング・ボックスの上枠を原点とします。
- OG_Middle_Vorigin** バウンディング・ボックスの上下枠の中央を原点とします。
- OG_Bottom_Vorigin** バウンディング・ボックスの下枠を原点とします。

構文

```
PROCEDURE OG_Set_Vorigin
(
    text          OG_Object,
    vorigin       NUMBER,
    damage        BOOLEAN    :=  TRUE,
    update_bbox   BOOLEAN    :=  TRUE);

FUNCTION OG_Get_Vorigin
(
    text OG_Object)
RETURN NUMBER;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>vorigin</i>	原点に対するテキスト・オブジェクト の垂直位置。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

垂直原点プロパティの例

```
/*The following procedure reads the
**vertical origin and readjusts it.
*/

PROCEDURE Vorigin IS
    num number;
```



```
text og_object;  
BEGIN  
  text := og_get_object('text object');  
  num := og_get_vorigin(text);  
  if num = og_top_vorigin then  
    og_set_vorigin(text, og_middle_vorigin);  
  elsif num = og_middle_vorigin then  
    og_set_vorigin(text, og_bottom_vorigin);  
  elsif num = og_bottom_vorigin then  
    og_set_vorigin(text, og_top_vorigin);  
  end if;  
END;
```

フォントの太さプロパティ

説明

フォントの太さです。すべての太さをすべてのシステムで利用できるわけではありません。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。このフィールドには、次のビルトイン定数を指定できます。

OG_Bold_Fontweight

OG_Demibold_Fontweight

OG_Demilight_Fontweight

OG_Extrabold_Fontweight

OG_Extralight_Fontweight

OG_Light_Fontweight

OG_Medium_Fontweight

OG_Ultrabold_Fontweight

OG_Ultralight_Fontweight

OG_Unknown_Fontweight フォントの太さが不明であることを意味します。フォントの太さをこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってその太さが判別されない場合は、この値が戻されます。

構文

```
PROCEDURE OG_Set_Font_Weight  
(text      OG_Object,  
 weight    NUMBER,  
 damage    BOOLEAN := TRUE,
```

```
update_bbox BOOLEAN := TRUE);
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>weight</i>	フォントの太さ。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

フォントの太さプロパティの例

```
/*The following button procedure
**rotates the font weight for a
**selected text object.
*/

/*The following button procedure
**rotates the font weight for a
**selected text object. PROCEDURE weight (buttonobj IN og_object,
                                         hitobj IN og_object,
                                         win IN og_window,
                                         eventinfo IN og_event) IS

    num number;
    text og_object;
BEGIN
    text := og_get_object('text object');
    num := og_get_font_weight(text,0,0);
    if num = og_bold_fontweight then
        og_set_font_weight(text, og_demibold_fontweight);
    elsif num = og_demibold_fontweight then
        og_set_font_weight(text, og_demilight_fontweight);
    elsif num = og_demilight_fontweight then
        og_set_font_weight(text, og_extrabold_fontweight);
    elsif num = og_extrabold_fontweight then
        og_set_font_weight(text, og_extralight_fontweight);
    elsif num = og_extralight_fontweight then
        og_set_font_weight(text, og_light_fontweight);
    elsif num = og_light_fontweight then
        og_set_font_weight(text, og_medium_fontweight);
    elsif num = og_medium_fontweight then
        og_set_font_weight(text, og_ultrabold_fontweight);
    elsif num = og_ultrabold_fontweight then
        og_set_font_weight(text, og_ultralight_fontweight);
    elsif num = og_ultralight_fontweight then
        og_set_font_weight(text, og_unknown_fontweight);
    elsif num = og_unknown_fontweight then
        og_set_font_weight(text, og_bold_fontweight);
```

```

    end if;
END;

```

フォントの幅プロパティ

説明

フォントの幅です。すべての幅をすべてのシステムで使用できるわけではありません。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。このフィールドには、次のビルトイン定数を指定できます。

OG_Dense_Fontwidth

OG_Expand_Fontwidth

OG_Extradense_Fontwidth

OG_Extraexpand_Fontwidth

OG_Normal_Fontwidth

OG_Semidense_Fontwidth

OG_Semiexpand_Fontwidth

OG_Ultradense_Fontwidth

OG_Ultraexpand_Fontwidth

OG_Unknown_Fontwidth 幅が不明であることを意味します。フォントの幅をこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってその幅が判別されない場合は、この値が戻されます。

構文

```

PROCEDURE OG_Set_Font_Width
(
    text      OG_Object,
    width     NUMBER,
    damage    BOOLEAN    := TRUE,
    update_bbox BOOLEAN    := TRUE);

```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>width</i>	フォントの幅。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

フォントの幅プロパティの例

```
/*The following button procedure
**rotates the font width for a
**selected test object.
*/

/*The following button procedure
**rotates the font width for a
**selected test object. PROCEDURE width (buttonobj IN og_object,
                                hitobj IN og_object,
                                win IN og_window,
                                eventinfo IN og_event) IS

    num number;
    text og_object;
BEGIN
    text := og_get_object('text object');
    num := og_get_font_width(text,0,0);
    if num = og_dense_fontwidth then
        og_set_font_width(text, og_expand_fontwidth);
    elsif num = og_expand_fontwidth then
        og_set_font_width(text, og_extradense_fontwidth);
    elsif num = og_extradense_fontwidth then
        og_set_font_width(text, og_extraexpand_fontwidth);
    elsif num = og_extraexpand_fontwidth then
        og_set_font_width(text, og_normal_fontwidth);
    elsif num = og_normal_fontwidth then
        og_set_font_width(text, og_semidense_fontwidth);
    elsif num = og_semidense_fontwidth then
        og_set_font_width(text, og_semiexpand_fontwidth);
    elsif num = og_semiexpand_fontwidth then
        og_set_font_width(text, og_ultradense_fontwidth);
    elsif num = og_ultradense_fontwidth then
        og_set_font_width(text, og_ultraexpand_fontwidth);
    elsif num = og_ultraexpand_fontwidth then
        og_set_font_width(text, og_unknown_fontwidth);
    elsif num = og_unknown_fontwidth then
        og_set_font_width(text, og_dense_fontwidth);
    end if;
END;
```

丸めプロパティ

説明

テキストを、テキスト・オブジェクトのバウンディング・ボックスに収まるように「ラップ（改行）」するかどうかを指定します。複数テキスト要素はテキストの行を表し、単一テキスト要素から構成されます。

構文

```
PROCEDURE OG_Set_Wrap
  (text      OG_Object,
   wrap      BOOLEAN,
   damage    BOOLEAN := TRUE,
   update_bbox BOOLEAN := TRUE);

FUNCTION OG_Get_Wrap
  (text OG_Object)
RETURN BOOLEAN;
```

パラメータ

<i>text</i>	定義中のテキスト・オブジェクト。
<i>wrap</i>	テキストを、テキスト・オブジェクトのバウンディング・ボックスに収まるように「ラップ（改行）」するかどうかを指定します。
<i>damage</i>	ダメージ・フラグ。
<i>update_bbox</i>	バウンディング・ボックス更新フラグ。

丸めプロパティの例

```
/*The following procedure checks if the text is 'wrapped'
** into the text's bounding box.
*/

PROCEDURE wrap IS
  val boolean;
  text og_object;
BEGIN
  text := og_get_object('text object');
  val := og_get_wrap(text);
  if val then
    og_set_wrap(text, false);
  else
    og_set_wrap(text, true);
  end if;
END;
```

タイマー・プロパティ

- アクティブ・プロパティ
- 間隔プロパティ
- 名前プロパティ
- プロシージャ・プロパティ

アクティブ・プロパティ

説明

タイマーをアクティブにするかどうかを指定します。

構文

```
PROCEDURE OG_Set_Active
  (timer  OG_Timer,
   active BOOLEAN);

FUNCTION OG_Get_Active
  (timer OG_Timer)
RETURN BOOLEAN;
```

パラメータ

<i>timer</i>	定義中のタイマー・オブジェクト。
<i>active</i>	タイマーをアクティブにするかどうかを指定します。

アクティブ・プロパティの例

```
/* The following sets the timer to inactive if it
**is currently in active mode.
*/

PROCEDURE TimerActive IS
  val boolean;
  timer og_timer;
BEGIN
  timer := og_get_timer('timer2');
```

```
val := og_get_active(timer);  
if val then  
  og_set_active(timer, false);  
end if;  
END;
```

間隔プロパティ

説明

タイマー・プロシージャの実行間隔の秒数です。

構文

```
PROCEDURE OG_Set_Interval  
  (timer    OG_Timer,  
   interval NUMBER);  
  
FUNCTION OG_Get_Interval  
  (timer OG_Timer)  
RETURN NUMBER;
```

パラメータ

<i>timer</i>	定義中のタイマー・オブジェクト。
<i>interval</i>	タイマー・プロシージャの実行間隔の秒数。

間隔プロパティの例

```
/* The following procedure adds two  
**seconds to the original timer  
**interval.  
*/  
  
PROCEDURE TimerInterval IS  
  interval number;  
  timer og_timer;  
BEGIN  
  timer := og_get_timer('timer2');  
  interval := og_get_interval(timer);  
  og_set_interval(timer, interval+2);
```

END;

名前プロパティ

説明

タイマーの名前です。

構文

```
PROCEDURE OG_Set_Name
  (timer OG_Timer,
   name  VARCHAR2);

FUNCTION OG_Get_Name
  (timer OG_Timer)
RETURN VARCHAR2;
```

パラメータ

<i>timer</i>	定義中のタイマー・オブジェクト。
<i>name</i>	タイマーの名前。

名前プロパティの例

```
/*The following procedure appends
**a '1' to the name of a timer.
*/

PROCEDURE TimerName IS
name varchar2(10);
timer og_timer;
BEGIN
timer := og_get_timer('timer1');
name := og_get_name(timer);
og_set_name(timer, name||'1');
END;
```


プロシージャ・プロパティ

説明

タイマーの起動時に実行するプロシージャの名前です。

構文

```
PROCEDURE OG_Set_Timerproc
    (timer      OG_Timer,
     timerproc  VARCHAR2);

FUNCTION OG_Get_Timerproc
    (timer OG_Timer)
RETURN VARCHAR2;
```

パラメータ

<i>timer</i>	定義中のタイマー・オブジェクト。
<i>timerproc</i>	タイマーの起動時に実行するプロシージャの名前。

プロシージャ・プロパティの例

```
/* The following procedure changes the
**timer procedure to "NewProc" if it is
** current timer procedure.
*/
```

```
PROCEDURE TimerProc IS
name varchar2(20);
timer og_timer;
BEGIN
timer := og_get_timer('timer2');
name := og_get_timerproc(timer);
if name != 'NewProc' then
og_set_timerproc(timer, 'NewProc');
end if;
END;
```

ウィンドウ・プロパティ

説明

ウィンドウの位置とサイズは、「画面解像度単位」、つまり一般的にピクセルと呼ばれる単位で表します。画面解像度の水平値と垂直値を取得するには、ビルトインOG_Get_Ap_Hscreen_ResおよびOG_Get_Ap_Vscreen_Resを使用します。

実際の数値ではなく、これらのビルトインを使用することにより、アプリケーションでは、画面解像度の異なるシステムで一貫性のある表示が維持されます。

高さプロパティ

ヘルプ・ターゲット・プロパティ

名前プロパティ

位置プロパティ

幅プロパティ

高さプロパティ

説明

ウィンドウの高さです（画面解像度単位）。

構文

```
( OG_Set_Window_Sizeを参照してください。 )  
  
FUNCTION OG_Get_Window_Height  
  (window OG_Window)  
RETURN NUMBER;
```

パラメータ

window	定義中のウィンドウ。
--------	------------

高さプロパティの例

```
/* /*The following procedure resizes  
**the window to half itsoriginal size.  
*/  
  
PROCEDURE WinSize IS
```

```

        window og_window;
        width number;
        height number;
BEGIN
    window := og_get_window('Main Layout');
    width := og_get_window_width(window);
    height := og_get_window_height(window);
    og_set_window_size(window, width/2,height/2);
END;
```

ヘルプ・ターゲット・プロパティ

説明

ウィンドウがアクティブの間に「ヘルプ・システム」を起動したときに表示される、ランタイム・ヘルプ内のハイパーテキスト・ターゲットです。

構文

```

PROCEDURE OG_Set_Helptarget
    (window      OG_Window,
     helptarget  VARCHAR2);

FUNCTION OG_Get_Helptarget
    (window OG_Window)
RETURN VARCHAR2;
```

パラメータ

<i>window</i>	定義中のウィンドウ。
<i>helptarget</i>	ウィンドウがアクティブの間に「ヘルプ・システム」を起動した時に表示される、ランタイム・ヘルプ内のハイパーテキスト・ターゲット。

ヘルプ・ターゲット・プロパティの例

```

/*The following procedure sets the help
**target to 'NewTarget' if "New Target" is not
** the current helptarget.
*/
```

```

PROCEDURE Help IS
    window og_window;
    help varchar2(20);
BEGIN
    window := og_get_window('Main Layout');
    help := og_get_helptarget(window);
```

```
if help != 'NewTarget' then
  og_set_helptarget(window, 'NewTarget');
end if;
END;
```

名前プロパティ

説明

ウィンドウの名前です。実行時には、レイアウト・ウィンドウのデフォルト名が「Main Layout」となります。

構文

```
PROCEDURE OG_Set_Name
  (window  OG_Window,
   name    VARCHAR2);

FUNCTION OG_Get_Name
  (window  OG_Window)
RETURN VARCHAR2;
```

パラメータ

<i>window</i>	定義中のウィンドウ。
<i>name</i>	ウィンドウの名前。

名前プロパティの例

```
/*The following procedure resets
**the name of the window if its name
**is not 'Main Layout'.
*/

PROCEDURE Name IS
  window og_window;
  name varchar2(20);
BEGIN
  window := og_get_window('Main Layout');
  name := og_get_name(window);
  if name != 'Main Layout' then
    og_set_name(window, 'Main Layout');
  end if;
END;
```

位置プロパティ

説明

ウィンドウの左上角のx座標とy座標です（画面解像度単位）。

構文

```
PROCEDURE OG_Set_Position
    (window    OG_Window,
     position  OG_Point);

FUNCTION OG_Get_Position
    (window  OG_Window)
RETURN OG_Point;
```

パラメータ

<i>window</i>	定義中のウィンドウ。
<i>position</i>	ウィンドウの左上角のx座標とy座標（画面解像度単位）。

位置プロパティの例

```
/*The following procedure repositions
**the window.
*/

/*The following procedure repositions
**the window. PROCEDURE Position IS
    window og_window;
    pos og_point;
BEGIN
    window := og_get_window('Main Layout');
    pos := og_get_position(window);
    pos.x := pos.x*2;
    pos.y := pos.y*2;
    og_set_position(window, pos);
END;
```

幅プロパティ

説明

ウィンドウの幅です（画面解像度単位）。

構文

```
PROCEDURE OG_Set_Window_Size
  (window  OG_Window,
   width   NUMBER,
   height  NUMBER);

FUNCTION OG_Get_Window_Width
  (window  OG_Window)
RETURN NUMBER;
```

パラメータ

<i>window</i>	定義中のウィンドウ。
<i>width</i>	ウィンドウの幅（画面解像度単位）。
<i>height</i>	ウィンドウの高さ（画面解像度単位）。

幅プロパティの例

```
/*The following procedure resizes the window
** to half itsoriginal size.
*/

PROCEDURE WinSize IS
  window og_window;
  width number;
  height number;
BEGIN
  window := og_get_window('Main Layout');
  width := og_get_window_width(window);
  height := og_get_window_height(window);
  og_set_window_size(window, width/2,height/2);
END;
```

属性

属性レコードの使用方法

- 概要
- ショートカット・ビルトイン
- アプリケーション属性レコード

概要

多くのビルトイン・サブプログラムは"属性レコード"と呼ばれる引数を受け取ります。"属性"とは、単にGraphics Builderオブジェクトのプロパティや特性のことをいいます。たとえば、四角形には、フォアグラウンド・カラーという属性があり、円弧には、開始角度と終了角度という属性があります。Graphics Builderでは、属性を確認した上で、オブジェクトの構造と外観を描写します。

Graphics Builderには、このような属性を制御するように、新しいビルトイン変数データ型がいくつか提供されています。これらのデータ型のほとんどは、レコード・データ型として定義されています。（レコード・データ型についての詳細は、『PL/SQLユーザーズ・ガイドおよびリファレンス』を参照してください）。1つのレコード内の各フィールドは、特定の属性を表します。したがって、"属性レコード"とは、レコード・データ型として宣言した変数のことを指します。

たとえば、線の属性レコード、OG_LINE_ATTRの型定義は、次のようになります。

```
TYPE og_line_attr IS RECORD
(mask          NUMBER(1,1),
 startpt       og_point,
 endpt         og_point,
 arrowstyle    NUMBER(1,0)
);
```

このレコードには、線の始点および終点、矢印スタイルの属性が指定されています（マスク属性については、後で説明します）。

オブジェクトにおける属性は、いくつかある属性レコードのどれかで表すことができます。1つの属性をプログラムによって変更する場合は（たとえば、塗りパターンを変更する）、対応する属性レコード内の対応するフィールドの値を変更してから、その属性レコードをプロシージャまたはファンクションに渡す必要があります（プロシージャおよびファンクションによって、求めるアクションが実行されますが、その場合、必ず属性レコードを使用して、実行するアクションを正確にプロシージャまたはファンクションに反映させてください）。

属性クラス

属性レコードには、多くのオブジェクト・タイプに共通する属性を含むものと、1つのオブジェクト・タイプに固有の属性を含むものがあります。たとえば、名前はすべてのオブジェクトにあります。フォント・サイズはテキスト・オブジェクトにしかありません。

すべての属性は、次のクラスに編成されます。

- 一般
- 図形
- オブジェクト固有

一般属性

一般属性は、ほとんどのオブジェクト・クラスに適用されます。たとえば、名前または対応付けられたボタン・プロシージャ、親オブジェクトは、ほとんどのオブジェクトにあります。

図形属性

図形属性は、多くのオブジェクト・クラスに適用されますが、すべてのオブジェクト・クラスに適用されるわけではありません。図形属性が適用されるのは、図形オブジェクト（レイアウト・エディタ内のいずれかのグラフィカル・ツールを使用して作成できるオブジェクト）のみです。たとえば、図形属性の1つである"塗りカラー"を使用すると、四角形、円弧、記号などを描写できます。しかし、この属性を使用してイメージを描写しようとしても、イメージに塗りカラーは不要なため、意味がありません。同じ理由で、グループ・オブジェクトも図形属性を使用しての描写はできません。（グループとしては図形オブジェクトでなくても、そのグループの個々のコンポーネントは図形オブジェクトであることがあります。この場合には、図形属性をそのコンポーネントに適用できます）。

オブジェクト固有の属性

オブジェクト固有の属性は、特定のオブジェクト・クラスにのみ適用されます。たとえば、"開始角度"は円弧を描写するための属性であり、この属性を使用しても、四角形または線、イメージ、その他のオブジェクトを描写できません。同じように、グループ・オブジェクトの場合にそれを構成している"子の数"を知ることができますが、この属性を他のオブジェクト・クラスに使用しても意味がありません。Graphics Builderでは、次に挙げるオブジェクトに固有の属性を与えます。アプリケーション、円弧、チャート、グループ、イメージ、線、多角形、四角形、丸い四角形、記号、テキスト、ウィンドウ。

一般属性および図形属性には、それぞれ特有のビルトイン属性レコードが定義されています。また、オブジェクト固有の属性にも、個別の属性レコードが定義されています。

次のリストは、Graphics Builderオブジェクトと、それに対して有効な属性レコードです。

オブジェクト・クラス	属性レコード
------------	--------

アプリケーション	アプリケーション
円弧	一般 図形 円弧
チャート	一般 グループ チャート
チャート要素	図形 チャート要素
図表	図表
図形	一般 図形
グループ	一般 グループ
イメージ	一般イメージ
線	一般 図形 線
多角形	一般 図形 多角形
プリンタ	プリンタ
問合せ	問合せ
四角形	一般 図形 四角形
丸い四角形	一般 図形 丸い四角形
サウンド	サウンド
記号I	一般 図形 記号
テキスト	一般 図形 テキスト
タイマー	タイマー
ウィンドウ	ウィンドウ

結合属性レコード

先に述べた属性レコードに加えて、Graphics Builderでは、"結合属性レコード"も定義されています。結合属性レコードでは、オブジェクトを描写するのに必要なすべての属性レコードが1つの

変数に結合されます。その名前が示すとおり、これは別のレコードですが、その中の各フィールドは一般属性レコードまたは図形属性レコード、オブジェクト固有の属性レコードのいずれかです。したがって、個別の属性レコードをいくつも使用してオブジェクトのそれぞれの属性クラスを示さなくても、1つの結合属性レコードを使用すると、あるオブジェクトの属性をすべて制御できます。

たとえば、四角形の結合属性レコードには、一般属性レコード、図形属性レコード、四角形属性レコードを表す3つのフィールドがあり、イメージの結合属性レコードには、一般属性レコードとイメージ属性レコードを表す2つのフィールドがあります。

次は、線オブジェクトの結合属性レコードOG_LINE_CAのタイプ定義です。

```
TYPE og_line_ca IS RECORD
(line_caob  og_generic_attr, /*generic attribute record*/
 line_caoh  og_graphic_attr, /* graphicattribute record */
 line_caol  og_line_attr    /* line attributerecord */
);
```

この結合属性レコードには、一般属性レコード、図形属性レコード、線属性レコードを表す3つのフィールドがあります。

マスク属性

各属性レコード（結合属性レコードは除く）には、“マスク”と呼ばれる数値フィールドがあります。このフィールドの値は、属性レコード内の変更（設定）または検査（取得）する属性を示します。属性レコードをプロシージャまたはファンクションの引数として使用すると、そのプロシージャやファンクションではマスクを使用して考慮すべき属性が判別されます。

たとえば、図形属性レコードにffcolor属性を設定することによって、あるオブジェクトの塗りフォアグラウンド・カラーを変更してから、その属性レコードとオブジェクトのハンドルを引数としてOG_SET_ATTRプロシージャに渡すと想定してください。この場合、プロシージャでは、設定する属性がどれなのかが判別できません。つまり、オブジェクトの図形属性すべてを変更する必要があるのか、オブジェクトの図形属性の一部のみを変更する必要があるのかを認識できません。これを認識するため、プロシージャによって、属性レコードのマスク属性が調べられます。

マスク属性の値は、プロシージャやファンクションで使用する必要がある属性レコード内の属性を示します。この値を、“マスク値”と呼びます。

マスク定数

属性レコードに対応するマスク値を判別しやすくするため、Graphics Builderでは、各属性に“マスク定数”と呼ばれる個別のビルトイン数値定数が対応付けられています。

次は、線の属性レコードのリストですが、この場合はマスク定数を付記しています。

```
TYPE og_line_attr IS RECORD                                Mask Constants:
(mask              NUMBER(1,0),
 startpt          og_point,                                OG_STARTPT_LINEA
 endpt            og_point,                                OG_ENDPT_LINEA
 arrowstyle       NUMBER(1,0)                               OG_ARROWSTYLE_LINEA
);

                                OG_ALL_LINEA
                                OG_NONE_LINEA
```

属性レコード内の属性から使用する属性を決定したら、それらの属性に対応付けられているマスク定数の合計を計算してください。その結果が、それらの属性を表すマスク値になります。属性レコード内のマスク属性をこのマスク値に設定すると、その属性レコードを渡したプロシージャまたはファンクションでは、それらの属性のみが考慮されます。

たとえば、前述の線の属性レコードのstartpt属性を変更する場合は、最初に、このタイプの変数を宣言します。

```
my_variable og_line_attr;
```

続いて、startpt属性の新しい値を設定します。

```
my_variable.startpt :=minew_point;
```

最後に、マスクを設定して、新しい始点の設定を指示します。

```
my_variable.mask := OG_STARTPT_LINEA;
```

(この一連の処理は、プロシージャやファンクションに使用される属性レコードを用意しているにすぎません。この処理が実際にオブジェクトを変更する場合にどう関係しているのかを把握するには、特定のプロシージャやファンクションについての説明を参照してください)。

線の始点と終点に新しい値を設定する場合には、マスクを設定してそのことを指示する必要があります。この場合、これら2つの属性のマスク定数の合計が、この場合の適切なマスク値となります。

```
my_variable.mask := OG_STARTPT_LINEA + OG_ENDPT_LINEA;
```

各属性のためのマスク定数に加えて、すべての属性レコードには、プロシージャやファンクションがすべての属性を使用することを指示する属性と、どの属性も使用しないことを指示する属性が追加されています。線属性レコードでは、これらのマスク定数は、OG_ALL_LINEAおよびOG_NONE_LINEAです。

これらのマスク定数は数値であり、数値として処理できることを知っておいてください。これらを加算して複数の属性を示すのみでなく、減算することもできます。たとえば、終点を除くすべての属性がプロシージャやファンクションに影響されることを示す場合は、マスク値を次のように設定できます。

```
my_variable.mask := OG_ALL_LINEA - OG_ENDPT_LINEA;
```

場合によっては、1つの属性レコード内で複数の属性を表すのに、同じマスク定数を使用することがあります。マスク値を計算する場合にマスク定数を使用すると、その定数で表されている属性はすべて、属性レコードが渡されるプロシージャまたはファンクションによって使用されます。

結合属性レコード内のマスク

前述したとおり、すべての属性レコードには、マスク属性がありますが、結合属性レコードにはありません。結合属性レコードを引数としてプロシージャに渡すと、プロシージャが使用するマスクは、その結合属性レコードに入っている各属性レコードのマスクになります。

たとえば、ある変数を線の結合属性レコードとして宣言する場合（線の結合属性レコードは、一般属性レコードおよび図形属性レコード、線固有の属性レコードで構成されることを思い出してください）、まず次のように指定します。

```
comb_variable    og_line_ca;
```

次に、このレコードの属性をいくつか変更します。一般属性レコードでは、どの値も変更せず、図形属性レコードでは、線種属性と端形式属性の値を変更し、線属性レコードでは、矢印スタイル属性の値のみを変更するとします。この場合、次のような文を使用できます。

```
comb_variable.line_caoh.dashstyle := minew_dashstyle;
comb_variable.line_caoh.capstyle  := minew_capstyle;
comb_variable.line_caol.arrowstyle := minew_arrowstyle;
```

変更を実行するプロシージャにこの結合属性レコードを渡すには、それぞれの属性レコードにマスクを設定して、そのレコード内でプロシージャが使用する必要のある属性を指示する必要があります。

```
comb_variable.line_caob.mask := OG_NONE_GENERICA;
comb_variable.line_caoh.mask := OG_DASHSTYLE_GRAPHICA +
                                OG_CAPSTYLE_GRAPHICA;
comb_variable.line_caol.mask := OG_ARROWSTYLE_LINEA;
```

属性レコード内のどの属性も使用しない場合でも、必ず結合属性レコード内のすべての属性レコードにマスクを設定する必要があります。この場合、マスクは、どの属性も使用しないことを指示したマスク定数に設定します。

個々の属性レコードに対してマスクを設定したら、結合属性レコードをプロシージャやファンクションに渡すことが可能になります。属性レコードのマスク値を使用した場合に限り、使用する属性をプロシージャやファンクションに知らせることができます。

作成可能、設定可能、取得可能な属性

以下のリストにある各属性のとなりには、3文字までのアルファベット文字が指定されています。

アルファベット文字	意味
-----------	----

C	作成可能な属性を示す。つまり、Graphics Builderでは、該当する属性を含むオブジェクトを最初に作成したとき、その属性に割り当てた値が認識されます。属性が作成可能でないと、Graphics Builderにより、オブジェクトの作成時にデフォルト値が提供されます。
S	設定可能な属性を示す。つまり、適切なGraphics Builderビルトイン・サブプログラムを起動して該当する属性の値を設定できます。
G	取得可能な属性を示す。つまり、適切なGraphics Builderビルトイン・サブプログラムを起動して該当する属性の値を取得できます。

ショートカット・ビルトイン

先に説明した属性レコードによる処理方法以外にも、Graphics Builderでは、オブジェクトの作成処理とその属性の取得または設定処理を簡単にする一連のビルトイン・サブプログラムを使用することもできます。こういったショートカット・サブプログラムを使用すると、オブジェクトの1つの属性を設定または取得できます。詳細は、Graphics Builder「ビルトインの概要」を参照してください。

たとえば、属性レコードによる処理方法を使用してオブジェクトの塗りパターンと枠パターンを設定する場合には、新しい塗りパターンの設定および適切なマスクの設定、OG_SET_ATTRのコールを次のように実行する必要があります。

```
PROCEDURE attr_rec_approach (my_obj OG_OBJECT) IS
  my_recog_graphic_ca;
BEGIN
  my_rec.graphic_caoh.fillpatt:='gray50';
  my_rec.graphic_caoh.edgepatt:='kangaroo';
  my_rec.graphic_caob.mask:=OG_NONE_GENERICA;
my_rec.graphic_caoh.mask:=OG_FILLPATT_GRAPHICA+
OG_EDGEPATT_GRAPHICA;
  og_set_attr(my_obj, my_rec);
END;
```

ショートカットを使用すると、前述と同じプロセスをたった2つのプロシージャのコールで実行できます。

```
PROCEDURE shortcut_approach (my_obj OG_OBJECT) IS
BEGIN
  og_set_fillpatt(my_obj, 'gray50');
  og_set_edgepatt(my_obj, 'kangaroo');
END;
```

メリット

属性レコードのかわりにショートカットを使用する場合には、次の利点があります。

- 必要なPL/SQLコードが少なくて済むため、開発時間を削減できる。
- プログラム単位の読取りと認識が簡単になる。

デメリット

属性レコードのかわりにショートカットを使用する場合には、次の欠点があります。

- 効率が悪くなる。Graphics Builderでは、属性レコードが内部で使用されるため、ショートカットをコールするたびに新しい内部属性レコードを定義し挿入する必要があります。また、複数の"set"ルーチンを実行する時間は、1つのルーチンを実行する時間に比べて長くなります。前述の例では、最初のプロシージャ("set"が1つ)に要する時間は、2番目のプロシージャ("set"が2つ)の約半分です。
- 複数のショートカットをコールして必要な属性をすべて設定すると、アプリケーションのパフォーマンスに悪影響を及ぼすおそれがあるので、アプリケーションはデフォルト設定に依存することになります。

アプリケーション属性レコード

アプリケーション属性レコードは、現行のアプリケーションで使用可能な属性で構成されます。

```
TYPE og_app_attr IS RECORD
(mask
(cursor          NUMBER(3,0),
hscreen_res     NUMBER(5,0),
vscreen_res     NUMBER(5,0),
hlayout_res     NUMBER(10,0),
vlayout_res     NUMBER(10,0),
platform        NUMBER(1,0),
username        VARCHAR2(255),
password        VARCHAR2(255),
connection      VARCHAR2(255)
);

Mask Constants:
OG_CURSOR_APPA
OG_SCREEN_RES_APPA
OG_SCREEN_RES_APPA
OG_LAYOUT_RES_APPA
OG_LAYOUT_RES_APPA
OG_PLATFORM_APPA
OG_USERNAME_APPA
OG_PASSWORD_APPA
OG_CONNECTION_APPA
OG_ALL_APPA
OG_NONE_APPA
```

	属性	説明
SG	cursor	使用するマウス・カーソルの名前。この属性の値として、次の文字列のいずれかを指定できます。 default insertion crosshair help busy 詳細は、使用しているマニュアルを参照してください。この属性に無効な値を設定すると、その値は"default"とみなされます。
G	hscreen_res	画面の水平解像度。この値は、画面の水平方向1インチ内の画面解像度(ピクセル)単位の数です。

G	vscreen_res	画面の垂直解像度。この値は、画面の垂直方向1インチ内の画面解像度単位（ピクセル）の数です。
G	hlayout_res	レイアウトの水平解像度。この値は、レイアウトの水平方向の1インチ内のレイアウト単位の数です。
G	vlayout_res	レイアウトの垂直解像度。この値は、レイアウトの垂直方向の1インチ内のレイアウト単位の数です。
G	platform	Graphics Builderが実行されるプラットフォーム。この属性の値として、次のビルトイン定数のいずれかを指定できます。
		OG_MACINTOSH_PLATFORM プラットフォームがApple Macintoshになる。
		OG_MOTIF_PLATFORM プラットフォームがOSF/MOTIFになる。
		OG_MSWINDOWS_PLATFORM プラットフォームがMicrosoft Windowsになる。
		OG_PM_PLATFORM プラットフォームがPresentation Managerになる。
		OG_X_PLATFORM プラットフォームがX Window Systemになる。
G	username	現行のデータベース接続用のユーザー名。接続していないと、この属性はNULLになります。
G	password	現行のデータベース接続用のパスワード。接続していない場合や、Keep_Password作業環境を「いいえ」に設定している場合、この属性はNULLになります。
G	connection	現行のデータベース接続用のデータベース接続文字列。接続していないと、この属性はNULLになります。

円弧結合属性レコード

円弧結合属性レコードは、一般属性レコードおよび図形属性レコード、円弧属性レコードで構成されます。

```
TYPE og_arc_ca IS RECORD
(arc_caob  og_generic_attr, /* generic */
 arc_caoh  og_graphic_attr, /* graphic */
 arc_caoa  og_arc_attr /* arc */
);
```

円弧属性レコード

記号属性レコードは、記号オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_arc_attr IS RECORD
(mask      NUMBER(1,0),
 basearc   og_arc, OG_BASEARC_ARCA
 arcfill   NUMBER(1,0),
 OG_ARCFILL_ARCA
Mask Constants:
```



```
closed    BOOLEAN OG_CLOSED_ARCA
);
```

	属性	説明
CSG	basearc	左上角のx座標とy座標、および楕円から円弧をカットする際に基準として使用する四角形の高さと幅。
CSG	arcfill	円弧の形状。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_CHORD_ARCFILL 円弧の形状が弦型になる。 OG_PIE_ARCFILL 円弧の形状が完全な扇型になる。
CSG	closed	円弧の閉じ形状。この属性の値として、次のいずれかを指定できます。
		TRUE 円弧が閉じた形状になる。
		FALSE 円弧が開いた形状になる。

値軸結合属性レコード

```
TYPE og_contaxis_ca IS RECORD
(ca_axis  og_axis_attr, /* generic axis */
 ca_cont  og_contaxis_attr /* continuous axis */
);
```

値軸属性レコード

```
TYPE og_contaxis_attr IS RECORD      Mask Constants:
(mask      NUMBER(4,0),
 automin   BOOLEAN,                    OG_MINIMUM_CONTAXISA
 minimum   NUMBER(6),
```

	属性	説明
SG	automin	軸最小値を「自動」に設定するかどうかを指定します。
SG	minimum	最小軸値を指定します（ただし、 <i>automin</i> がFALSEの場合のみ）。
SG	autostep	主目盛間隔を「自動」に設定するかどうかを指定します。
SG	step	主目盛間隔を指定します（ただし、 <i>autostep</i> がFALSEの場合のみ）。
SG	automax	軸最大値を「自動」に設定するかどうかを指定します。
SG	maximum	最大軸値を指定します（ただし、 <i>automax</i> がFALSEの場合のみ）。
SG	scale	軸のスケーリングに使用されるアルゴリズムを指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。
		OG_LINEAR_SCALE 最小軸値から最大軸値まで固定した間隔で軸をスケーリングします。

		OG_LOG_SCALE 最小軸値から最大軸値までの間隔を決めるために、対数アルゴリズム（10の累乗に基づく）を使用して軸をスケーリングします。
		OG_PCT_SCALE データ値をpct_ofで指定した数と対応させて表示するように軸をスケーリングします。
SG	pct_of	相対スケール変更係数を指定する（scaleがOG_PCT_SCALEに設定されている場合）。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_MAXIMUM_PCTOF 各データ値を最大データ値のパーセントで表示する。 OG_MINIMUM_PCTOF 各データ値を最小データ値のパーセントで表示する。 OG_SUM_PCTOF 各データ値をすべてのデータ値の合計のパーセントで表示する。
SG	pct_by	pct_ofスケール値をどう計算するかを指定する。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_CATEGORY_PCTBY 各データ値のパーセントを他の項目内の同じフィールドのデータ値と対応させて計算する。 OG_FIELD_PCTBY 各データ値のパーセントを他のフィールドの同じ項目内のデータ値と対応させて計算する。
SG	numfmt	軸目盛きざみラベルの数値書式を指定します。これは有効なSQLフォーマット文字列である必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

日付軸結合属性レコード

```
TYa_axis og_axis_attr, /* generic axis */
ca_date og_dateaxis_attr /*PE og_dateaxis_ca IS RECORD
(c date axis */
);
```

日付軸属性レコード

```
TYPE og_dateaxis_attr IS RECORD      マスク定数:
(mask          NUMBER(5,0),
automin        BOOLEAN,              OG_MINIMUM_DATEAXISA
minimum        DATE,                 OG_MINIMUM_DATEAXISA
autostep        BOOLEAN,              OG_STEP_DATEAXISA
step           NUMBER(2,0),           OG_STEP_DATEAXISA
automax         BOOLEAN,              OG_MAXIMUM_DATEAXISA
maximum        DATE,                 OG_MAXIMUM_DATEAXISA
firstmon        NUMBER(2,0),          OG_FIRSTMON_DATEAXISA
```

```
skipwknds  BOOLEAN,          OG_SKIPWKNDSDATEAXISA
labels     NUMBER(4,0),       OG_LABELSDATEAXISA
dayfmt     NUMBER(1,0),       OG_DAYFMTDATEAXISA
monthfmt   NUMBER(1,0),       OG_MONTHFMTDATEAXISA
qtrfmt     NUMBER(1,0),       OG_QTRFMTDATEAXISA
yearfmt    NUMBER(1,0),       OG_YEARFMTDATEAXISA
custfmt    VARCHAR2(255)      OG_CUSTMTDATEAXISA
);

OG_ALLDATEAXISA
OG_NONEDATEAXISA
```

	属性	説明
SG	automin	軸最小値を「自動」に設定するかどうかを指定します。
SG	minimum	最小軸値を指定します（ただし、 <i>automin</i> がFALSEの場合のみ）。
SG	autostep	主目盛間隔を「自動」に設定するかどうかを指定します。
SG	step	主目盛間隔を指定します（ただし、 <i>autostep</i> がFALSEの場合のみ）。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_SECOND_STEP OG_MINUTE_STEP OG_HOUR_STEP OG_DAY_STEP OG_WEEK_STEP OG_MONTH_STEP OG_QUARTER_STEP OG_YEAR_STEP
SG	automax	軸最大値を「自動」に設定するかどうかを指定します。
SG	maximum	最大軸値を指定します（ただし、 <i>automax</i> がFALSEの場合のみ）。

SG	firstmonth	<p>新しい年度が始まる最初の月です。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_JAN_MONTH OG_FEB_MONTH OG_MAR_MONTH OG_APR_MONTH OG_MAY_MONTH OG_JUN_MONTH OG_JUL_MONTH OG_AUG_MONTH OG_SEP_MONTH OG_OCT_MONTH OG_NOV_MONTH OG_DEC_MONTH</p>
SG	skipweekends	<p>軸の値を計算するときに週末を無視するかどうかを指定します。</p>
SG	labels	<p>主目盛きざみと目盛きざみラベルが表示される軸に沿った主目盛間隔のみでなく、目盛きざみラベルの表示も指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_NO_LABELS OG_SECOND_LABELS OG_MINUTE_LABELS OG_HOUR_LABELS OG_AMPM_LABELS OG_DAY_LABELS OG_DAYOFWEEK_LABELS OG_WEEK_LABELS OG_MONTH_LABELS OG_QUARTER_LABELS OG_YEAR_LABELS OG_CUSTOM_LABELS (この値をlabelsに設定した場合は、必ず customfmt属性にカスタム日付書式を指定してください)。</p>
SG	dayfmt	<p>曜日ラベルの表示形式を決めます。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_FIRSTLETTER_FMT OG_THREELETTER_FMT</p>

SG	monthfmt	月ラベルの表示形式を決めます。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_FIRSTLETTER_FMT OG_THREELETTER_FMT
SG	quarterfmt	四半期ラベルの表示形式を決めます。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_ARABIC_FMT OG_ROMAN_FMT
SG	yearfmt	年ラベルの表示形式を決めます。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_FOURDIGIT_FMT OG_TWODIGIT_FMT
SG	custfmt	軸目盛きざみラベルのカスタム日付書式です。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

項目軸結合属性レコード

```
TYPE og_discaxis_ca IS RECORD
(ca_axis  og_axis_attr, /* generic axis */
 ca_disc  og_discaxis_attr /* discrete axis */
);
```

項目軸属性レコード

```
TYPE og_discaxis_attr IS RECORD
(mask      NUMBER(3,0),
 automin   BOOLEAN,
 mincat     NUMBER(10,0),
 automax    BOOLEAN,
 maxcat     NUMBER(10,0),
 numfmt     VARCHAR2(255),
 datefmt    VARCHAR2(255)
);
```

Mask Constants:

OG_MINCAT_DISCAXISA
OG_MINCAT_DISCAXISA
OG_MAXCAT_DISCAXISA
OG_MAXCAT_DISCAXISA
OG_NUMFMT_DISCAXISA
OG_DATEFMT_DISCAXISA

OG_ALL_DISCAXISA
OG_NONE_DISCAXISA

	属性	説明
SG	automin	軸に表示する最小項目数を「自動」に設定するかどうかを指定します。
SG	mincat	軸上に表示される最小項目数を指定します（autominがFALSEの場合）。

SG	automax	軸に表示する最大項目数を「自動」に設定するかどうかを指定します。
SG	maxcat	軸に表示する最大項目数を指定します（ただし、 <i>automax</i> がFALSEの場合のみ）。
SG	numfmt	軸目盛きざみラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。
SG	datefmt	軸目盛きざみラベルの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

軸属性レコード

```
TYPE og_axis_attr IS RECORD
(mask          NUMBER(5,0),
axistype       NUMBER(1,0),
custlabel      VARCHAR2(255),
position       NUMBER(1,0),
direction      NUMBER(1,0),
tickpos        NUMBER(1,0),
ticklabelrot   NUMBER(1,0),
minorct        NUMBER(1,0),
majorticks     BOOLEAN,
minorticks     BOOLEAN,
majorgrid      BOOLEAN,
minorgrid      BOOLEAN,
axislabel      BOOLEAN,
ticklabels     BOOLEAN
);
```

Mask Constants:

```
OG_AXISTYPE_AXIS
OG_CUSTLABEL_AXIS
OG_POSITION_AXIS
OG_DIRECTION_AXIS
OG_TICKPOS_AXIS
OG_TICKLABELROT_AXIS
OG_MINORCT_AXIS
OG_MAJORTICKS_AXIS
OG_MINORTICKS_AXIS
OG_MAJORGRID_AXIS
OG_MINORGRID_AXIS
OG_AXISLABEL_AXIS
OG_TICKLABELS_AXIS

OG_ALL_AXIS
OG_NONE_AXIS
```

	属性	説明
SG	axistype	この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_CONTINUOUS_AXIS OG_DATE_AXIS OG_DISCRETE_AXIS
SG	custlabel	軸に沿って表示するラベルのテキストを指定します。

SG	position	<p>どのチャート枠に沿って軸を表示するかを指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_BOTTOM_POSITION OG_LEFT_POSITION OG_RIGHT_POSITION OG_TOP_POSITION</p>
SG	direction	<p>増分値や連続した項目を軸に沿ってどの方向に設定するかを指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_DOWN_DIRECTION OG_LEFT_DIRECTION OG_RIGHT_DIRECTION OG_UP_DIRECTION</p>
SG	tickpos	<p>主目盛きざみと補助目盛きざみをどう表示するかを指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_CROSS_TICKPOS OG_INSIDE_TICKPOS OG_OUTSIDE_TICKPOS</p>
SG	ticklabelrot	<p>この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_CCW_ROTATION OG_CW_ROTATION OG_NO_ROTATION</p>
SG	minorct	それぞれの主目盛きざみと主目盛きざみの間に定義される補助目盛きざみ数です。
SG	majorticks	主目盛きざみを主目盛間隔で表示するかどうかを指定します。
SG	minorticks	「補助きざみ数」で設定された値の指定通りに補助目盛きざみを表示するかどうかを指定します。
SG	majorgrid	それぞれの主目盛きざみにグリッドを表示するかどうかを指定します。
SG	minorgrid	それぞれの補助目盛きざみにグリッドを表示するかどうかを指定します。
SG	axislabel	軸に沿って値を識別する項目ラベルを表示するかどうかを指定します。
SG	ticklabels	軸に沿って値を識別する項目ラベルを表示するかどうかを指定します。

チャート結合属性レコード

チャートは、グループ・オブジェクトと同様に扱われ、線、四角形、テキストなどで構成されます。したがって、チャート結合属性レコードを使用すると、チャートに固有の属性にアクセスできるのみでなく、グループ属性にもアクセスできます。また、チャートそのものは図形オブジェクトではないので（チャートを構成するオブジェクトは図形オブジェクトであっても）、このレコードを使用しても図形属性にはアクセスできません。チャートの個々の要素の図形属性を設定する場合には、チャート要素属性レコードを使用してください（後の説明を参照）。

チャートが動的チャートとして指定されている場合に限り、このレコードを使用して、チャートの属性にアクセスできます。チャートがアートワークの場合は、そのチャートはグループ・オブジェクトとみなされ、チャート・オブジェクトとして扱われません。プログラムによって作成されたチャートは、動的チャートとなります。

チャート結合属性レコードは、一般属性レコード、グループ属性レコードおよびチャート属性レコードで構成されます。

```
TYPE og_chart_ca IS RECORD
(chart_caob  og_generic_attr, /* generic */
 chart_caog  og_group_attr,   /* graphic */
 chart_caoc  og_chart_attr    /* chart */
);
```

チャート属性レコード

チャート属性レコードは、チャート・オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_chart_attr IS RECORD
(mask          NUMBER(4,0),
 frame         og_rectangle,
 template      og_template,
 query         og_query,
 title         VARCHAR2(255),
 autoupdate    BOOLEAN,
 rangeflag     BOOLEAN,
 startrow      NUMBER(10,0),
 endrow        NUMBER(10,0),
 filter        VARCHAR2(255)
);
```

Mask Constants:
OG_FRAME_CHARTA
OG_TEMPLATE_CHARTA
OG_QUERY_CHARTA
OG_TITLE_CHARTA
OG_AUTOUPDATE_CHARTA
OG_ROWS_CHARTA
OG_ROWS_CHARTA
OG_ROWS_CHARTA
OG_FILTER_CHARTA
OG_ALL_CHARTA
OG_NONE_CHARTA

	属性	説明
CSG	frame	チャートの枠のx座標とy座標、および高さ、幅です（レイアウト単位）。
CSG	template	チャートに使用するテンプレートのハンドルです。
CSG	query	チャートに使用する問合せのハンドルです。

CSG	title	チャートのタイトルです。
CSG	autoupdate	問合せの実行時にチャートを自動的に更新するように指定します。
CSG	rangeflag	チャートに表示する問合せ行の数を、 <i>startrow</i> と <i>endrow</i> で指定した範囲に制限するかどうかを指定します。
CSG	startrow	チャートに表示するデータの最初の行です。最初のデータ行は0、2番目の行は1となり、以下同様です。
CSG	endrow	チャートに表示するデータの最後の行です。
CSG	filter	問合せのフィルタ・トリガー・プロシージャの名前です。

チャート要素結合属性レコード

チャート要素は、フィールドの単一の値を表す図形オブジェクトです。たとえば、棒グラフの棒や円グラフの円弧がチャート要素です。この結合属性レコードをOG_SET_ATTRプロシージャとともに使用すると、チャート要素の属性を変更できます。

チャート要素結合属性レコードは、図形属性レコードおよびチャート要素属性レコードで構成されます。

```
TYPE og_chelement_ca IS RECORD
(chement_cagr  og_graphic_attr,    /* graphic */
 chelement_cace og_chelement_attr /* chart element */
);
```

チャート要素属性レコード

チャート要素属性レコードは、チャート要素にのみ使用可能な属性で構成されます。

```
TYPE og_chelement_attr IS RECORD  Mask Constants:
(mask      NUMBER(1,0),
 button    og_buttonproc,          OG_BUTTON_CHELEMENTA
 events    NUMBER(2,0),            OG_BUTTON_CHELEMENTA
 explosion NUMBER(10,0),           OG_EXPLOSION_CHELEMENTA
 name      VARCHAR2(255)           OG_NAME_CHELEMENTA
);

OG_ALL_CHELEMENTA
OG_NONE_CHELEMENTA
```

	属性	説明
S	button	このチャート要素と対応付けるボタン・プロシージャのハンドルです。要求したマウス・イベントをこのプロシージャで確実に受信できるようにするには、必ずイベント属性を正しく設定してください。

S	events	<p>ボタン・プロシージャが受け取るマウス・イベントのタイプです。この属性の値として、次のビルトイン定数のいずれかを指定できます。プロシージャで複数のイベント・タイプを確実に受信できるようにするには、この属性を、要求したイベントの定数の合計となるように設定します。</p> <p>OG_NO_EVENTS どのマウス・イベントも受信しない。</p> <p>OG_MOUSE_DOWN マウス・ボタンを押すイベントのみ受信する。</p> <p>OG_MOUSE_MOVE_DOWN マウス・ボタンを押したままマウスを移動させるイベントのみ受信する。</p> <p>OG_MOUSE_UP マウス・ボタンを放すイベントのみ受信する。</p> <p>OG_MOUSE_MOVE_UP マウス・ボタンを放したまま、マウスを移動させるイベントのみ受信する。</p>
S	explosion	<p>チャート要素（円グラフの円弧）を突出させる距離。チャートのx半径およびy半径の割合で表されます。この属性は、円グラフのチャートにしか使用できません。また、チャート要素として項目を指定するとすべての円グラフが同じ割合で展開されます。したがって、突出値を指定する属性レコードは、個々のフィールドと対応付ける必要があります。</p>
S	name	<p>チャート要素の名前です。チャート要素の名前を取得する場合は、一般属性レコードを使用してください。</p>

図表属性レコード

図表属性レコードは、現行の図表にのみ使用可能な属性で構成されます。

```
TYPE og_display_attr IS RECORD                                Mask Constants:
(mask                NUMBER(2,0),
opentrigger          VARCHAR2(255),                            OG_OPENTRIGGER_DISPLAYA
closetrigger          VARCHAR2(255),                            OG_CLOSETRIGGER_DISPLAYA
width                 NUMBER(10,0),                             OG_SIZE_DISPLAYA
height                NUMBER(10,0),                             OG_SIZE_DISPLAYA
dateformat            VARCHAR2(255)                             OG_DATEFORMAT_DISPLAYA
);
                                                                OG_ALL_DISPLAYA
                                                                OG_NONE_DISPLAYA
```

	属性	説明
SG	opentrigger	図表のオープン・トリガーの名前。
SG	closetrigger	図表のクローズ・トリガーの名前。
SG	width	レイアウトの幅です（レイアウト単位）。
SG	height	レイアウトの高さです（レイアウト単位）。
SG	dateformat	パラメータの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

軸フィールド・テンプレート結合属性レコード

```
TYPE og_axisftemp_ca IS RECORD
(ca_ftemp   og_ftemp_attr,      /* generic field template */
 ca_aftemp  og_axisftemp_attr /* axis field template */
);
```

軸フィールド・テンプレート属性レコード

```
TYPE og_axisftemp_attr IS RECORD      Mask Constants:
(mask                NUMBER(3,0),
 plottype            NUMBER(3,0),      OG_PLOTTYPE_AXISFTEMPA
 linestyle           NUMBER(1,0),      OG_LINESTY_AXISFTEMPA
 labelrot            NUMBER(1,0),      OG_LABELROT_AXISFTEMPA
 plotpos             NUMBER(1,0),      OG_PLOTPOS_AXISFTEMPA
 overlap             NUMBER(3),        OG_OVERLAP_AXISFTEMPA
 axis                NUMBER(1,0),      OG_AXIS_AXISFTEMPA
 curvefit            NUMBER(1,0),      OG_CURVEFIT_AXISFTEMPA
);

OG_ALL_AXISFTEMPA
OG_NONE_AXISFTEMPA
```

	属性	説明
SG	plottype	<p>このフィールドをチャート上に表示するために使用する要素を指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_NONE_PLOTTYPE OG_BAR_PLOTTYPE OG_LINE_PLOTTYPE OG_SYMBOL_PLOTTYPE OG_FILL_PLOTTYPE OG_SPIKE_PLOTTYPE OG_LABEL_PLOTTYPE</p>
SG	linestyle	<p>線表示形式を含む各フィールドのデータ・ポイント間を結ぶために使用する線の形式を指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_SPLINE_LINESTYLE OG_STEP_LINESTYLE OG_STRAIGHT_LINESTYLE</p>
SG	labelrot	<p>ラベル表示形式を含む各フィールドのラベルの回転角度を指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p>

SG	plotpos	各項目において2つ以上のフィールドのデータ値間の関係を指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_NORMAL_PLOTPOS OG_FROMPREV_PLOTPOS OG_STACKED_PLOTPOS
SG	overlap	複数のフィールドからのデータ値を横棒チャートや縦棒チャートで表示する棒どうしがそれぞれオーバーラップする割合を指定します。
SG	axis	フィールドの表示方法を決定するのに必要なデータ値が比較される軸を指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_X_AXIS OG_Y1_AXIS OG_Y2_AXIS
SG	curvefit	カーブ調整をチャートに適用するかどうか、また適用する場合にはどのアルゴリズムを使用するかを指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_NO_CURVEFIT OG_LINEAR_CURVEFIT OG_LOG_CURVEFIT OG_EXP_CURVEFIT OG_POWER_CURVEFIT

フィールド・テンプレート属性レコード

```
TYPE og_ftemp_attr IS RECORD
(mask          NUMBER(3,0),
 name          VARCHAR2(255),
 root          OG_OBJECT,
 colorrot      NUMBER(1,0),
 numfmt        VARCHAR2(255),
 datefmt       VARCHAR2(255)
);
```

Mask Constants:

OG_NAME_FTEMPA
OG_ROOT_FTEMPA
OG_COLORROT_FTEMPA
OG_NUMFMT_FTEMPA
OG_DATEFMT_FTEMPA

OG_ALL_FTEMPA
OG_NONE_FTEMPA

	属性	説明
SG	name	フィールド・テンプレートの名前です。
G	root	フィールド・テンプレートが含まれるチャート・テンプレートのハンドル。

SG	colorrot	Graphics Builderがカラー・パレットやパターン・パレットを自動的に回転させて、このフィールド・テンプレートを使用する各フィールドに固有のパレット設定を選択するかどうかを指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_NO_COLORROT OG_AUTO_COLORROT OG_COLOR_COLORROT OG_PATTERN_COLORROT OG_BOTH_COLORROT
SG	numfmt	フィールド・ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列です必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。
SG	datefmt	フィールド・ラベルの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

軸枠結合属性レコード

```
TYPE og_axisframe_ca IS RECORD
(ca_frame
```

軸枠属性レコード

```
TYPE og_axisframe_attr IS RECORD  Mask Constants:
(mask          NUMBER(3,0),
reflinect     NUMBER(3,0),        OG_REFLINECT_AXISFRAMEA
basevalue     NUMBER(1,0),        OG_BASEVALUE_AXISFRAMEA
cust_num      NUMBER(6),          OG_BASEVALUE_AXISFRAMEA
cust_date     DATE,               OG_BASEVALUE_AXISFRAMEA
base_axis     NUMBER(1,0),        OG_BASEAXIS_AXISFRAMEA
catwidth      NUMBER(3,0),        OG_CATWIDTH_AXISFRAMEA
second_y      BOOLEAN             OG_SECONDY_AXISFRAMEA
);

OG_ALL_AXISFRAMEA
OG_NONE_AXISFRAMEA
```

	属性	説明
G	reflinect	チャート・テンプレートに含まれる参照線の数。

SG	baseline_value	値軸に沿ってフィールドを表示する際に開始点として使用される値です。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_MIN_BASELINE OG_MAX_BASELINE OG_ZERO_BASELINE OG_CUSTOM_BASELINE
SG	custom_num	数値軸タイプの基本線の値を指定します。この値は、自動的にOG_CUSTOM_BASELINEに設定されます。
SG	custom_date	日付軸タイプの日付の値を指定します。この値は、自動的にOG_CUSTOM_BASELINEに設定されます。
SG	baseline_axis	位置を求めるために基本線の値が比較される軸を指定します。
SG	catwidth	これは、「ストライプ幅」の割合で示した、横棒グラフや縦棒グラフ内の棒の幅です。ストライプ幅は、棒が互いに重なることなくとれる棒の最大の幅で、項目軸の長さを、表示される棒の数で割って決定されます。
SG	second_y	第2-Y軸をチャート内に表示するかどうかを指定します。

枠属性レコード

```
TYPE og_frame_attr IS RECORD
(mask          NUMBER(4,0),
 name          VARCHAR2(255),
 frametype     NUMBER(1,0),
 ftempct       NUMBER(5,0),
 root          OG_OBJECT,
 depthsize     NUMBER(1,0),
 shadowsize    NUMBER(1,0),
 shadowdir     NUMBER(1,0),
 plotframe     BOOLEAN,
 legend        BOOLEAN,
 legendcolct   NUMBER(3,0)
);
```

Mask Constants:

OG_NAME_FRAMEA
OG_FRAMETYPE_FRAMEA
OG_FTEMPCT_FRAMEA
OG_ROOT_FRAMEA
OG_DEPTHSIZE_FRAMEA
OG_SHADOWSIZE_FRAMEA
OG_SHADOWDIR_FRAMEA
OG_PLOTFRAME_FRAMEA
OG_LEGEND_FRAMEA
OG_LEGENDCOLCT_FRAMEA

OG_ALL_FRAMEA
OG_NONE_FRAMEA

	属性	説明
SG	name	チャート・テンプレートの名前です。
G	frametype	このテンプレートで表されるチャートのタイプ。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_AXIS_FRAMETYPE OG_PIE_FRAMETYPE OG_TABLE_FRAMETYPE

G	ftempct	チャート・テンプレートに含まれるフィールド・テンプレートの数。
G	root	チャート・テンプレートのハンドル。
SG	depthsize	チャート枠と要素を3次元表示するために、それらが描画される深さのサイズを指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_NONE_DEPTHSIZE OG_SMALL_DEPTHSIZE OG_MEDIUM_DEPTHSIZE OG_LARGE_NONE_DEPTHSIZE OG_XLARGE_DEPTHSIZE
SG	shadowsize	チャート枠と要素が描画されるシャドウのサイズを指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_NONE_SHADOWSIZE OG_SMALL_SHADOWSIZE OG_MEDIUM_SHADOWSIZE OG_LARGE_SHADOWSIZE OG_XLARGE_SHADOWSIZE
SG	shadowdir	チャート枠と要素が描画されるシャドウの方向を指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_UPPERRIGHT_SHADOWDIR OG_UPPERLEFT_SHADOWDIR OG_LOWERRIGHT_SHADOWDIR OG_LOWERLEFT_SHADOWDIR
SG	plotframe	チャートの周囲に四角形を表示するかどうかを指定します。（円グラフ・チャートには適用されません。）
SG	legend	チャートの凡例を表示するかどうかを指定します。（表チャートには適用されません。）
SG	legendcolct	凡例内にラベルを表示するために使用する列数です。

円グラフ枠結合属性レコード

```
TYPE og_pieframe_ca IS RECORD
(ca_frame  og_frame_attr,
```

円グラフ枠属性レコード

```
TYPE og_pieframe_attr IS RECORD
(mask          NUMBER(3,0),
usage         NUMBER(1,0),
usagevalue    NUMBER(6),
plotorder     NUMBER(1,0),
Mask Constants:
OG_USAGE_PIEFRAMEA
OG_USAGE_PIEFRAMEA
OG_PLOTORDER_PIEFRAMEA
```

```

catego          BOOLEAN,
catnumfmt        VARCHAR2 (255),
catdatefmt       VARCHAR2 (255),
valuefmt         VARCHAR2 (255),
pctfmt           VARCHAR2 (255)
);

OG_CATEGS_PIEFRAMEA
OG_DATAVALS_PIEFRAMEA
OG_PCTVALUES_PIEFRAMEA
OG_TICKS_PIEFRAMEA
OG_OTHER_PIEFRAMEA
OG_NOOVERLAP_PIEFRAMEA
OG_CATNUMFMT_PIEFRAMEA
OG_CATDATEFMT_AXSFRAMEA
OG_VALUEFMT_PIEFRAMEA
OG_PCTFMT_PIEFRAMEA

OG_ALL_PIEFRAMEA
OG_NONE_PIEFRAMEA
```

	属性	説明
SG	usage	円グラフの各円弧と完成したチャートとの関係を指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_TOTALVALUE_USAGE OG_PCT_USAGE
SG	usagevalue	円グラフの各円弧は、そのデータ値におけるここで指定した値に対するパーセンテージとして表示されます。（usageがOG_TOTALVALUE_USAGEに設定されている場合に限り有効です。）
SG	plotorder	データ値が表示される方向を指定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_CCW_PLOTORDER OG_CW_PLOTORDER
SG	catego	円グラフの各円弧を、示されている項目の名前でラベル表示するかどうかを指定します。
SG	catnumfmt	円グラフの各円弧を、そのデータ値でラベル表示するかどうかを指定します。
SG	pctvalues	円グラフの各円弧を、示されている完成したチャートの割合でラベル表示するかどうかを指定します。
SG	ticks	円グラフの各円弧とそのラベルを結び目盛きざみを表示するかどうかを指定します。
SG	other	ここに入力される値より小さい割合の円弧をラベル「その他」で1つの円弧に結合することを指定します。
SG	nooverlap	円グラフの各円弧のラベルが互いにオーバーラップしないように指定します。
SG	catnumfmt	項目ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

SG	catdatefmt	項目ラベルの日付書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。
SG	valuefmt	データ値ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。
SG	pctfmt	パーセント値ラベルの数値書式を指定します。これは有効なSQLフォーマット文字列で指定する必要があります。詳細は、『Oracle8 Server SQLリファレンス』を参照してください。

表枠結合属性レコード

```
TYPE og_tableframe_ca IS RECORD
(ca_frame og_frame_attr,      /* generic frame */
 ca_table og_tableframe_attr /* table frame */
);
```

表枠属性レコード

```
TYPE og_tableframe_attr IS RECORD      Mask Constants::
(mask          NUMBER(3,0),
 automin       BOOLEAN,                OG_MIN_TABLEFRAMEA
 minrows       NUMBER(10,0),           OG_MIN_TABLEFRAMEA
 automax       BOOLEAN,                OG_MAX_TABLEFRAMEA
 maxrows       NUMBER(10,0),           OG_MAX_TABLEFRAMEA
 cname         BOOLEAN,                OG_CNAME_TABLEFRAMEA
 vgrid         BOOLEAN,                OG_VGRID_TABLEFRAMEA
 hgrid         BOOLEAN,                OG_HGRID_TABLEFRAMEA
 gridct        NUMBER(10,0)            OG_GRIDCT_TABLEFRAMEA
);

OG_ALL_TABLEFRAMEA
OG_NONE_TABLEFRAMEA
```

	属性	説明
SG	automin	チャートに表示される最小行数を「自動」に設定するかどうかを指定します。
SG	minrows	チャート上に表示する最小行数を指定します (autominがFALSEの場合)。
SG	automax	チャートに表示される最大行数を「自動」に設定するかどうかを指定します。
SG	maxrows	チャートに表示される最大行数を指定します (ただし、automaxがFALSEの場合のみ)。

SG	colnames	列の名前をチャート内の最初の行として表示するかどうかを指定します。
SG	vgrid	列と列の間に垂直グリッドを表示するかどうかを指定します。
SG	hgrid	行と行の間に水平グリッドを表示するかどうかを指定します。
SG	gridct	各水平グリッド間に表示されるデータの行数 (hgridがTRUEに設定されている場合)。

一般属性レコード

一般属性レコードは、あらゆる問合せに使用可能な属性で構成されます。

```
TYPE og_generic_attr IS RECORD      Mask Constants:
(mask                               NUMBER(6,0),
 name                               VARCHAR2(255),      OG_NAME_GENERICA
 parent                             og_object,           OG_PARENT_GENERICA
 ibbox                              og_rectangle,        OG_IBBOX_GENERICA
 obbox                              og_rectangle,        OG_OBBOX_GENERICA
 objtype                            NUMBER(2,0),         OG_OBJTYPE_GENERICA
 button                             og_buttonproc,      OG_BUTTON_GENERICA
 events                             NUMBER(2,0),         OG_EVENTS_GENERICA
 keycol                             VARCHAR2(255),      OG_KEYCOL_GENERICA
 execquery                          og_query,          OG_EXECQUERY_GENERICA
 setparam                           VARCHAR2(255),      OG_SETPARAM_GENERICA
 fmttrig                            VARCHAR2(255),      OG_FMTTRIG_GENERICA
 hide                               BOOLEAN              OG_HIDE_GENERICA
);

OG_ALL_GENERICA
OG_NONE_GENERICA
```

	属性	説明
CSG	name	オブジェクト名。
CG	parent	オブジェクトの親オブジェクトのハンドル。
G	ibbox	オブジェクトの内側のバウンディング・ボックス。これは、枠の太さやその他の属性の設定に関係なく、オブジェクトの理想の形状を構成している (オブジェクトの4つの制御点を結んでいる) 四角形です。
G	obbox	オブジェクトの外側のバウンディング・ボックス。これは、オブジェクトを完全に囲める最小サイズの四角形です。外側のバウンディング・ボックスは、オブジェクトの枠が太い場合には内側のバウンディング・ボックスと異なる場合があります。内側のバウンディング・ボックスではオブジェクトの本来あるべき形をたどっているのに対し、外側のバウンディング・ボックスではオブジェクト全体を囲みます。

G	objtype	<p>オブジェクトのタイプ。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_ARC_OBJTYPE オブジェクトが円弧である。</p> <p>OG_CHART_OBJTYPE オブジェクトがチャートである。</p> <p>OG_GROUP_OBJTYPE オブジェクトがグループである。</p> <p>OG_IMAGE_OBJTYPE オブジェクトがイメージである。</p> <p>OG_LINE_OBJTYPE オブジェクトが線である。</p> <p>OG_POLY_OBJTYPE オブジェクトが多角形または折れ線である。</p> <p>OG_RECT_OBJTYPE オブジェクトが四角形である。</p> <p>OG_RRECT_OBJTYPE オブジェクトが角の丸い四角形である。</p> <p>OG_SYMBOL_OBJTYPE オブジェクトが記号である。</p> <p>OG_TEXT_OBJTYPE オブジェクトがテキスト・オブジェクトである。</p>
CSG	button	<p>オブジェクトと対応付けるボタン・プロシージャのハンドル。要求したマウス・イベントをこのプロシージャで確実に受信できるようにするには、必ずイベント属性を正しく設定してください。</p>
CSG	events	<p>ボタン属性で指定されたプロシージャで受信するマウス・イベントのタイプ。この属性の値として、次のビルトイン定数のいずれかを指定できます。プロシージャで複数のイベント・タイプを確実に受信できるようにするには、この属性を、要求したイベントの定数の合計となるように設定します。</p> <p>OG_NO_EVENTS どのマウス・イベントも受信しない。</p> <p>OG_MOUSE_DOWN マウス・ボタンを押すイベントのみ受信する。</p> <p>OG_MOUSE_UP マウス・ボタンを放すイベントのみ受信する。</p> <p>OG_MOUSE_MOVE_UP マウス・ボタンを放したまま、マウスを移動させるイベントのみ受信する。</p>
CSG	keycol	<p>ドリルダウン・チャートに設定する列。この属性は、チャート要素にしか適用されません。</p>
CSG	execquery	<p>オブジェクトを選択したときに実行される問合せを指定します。</p>
CSG	setparam	<p>オブジェクトを選択すると値が設定されるパラメータです。</p>
CSG	fmttrig	<p>フォーマット・トリガー。この属性は、チャート要素にしか適用されません。</p>
SG	hide	<p>指定されたGraphics Builderオブジェクトを隠します。</p>

図形結合属性レコード

図形結合属性レコードは、一般属性レコードと図形属性レコードで構成されます。

```
TYPE og_graphic_ca IS RECORD
(graphic_caob  og_generic_attr,
```

図形属性レコード

図形属性レコードは、図形オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_graphic_attr IS RECORD      Mask Constants:
(mask                               NUMBER(4,0),
ewidth                             NUMBER(10,0),      OG_EWIDTH_GRAPHICA
rotang                             NUMBER(5,2),        OG_ROTANG_GRAPHICA
fecolor                             VARCHAR2(255),     OG_FECOLOR_GRAPHICA
becolor                             VARCHAR2(255),     OG_BECOLOR_GRAPHICA
edgepatt                             VARCHAR2(255),   OG_EDGEPAATT_GRAPHICA
ffcolor                             VARCHAR2(255),     OG_FFCOLOR_GRAPHICA
bfcolor                             VARCHAR2(255),     OG_BFCOLOR_GRAPHICA
fillpatt                             VARCHAR2(255),   OG_FILLPATT_GRAPHICA
dashstyle                           NUMBER(1,0),      OG_DASHSTYLE_GRAPHICA
capstyle                             NUMBER(2,0),      OG_CAPSTYLE_GRAPHICA
joinstyle                           NUMBER(2,0),      OG_JOINSTYLE_GRAPHICA
transfer                             NUMBER(1,0)       OG_TRANSFER_GRAPHICA
bevelstyle                           NUMBER(2,0)       OG_BEVELSTYLE_GRAPHICA
);

OG_ALL_GRAPHICA
OG_NONE_GRAPHICA
```

	属性	説明
CSG	ewidth	オブジェクトの枠の幅（レイアウト単位）。
CSG	rotang	オブジェクトの回転角度。オブジェクトが最初に作成されときの角度は0とみなされ、この属性は、オブジェクトが現在、開始角度から時計回りに回転している度数です。この属性を設定すると、オブジェクトを絶対角度に対して回転させることができ、またOG_ROTATEプロシーダを使用すると、オブジェクトを相対角度の度数だけ回転させることができます。（OG_ROTATEを使用してオブジェクトを回転させると、新しい絶対角度を反映するようにrotang属性が自動的に更新されます。）
CSG	fecolor	オブジェクトのフォアグラウンド線カラー。有効なカラー・パレットについて詳細は、「デフォルト・カラー・パレット」を参照してください。
CSG	becolor	オブジェクトのバックグラウンド線カラー。有効なカラー名について詳細は、「デフォルト・カラー・パレット」を参照してください。
CSG	edgepatt	オブジェクトの枠パターン。有効なパターン名について詳細は、「パターン・パレット」を参照してください。

CSG	ffcolor	オブジェクトのフォアグラウンド塗りカラー。有効なカラー名について詳細は、「デフォルト・カラー・パレット」を参照してください。
CSG	bfcolor	オブジェクトのバックグラウンド塗りカラー。有効なカラー名について詳細は、「デフォルト・カラー・パレット」を参照してください。
CSG	fillpatt	オブジェクトの塗りパターン。有効なパターン名について詳細は、「パターン・パレット」を参照してください。
CSG	dashstyle	<p>オブジェクトの枠の線種スタイル。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_SOLID_DSTYLE 実線を表示する。 OG_DOT_DSTYLE 点線を表示する。 OG_LONG_DSTYLE ダッシュ記号を連ねて、長い破線として表示する。 OG_DASHDOT_DSTYLE ダッシュ記号とダッシュ記号の間に点を含む破線を表示する。 OG_DOTDOT_DSTYLE 2つの連続した点からなる点線を表示する。 OG_SHORT_DSTYLE 短いダッシュ記号を連ねて、破線として表示する。 OG_DASHDOTDOT_DSTYLE ダッシュ記号とダッシュ記号の間に2つの連続した点を含む破線を表示する。</p>
CSG	capstyle	<p>オブジェクトの枠の端形式。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_BUTT_CSTYLE 切り端形式で表示する。 OG_PROJECTING_CSTYLE 伸ばし端形式で表示する。 OG_ROUND_CSTYLE 丸め端形式で表示する。</p>
CSG	joinstyle	<p>オブジェクトの枠の結合形式。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_MITRE_JSTYLE 角結合形式で表示する。 OG_BEVEL_JSTYLE 斜め結合形式で表示する。 OG_ROUND_JSTYLE 丸め結合形式で表示する。</p>
CSG	transfer	<p>オブジェクトの転送モード。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_COPY_TRANSFER 転送モードをコピーにする。 OG_REVCOPY_TRANSFER 転送モードを逆転コピーにする。 OG_OR_TRANSFER 転送モードをバック抜きにする。 OG_REVOR_TRANSFER 転送モードを逆転バック抜きにする。 OG_CLEAR_TRANSFER 転送モードをフォア抜きにする。 OG_REVCLEAR_TRANSFER 転送モードを逆転フォア抜きにする。 OG_INVERT_TRANSFER 転送モードをモノコピー（透明）にする。 OG_BACKINVERT_TRANSFER 転送モードを逆転モノコピー（透明）にする。</p>

CSG	bevelstyle	オブジェクトの凹凸スタイル。この属性の値として、次のビルトイン定数のいずれかを指定できます。	
		OG_INSET_BSTYLE	凸枠で表示する。
		OG_LOWERED_BSTYLE	凹で表示する。
		OG_OUTSET_BSTYLE	凹枠で表示する。
		OG_PLAIN_BSTYLE	オブジェクトに凹凸スタイルを設定しない。
		OG_RAISED_BSTYLE	凸で表示する。

グループ結合属性レコード

グループ結合属性レコードは、一般属性レコードとグループ属性レコードで構成されます。

```
TYPE og_group_ca IS RECORD
(group_caob og_generic_attr, /* generic */
 group_caog og_group_attr /* group */
);
```

グループ属性レコード

グループ属性レコードは、グループ・オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_group_attr IS RECORD
(mask          NUMBER(1,0),
 childcount    NUMBER(10,0),
 clipflag      BOOLEAN
);
```

Mask Constants:
OG_CHILDCOUNT_GROUPA
OG_CLIPFLAG_GROUPA

OG_ALL_GROUPA
OG_NONE_GROUPA

	属性	説明
G	childcount	グループ・オブジェクトに含まれる子の数。別のグループ・オブジェクトが、対象としているグループの子である場合、そのオブジェクトは単に1つのオブジェクトとしてカウントされます。
CSG	clipflag	グループ内の最初のオブジェクトの表示領域を、グループを表示する枠にするかどうかを指定します。TRUEに設定すると、レイアウトに表示されるオブジェクトは、枠内に含まれるグループ・オブジェクト、またはグループ・オブジェクトの一部となります。枠になっているオブジェクト自体も表示されます。この属性の値として、次のいずれかを指定できます。 TRUE グループ内の最初のオブジェクトがクリップする枠と見なされる。 FALSE グループ内の最初のオブジェクトはクリップする枠と見なされない。

イメージ結合属性レコード

イメージ結合属性レコードは、一般属性レコードとイメージ属性レコードで構成されます。

```
TYPE og_image_ca IS RECORD
(image_caob  og_generic_attr, /* generic */
 image_caoi  og_image_attr
```

イメージ属性レコード

イメージ属性レコードは、イメージ・オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_image_attr IS RECORD
(mask          NUMBER(3,0),
 cliprect      og_rectangle,
 upperleft     og_point,
 width         NUMBER(10,0),
 height        NUMBER(10,0),
 query         og_query,
 which_data    NUMBER(1,0),
 colname       VARCHAR2(255),
 quality       NUMBER(5,0),
 dither        BOOLEAN
);

Mask Constants:
OG_CLIPRECT_IMAGEA
OG_UPPERLEFT_IMAGEA
OG_SIZE_IMAGEA
OG_SIZE_IMAGEA
OG_DATA_IMAGEA
OG_DATA_IMAGEA
OG_DATA_IMAGEA
OG_QUALITY_IMAGEA
OG_DITHER_IMAGEA
OG_ALL_IMAGEA
OG_NONE_IMAGEA
```

	属性	説明
SG	cliprect	イメージを表示する枠のx座標とy座標、および高さ、幅（レイアウト単位）。この枠の中に含まれるイメージのみが表示されます。この属性を指定しないと、クリップする枠がイメージの全寸法と同じ大きさになります。
SG	upperleft	イメージの左上角のx座標とy座標（レイアウト単位）。
SG	width	イメージの幅（レイアウト単位）。この属性にイメージのデフォルトの幅以外の値を設定すると、イメージは新しい幅に合うように調整サイズされます。
SG	height	イメージの高さ（レイアウト単位）。この属性にイメージのデフォルトの高さ以外の値を設定すると、イメージは新しい高さに合うようにサイズ調整されます。
C	query	データベース内の表からイメージを取り出す問合せのハンドル。この表は、必ずユーザー表であり、Graphics Builderが、図表または描画、チャート・テンプレート、カラー・パレット、イメージ、サウンドをデータベースに保存またはエクスポートするときに使用されるプライベート表であってはいけません。データベースに格納できるのは、Oracleイメージ・フォーマットのみです。

C	which_data	<p>作成するイメージを問合せの新しいデータ・セットに入れるか古いデータ・セットに入れるかを指定します。 Graphics Builderには、この属性の値として使用可能なビルトイン数値定数が2つあります。</p> <p>OG_NEWDATA イメージを問合せの新しいデータ・セットに入れる。</p> <p>OG_OLDDATA イメージを問合せの古いデータ・セットに入れる。</p>
C	colname	<p>イメージ・データを入れる問合せ列の名前。作成されたイメージは、この属性で指定した列と問合せのカーソルが指し示す行が交わる問合せセルに入れられます。</p>
CSG	quality	<p>イメージを描画するときの品質を指定します。イメージの品質を高くすると、見栄えはよくなりますが、操作（たとえば、描画、移動、スケーリングなど）の処理時間が長くなります。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_HIGH_IQUALITY 品質が高くなる。</p> <p>OG_MED_IQUALITY 品質が標準になる。</p> <p>OG_LOW_IQUALITY 品質が低くなる。</p>
CSG	dither	<p>Graphics Builderでイメージを表示するときに、そのイメージをディザリングするかどうかを指定します。この属性の値として、次のいずれかを指定できます。</p> <p>TRUE イメージをディザリングする。</p> <p>FALSE イメージをディザリングしない。</p>

線結合属性レコード

線結合属性レコードは、一般属性レコードおよび図形属性レコード、線属性レコードで構成されます。

```
TYPE og_line_ca IS RECORD
(line_caob og_generic_attr,      /* generic */
 line_caoh og_graphic_attr,      /* graphic */
 line_caol og_line_attr /* line */
);
```

線属性レコード

線属性レコードは、線オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_line_attr IS RECORD
(mask      NUMBER(1,0),
 startpt   og_point,
Mask Constants:
OG_STARTPT_LINEA
```

```
endpt      og_point,          OG_ENDPT_LINEA
arrowstyle  NUMBER(1,0)       OG_ARROWSTYLE_LINEA
);

OG_ALL_LINEA
OG_NONE_LINEA
```

	属性	説明
CSG	startpt	線の始点のx座標とy座標（レイアウト単位）。
CSG	endpt	線の終点のx座標とy座標（レイアウト単位）。
CSG	arrowstyle	線の矢印のスタイル。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_NOARROW_ASTYLE 線に矢印を付けない。 OG_START_ASTYLE 線の始点に矢印を付ける。 OG_END_ASTYLE 線の終点に矢印を付ける。 OG_BOTH_ASTYLE 線の両側の終点に矢印を付ける。 OG_MIDTOSTART_ASTYLE 線の中央に始点方向を指す矢印を付ける。 OG_MIDTOEND_ASTYLE 線の中央に終点方向を指す矢印を付ける。

多角形結合属性レコード

多角形結合属性レコードは、一般属性レコードおよび図形属性レコード、多角形属性レコードで構成されます。

```
TYPE og_poly_ca IS RECORD
(poly_caob      og_generic_attr,
```

多角形属性レコード

多角形属性レコードは、多角形オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_poly_attr IS RECORD Mask Constants:
(mask      NUMBER(1,0) ,
pointct    NUMBER(10,0) ,      OG_POINTCT_POLYA
closed     BOOLEAN             OG_CLOSED_POLYA
);

OG_ALL_POLYA
OG_NONE_POLYA
```

	属性	説明
G	pointct	多角形オブジェクトを構成する点の数。

CSG	closed	多角形の閉じ形状。この属性の値として、次のいずれかを指定できます。 TRUE 多角形を閉じる。 FALSE 多角形を開く。
-----	--------	---

プリンタ属性レコード

```

TYPE og_printer_attr IS RECORD
(mask          NUMBER(3,0),
 name          VARCHAR2(255),
 landscape     BOOLEAN,
 startpage     NUMBER(5,0),
 endpage       NUMBER(5,0),
 width         NUMBER(10,0),
 height        NUMBER(10,0),
 copies        NUMBER(5,0),
 printfile     VARCHAR2(255)
);

Mask Constants:
OG_NAME_PRINTERA
OG_LANDSCAPE_PRINTERA
OG_STARTPAGE_PRINTERA
OG_ENDPAGE_PRINTERA
OG_WIDTH_PRINTERA
OG_HEIGHT_PRINTERA
OG_COPIES_PRINTERA
OG_PRINTFILE_PRINTERA
OG_ALL_PRINTERA
OG_NONE_PRINTERA

```

	属性	説明
SG	name	現在のプリンタの名前。
SG	landscape	図表をLandscapeモードとPortraitモードのどちらで印刷するかを指定します
SG	startpage	印刷する第1ページ。
SG	endpage	印刷する最終ページ。
S	width	ページの幅。
S	height	ページの高さ。
SG	copies	印刷部数。
SG	printfile	出力先のPostScriptファイルの名前。このプロパティをNULLにすると、出力がプリンタへ送られます。

問合せ属性レコード

問合せ属性レコードは、問合せにのみ使用可能な属性で構成されます。

```

TYPE og_query_attr IS RECORD
(mask          NUMBER(4,0),
 name          VARCHAR2(255),
 dateformat    VARCHAR2(255),
 querysource   VARCHAR2(2000),
 querytype     NUMBER(1,0),
 cachetype     NUMBER(1,0),
);

Mask Constants:
OG_NAME_QUERYA
OG_DATEFORMAT_QUERYA
OG_QUERYSOURCE_QUERYA
OG_QUERYTYPE_QUERYA
OG_CACHETYPE_QUERYA

```

```
maxflag      BOOLEAN,          OG_MAXFLAG_QUERYA
maxrows      NUMBER(10,0),     OG_MAXROWS_QUERYA
execopen     BOOLEAN,         OG_EXECOPEN_QUERYA
exectimer    VARCHAR2(255),    OG_EXECTIMER_QUERYA
execalert    VARCHAR2(255),    OG_EXECALERT_QUERYA
customproc   VARCHAR2(255),    OG_CUSTOMPROC_QUERYA
postproc     VARCHAR2(255)     OG_POSTPROC_QUERYA
);

OG_ALL_QUERYA
OG_NONE_QUERYA
```

	属性	説明
CSG	name	問合せの名前です。
CSG	dateformat	問合せの日付書式マスク。
CSG	querysource	問合せデータのソース。データのソースがデータベースであれば、この属性に問合せのSQL SELECT文のテキストを入れる必要があります。データがファイル・システムに格納されている場合は、この属性にデータ・ファイルのパスと名前を入れる必要があります。
CSG	querytype	問合せのタイプです。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_CUSTOM_QTYPE 通常の問合せを使用する。 OG_EXSQL_QTYPE SQL SELECT文で構成されるテキスト・ファイルからデータをフェッチする。 OG_PRN_QTYPE PRNファイルを使用する。 OG_SQL_QTYPE SQL SELECT文を使用する。 OG_SYLK_QTYPE SYLKファイルを使用する。 OG_WKS_QTYPE WKSファイルを使用する。
CSG	cachetype	問合せの実行により新しく取り出されたデータを処理する方法を決定します。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_APPEND_CACHETYPE 既存のデータ行がすべて保持され、新しいデータ行が既存のデータ・セットの一番下に追加される。 OG_COPY_CACHETYPE 前に実行されたデータがすべてバッファにコピーされ、新しく取り出されたデータに置換される。 OG_NONE_CACHETYPE 前に実行されたデータがすべて廃棄され、新しく取り出されたデータに置換される。
CSG	maxflag	データ・セット内の行数に制限を設定するかどうかを指定します。
CSG	maxrows	問合せデータ・セットに保持されるデータの最大行数を指定します。
CSG	execopen	図表を実行中にオープンした時に問合せを自動的に実行するかどうかを指定します。
CSG	exectimer	問合せを実行するときに使用するタイマーの名前。

CSG	execalert	未使用。
CSG	customproc	カスタムの問合せが実行されるときに起動するPL/SQLプロシージャです。
CSG	postproc	問合せが実行された後で起動するPL/SQLプロシージャです。

四角形結合属性レコード

四角形結合属性レコードは、一般属性レコード、図形属性レコードおよび四角形属性レコードで構成されます。

```
TYPE og_rect_ca IS RECORD
(rect_caob og_generic_attr, /* generic */
 rect_caoh og_graphic_attr, /* graphic */
 rect_caor og_rect_attr      /* rectangle */
);
```

四角形属性レコード

四角形属性レコードは、四角形オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_rect_attr IS RECORD Mask Constants:
(mask          NUMBER(1,0),
 baserect      og_rectangle      OG_BASERECT_RECTA
);

                                OG_ALL_RECTA
                                OG_NONE_RECTA
```

	属性	説明
CSG	baserect	四角形オブジェクトの基礎として使用する四角形の左上角のx座標とy座標。

参照線属性レコード

```
TYPE og_refline_attr IS RECORD      Mask Constants:
(mask          NUMBER(2,0),
 numvalue      NUMBER(6),           OG_VALUE_REFLINEA
 datevalue     DATE,                OG_VALUE_REFLINEA
 label         VARCHAR2(255),       OG_LABEL_REFLINEA
 axis          NUMBER(1,0)          OG_AXIS_REFLINEA
);

                                OG_ALL_REFLINEA
                                OG_NONE_REFLINEA
```

	属性	説明
SG	numvalue	参照線の数値。

SG	datevalue	参照線の日付値。
SG	label	凡例に表示する参照線を識別するテキスト・ラベル。
SG	axis	参照値の基準となる軸を指定します。

角の丸い四角形結合属性レコード

角の丸い四角形結合属性レコードは、一般属性レコード、図形属性レコードおよび丸い四角形属性レコードで構成されます。

```
TYPE og_rrect_ca IS RECORD
(rrect_caob          og_generic_attr,          /* generic */
 rrect_caohog_graphic_attr,          /* graphic */
 rrect_caorog_rrect_attr /* rounded rectangle */
);
```

角の丸い四角形属性レコード

角の丸い四角形属性レコードは、角の丸い四角形オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_rrect_attr IS RECORD          Mask Constants:
(mask          NUMBER(1,0),
 baserect      og_rectangle,          OG_BASERECT_RRECTA
 corner        og_point              OG_CORNER_RRECTA
);
                                     OG_ALL_RRECTA
                                     OG_NONE_RRECTA
```

	属性	説明
CSG	baserect	四角形オブジェクトの基礎として使用する四角形の左上角のx座標とy座標。
CSG	corner	丸める角を形成する弧を360度になるまで延長してできる楕円のx半径とy半径（レイアウト単位）。

サウンド属性レコード

サウンド属性レコードは、サウンドにのみ使用可能な属性で構成されます。

```
TYPE og_sound_attr IS RECORD          Mask Constants:
(mask          NUMBER(1,0),
 query         og_query,              OG_DATA_SOUNDA
 which_data    NUMBER(1,0),          OG_DATA_SOUNDA
 colname       VARCHAR2(255),        OG_DATA_SOUNDA
```

```
name          VARCHAR2 (255) ,          OG_NAME_SOUNDA
);

OG_ALL_SOUNDA
OG_NONE_SOUNDA
```

	属性	説明
C	query	データベースの表からサウンドを検索する問合せのハンドル。この表は、必ずユーザー表であり、Graphics Builderが、図表または描画、チャート・テンプレート、カラー・パレット、イメージ、サウンドをデータベースに保存またはエクスポートするときに使用されるプライベート表であってはいけません。
C	which_data	作成されるサウンドが、問合せの新規データ・セット内に含まれているか、旧データ・セットに含まれているかを指定します。Graphics Builderには、この属性の値として使用可能なビルトイン数値定数が2つあります。 OG_NEWDATA サウンドを問合せの新しいデータ・セットに入れる。 OG_OLDDATA サウンドを問合せの古いデータ・セットに入れる。
C	colname	サウンド・データが入る問合せ列の名前。作成されるサウンドは、この属性により指定される列と問合せのカーソルが指している行の交差位置の問合せセルに含まれているものです。
CSG	name	サウンドの名前。

記号結合属性レコード

記号結合属性レコードは、一般属性レコードおよび図形属性レコード、記号属性レコードで構成されます。

```
TYPE og_symbol_ca IS RECORD
(symbol_caob          og_generic_attr,          /* generic */
 symbol_caoh          og_graphic_attr,          /* graphic */
 symbol_caos          og_symbol_attr /* symbol */
);
```

記号属性レコード

記号属性レコードは、記号オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_symbol_attr IS RECORD      Mask Constants:
(mask          NUMBER (1,0) ,
 center       og_point,           OG_CENTER_SYMBOLA
 indx        NUMBER (3,0) ,       OG_INDX_SYMBOLA
 symsize     NUMBER (1,0)         OG_SYMSIZE_SYMBOLA
);

OG_ALL_SYMBOLA
OG_NONE_SYMBOLA
```

	属性	説明
CSG	center	記号の中央のx座標とy座標（レイアウト単位）。
CSG	indx	デザイン内の記号パレットに表示される記号の索引(または番号)。
CSG	symsize	記号のサイズ。この属性の値として、次のビルトイン定数のいずれかを指定できます。 OG_LARGE_SYMSIZE サイズが大きくなる。 OG_MEDIUM_SYMSIZE サイズが標準になる。 OG_SMALL_SYMSIZE サイズが小さくなる。

テキスト属性の概要

テキスト属性レコードには、テキスト・オブジェクト内に表示されるテキストは含まれません。そのかわりに、まずテキスト・オブジェクトを作成してから、OG_INSERT_CMPTEXTプロシージャを使用してそのテキスト・オブジェクトに複数テキスト要素を挿入する必要があります。1つのテキスト・オブジェクトに複数の複数テキスト要素を挿入することができ、各要素はそのオブジェクト内の1行のテキストを表します。また、それぞれの複数テキスト要素に、1つ以上の単一テキスト要素を挿入することもできます。単一テキスト要素は、実際のテキスト文字列で構成され、複数テキスト要素に挿入する場合にはOG_INSERT_SMPTEXTプロシージャを使用して実行する必要があります。

複数テキストおよび単一テキストの属性レコードについては、後のリストを参照してください。

テキスト結合属性レコード

テキスト結合属性レコードは、一般属性レコード、図形属性レコードおよびテキスト属性レコードで構成されます。

```
TYPE og_text_ca IS RECORD
(text_caob          og_generic_attr, /* generic */
 text_caoh          og_graphic_attr, /* graphic */
 text_caot          og_text_attr    /* text */
);
```

テキスト属性レコード

テキスト属性レコードは、テキスト・オブジェクトにのみ使用可能な属性で構成されます。

```
TYPE og_text_attr IS RECORD
(mask          NUMBER(6,0),
 origin       og_point,
Mask Constants:
OG_ORIGIN_TEXTA
```



```
ctcount      NUMBER(10,0),      OG_CTCOUNT_TEXTA
gfont        og_font_attr,      OG_GFONT_TEXTA
gcolor       VARCHAR2(255),     OG_GCOLOR_TEXTA
spacing      NUMBER(1,0),       OG_SPACING_TEXTA
custom       NUMBER(10,0),      OG_SPACING_TEXTA
horigin      NUMBER(1,0),       OG_HORIGIN_TEXTA
vorigin      NUMBER(1,0),       OG_VORIGIN_TEXTA
halign       NUMBER(2,0),       OG_HALIGN_TEXTA
valign       NUMBER(3,0),       OG_VALIGN_TEXTA
fixed        BOOLEAN,          OG_FIXED_TEXTA
wrap         BOOLEAN,          OG_WRAP_TEXTA
bbscale      BOOLEAN,          OG_BBSCALE_TEXTA
fontscale    BOOLEAN,          OG_FONTSCALE_TEXTA
invisible    BOOLEAN,          OG_INVISIBLE_TEXTA
width        NUMBER(10,0),      OG_FIXEDWH_TEXTA
height       NUMBER(10,0),      OG_FIXEDWH_TEXTA
);

OG_ALL_TEXTA
OG_NONE_TEXTA
```

	属性	説明
CSG	origin	テキスト・オブジェクトの左上角のx座標とy座標です(レイアウト単位)。
G	Ctcount	テキスト・オブジェクトを構成する複数テキスト要素の数。
S	Gfont	テキスト・オブジェクトのグローバル・フォント。この属性を設定すると、テキスト・オブジェクトのすべての単一テキスト要素のフォント属性がこのフォントに設定されます。この属性の設定によって影響されるのは、既存の単一テキスト要素のみです。新しく追加された単一テキスト要素は、その単一テキスト属性レコードに指定されたフォントで表示されます。
S	gcolor	テキスト・オブジェクトのグローバル・カラー。この属性を設定すると、テキスト・オブジェクトのすべての単一テキスト要素のカラー属性がこのカラーに設定されます。この属性の設定によって影響されるのは、既存の単一テキスト要素のみです。新しく追加された単一テキスト要素は、その単一テキスト属性レコードに指定されたカラーで表示されます。
CSG	spacing	テキスト・オブジェクトの行間隔。この属性の値として、次のビルトイン定数のいずれかを指定できます。カスタム間隔を設定する場合、custom属性の値には、正確な間隔の値を指定してください。 OG_SINGLE_SPACE テキストの行間隔をシングル・スペースに設定する。 OG_ONEHALF_SPACE テキストの行間隔を1-1/2スペースに設定する。 OG_DOUBLE_SPACE テキストの行間隔をダブル・スペースに設定する。 OG_CUSTOM_SPACE テキストの行間隔をカスタム・サイズに設定する。使用する実際の間隔は、custom属性に定義します。

CSG	custom	テキスト・オブジェクトのカスタム間隔（レイアウト単位）。この属性を使用して間隔を指定できるのは、spacing属性がカスタム間隔に設定されている場合のみです。
CSG	horigin	<p>原点に対するテキスト・オブジェクトの水平位置。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_LEFT_HORIGIN 原点が左枠に沿って配置される。 OG_CENTER_HORIGIN 原点が左枠と右枠のちょうど中間に配置される。 OG_RIGHT_HORIGIN 原点が右枠に沿って配置される。</p>
CSG	vorigin	<p>原点に対するテキスト・オブジェクトの垂直位置。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_TOP_VORIGIN 原点が上枠に沿って配置される。 OG_MIDDLE_VORIGIN 原点が上枠と下枠のちょうど中間に配置される。 OG_BOTTOM_VORIGIN 原点が下枠に沿って配置される。</p>
CSG	halign	<p>テキスト・オブジェクトの水平文字揃え。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_LEFT_HALIGN テキストを左揃えで表示する。 OG_CENTER_HALIGN テキストを中央揃えで表示する。 OG_RIGHT_HALIGN テキストを右揃えで表示する。</p>
CSG	valign	<p>テキスト・オブジェクトの垂直文字揃え。この属性の値として、次のビルトイン定数のいずれかを指定できます。</p> <p>OG_TOP_VALIGN テキストを上揃えで表示する。 OG_MIDDLE_VALIGN テキストを中央揃えで表示する。 OG_BOTTOM_VALIGN テキストを下揃えで表示する。</p>
CSG	wrap	<p>テキストを、テキスト・オブジェクトのバウンディング・ボックスに収まるように「ラップ（改行）」するかどうかを指定します。複数テキスト要素はテキストの行を表し、単一テキスト要素から構成されます。この属性の値として、次のいずれかを指定できます。</p> <p>TRUE テキストをラップする。 FALSE テキストをラップしない。</p>
CSG	bbscale	<p>テキスト・オブジェクトをスケーリングするときにテキスト・オブジェクトの境界線もスケーリングするかどうかを指定します。この属性の値として、次のいずれかを指定できます。</p> <p>TRUE バウンディング・ボックスをサイズ調整する。 FALSE バウンディング・ボックスをサイズ調整しない。</p>
CSG	fontscale	<p>テキスト・オブジェクトをスケーリングするときにフォントのサイズもスケーリングするかどうかを指定します。この属性の値として、次のいずれかを指定できます。</p> <p>TRUE ポイント・サイズをサイズ調整する。 FALSE ポイント・サイズをサイズ調整しない。</p>

CSG	fixed	<p>テキスト・オブジェクトのバウンディング・ボックスを固定サイズのままにするかどうかを指定します。この属性をTRUEにする場合は、<i>width</i>属性および<i>height</i>属性の値に、バウンディング・ボックスのサイズを指定する必要があります。この属性の値として、次のいずれかを指定できます。</p> <p>TRUE バウンディング・ボックスのサイズを固定する。バウンディング・ボックスのサイズは、<i>width</i> 属性および <i>height</i> 属性に定義します。</p> <p>FALSE バウンディング・ボックスのサイズを固定しない。</p>
CSG	width	<p>バウンディング・ボックスの幅（レイアウト単位）。バウンディング・ボックスが変化するたびに、この属性は新しい幅になるように自動的に更新されます。この属性を使用して幅を設定できるのは、<i>fixed</i>属性がTRUEの場合のみです。</p>
CSG	height	<p>バウンディング・ボックスの高さ（レイアウト単位）。バウンディング・ボックスが変化するたびに、この属性は新しい高さになるように自動的に更新されます。この属性を使用して高さを設定できるのは、<i>fixed</i>属性がTRUEの場合のみです。</p>
CSG	invisible	<p>テキスト・オブジェクト内のテキストを表示しないようにするかどうかを指定します。これは、パスワードを見せたくない場合に、パスワードを入力するテキスト・フィールドに指定すると便利です。この属性の値として、次のいずれかを指定できます。</p> <p>TRUE テキストが非表示になる。</p> <p>FALSE テキストが表示される。</p>

フォント属性レコード

フォント属性レコードを使用すると、フォント・スタイル、ポイント・サイズなどのフォントのプロパティを指定できます。

```
TYPE og_font_attr IS RECORD
(mask          NUMBER(3,0),
 typeface      VARCHAR2(255),
 psize         NUMBER(10,2),
 style         NUMBER(5,0),
 weight       NUMBER(5,0),
 width        NUMBER(5,0),
 kerning      BOOLEAN,
 nearest      BOOLEAN,
 synthesize    BOOLEAN,
 charset      NUMBER(5,0)
);
```

Mask Constants:

```
OG_TYPEFACE_FONTA
OG_PSIZE_FONTA
OG_STYLE_FONTA
OG_WEIGHT_FONTA
OG_WIDTH_FONTA
OG_KERNING_FONTA
OG_NEAREST_FONTA
OG_SYNTHESIZE_FONTA
OG_CHARSET_FONTA

OG_ALL_FONTA
OG_NONE_FONTA
```

	属性	説明																				
CG	typeface	フォントのスタイル。この値はシステムによって異なります。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。																				
CG	ptsize	フォントのサイズ。このフィールドの値はシステムによって異なります。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。																				
CG	style	<p>フォントのスタイル。すべてのスタイルをすべてのシステムで使用するわけではありません。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。このフィールドには、次のビルトイン定数を指定できます。</p> <table><tr><td>OG_BLINK_FONTSTYLE</td><td>隠し文字にする。</td></tr><tr><td>OG_INVERTED_FONTSTYLE</td><td>反転表示をする。</td></tr><tr><td>OG_ITALIC_FONTSTYLE</td><td>イタリック表示をする。</td></tr><tr><td>OG_OBLIQUE_FONTSTYLE</td><td>斜体表示をする。</td></tr><tr><td>OG_OUTLINE_FONTSTYLE</td><td>アウトライン表示をする。</td></tr><tr><td>OG_OVERSTRIKE_FONTSTYLE</td><td>取消し線を表示する</td></tr><tr><td>OG_PLAIN_FONTSTYLE</td><td>プレーン表示をする。</td></tr><tr><td>OG_SHADOW_FONTSTYLE</td><td>影付き表示をする。</td></tr><tr><td>OG_UNDERLINE_FONTSTYLE</td><td>下線表示をする。</td></tr><tr><td>OG_UNKNOWN_FONTSTYLE</td><td>不明。スタイルをこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってそのフォント・スタイルが判別されない場合は、この値が戻されます。</td></tr></table>	OG_BLINK_FONTSTYLE	隠し文字にする。	OG_INVERTED_FONTSTYLE	反転表示をする。	OG_ITALIC_FONTSTYLE	イタリック表示をする。	OG_OBLIQUE_FONTSTYLE	斜体表示をする。	OG_OUTLINE_FONTSTYLE	アウトライン表示をする。	OG_OVERSTRIKE_FONTSTYLE	取消し線を表示する	OG_PLAIN_FONTSTYLE	プレーン表示をする。	OG_SHADOW_FONTSTYLE	影付き表示をする。	OG_UNDERLINE_FONTSTYLE	下線表示をする。	OG_UNKNOWN_FONTSTYLE	不明。スタイルをこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってそのフォント・スタイルが判別されない場合は、この値が戻されます。
OG_BLINK_FONTSTYLE	隠し文字にする。																					
OG_INVERTED_FONTSTYLE	反転表示をする。																					
OG_ITALIC_FONTSTYLE	イタリック表示をする。																					
OG_OBLIQUE_FONTSTYLE	斜体表示をする。																					
OG_OUTLINE_FONTSTYLE	アウトライン表示をする。																					
OG_OVERSTRIKE_FONTSTYLE	取消し線を表示する																					
OG_PLAIN_FONTSTYLE	プレーン表示をする。																					
OG_SHADOW_FONTSTYLE	影付き表示をする。																					
OG_UNDERLINE_FONTSTYLE	下線表示をする。																					
OG_UNKNOWN_FONTSTYLE	不明。スタイルをこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってそのフォント・スタイルが判別されない場合は、この値が戻されます。																					

CG	weight	<p>フォントの太さ。すべての太さをすべてのシステムで利用できるわけではありません。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。このフィールドには、次のビルトイン定数を指定できます。</p> <table><tr><td>OG_BOLD_FONTWEIGHT</td><td>太字にする。</td></tr><tr><td>OG_DEMIBOLD_FONTWEIGHT</td><td>やや細い太字にする。</td></tr><tr><td>OG_DEMILIGHT_FONTWEIGHT</td><td>やや太い細字にする。</td></tr><tr><td>OG_EXTRABOLD_FONTWEIGHT</td><td>さらに太い太字にする。</td></tr><tr><td>OG_EXTRALIGHT_FONTWEIGHT</td><td>さらに細い細字にする。</td></tr><tr><td>OG_LIGHT_FONTWEIGHT</td><td>細字にする。</td></tr><tr><td>OG_MEDIUM_FONTWEIGHT</td><td>標準にする。</td></tr><tr><td>OG_ULTRABOLD_FONTWEIGHT</td><td>最も太い太字にする。</td></tr><tr><td>OG_ULTRALIGHT_FONTWEIGHT</td><td>最も細い細字にする。</td></tr><tr><td>OG_UNKNOWN_FONTWEIGHT</td><td>不明。フォントの太さをこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってその太さが判別されない場合は、この値が戻されます。</td></tr></table>	OG_BOLD_FONTWEIGHT	太字にする。	OG_DEMIBOLD_FONTWEIGHT	やや細い太字にする。	OG_DEMILIGHT_FONTWEIGHT	やや太い細字にする。	OG_EXTRABOLD_FONTWEIGHT	さらに太い太字にする。	OG_EXTRALIGHT_FONTWEIGHT	さらに細い細字にする。	OG_LIGHT_FONTWEIGHT	細字にする。	OG_MEDIUM_FONTWEIGHT	標準にする。	OG_ULTRABOLD_FONTWEIGHT	最も太い太字にする。	OG_ULTRALIGHT_FONTWEIGHT	最も細い細字にする。	OG_UNKNOWN_FONTWEIGHT	不明。フォントの太さをこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってその太さが判別されない場合は、この値が戻されます。
OG_BOLD_FONTWEIGHT	太字にする。																					
OG_DEMIBOLD_FONTWEIGHT	やや細い太字にする。																					
OG_DEMILIGHT_FONTWEIGHT	やや太い細字にする。																					
OG_EXTRABOLD_FONTWEIGHT	さらに太い太字にする。																					
OG_EXTRALIGHT_FONTWEIGHT	さらに細い細字にする。																					
OG_LIGHT_FONTWEIGHT	細字にする。																					
OG_MEDIUM_FONTWEIGHT	標準にする。																					
OG_ULTRABOLD_FONTWEIGHT	最も太い太字にする。																					
OG_ULTRALIGHT_FONTWEIGHT	最も細い細字にする。																					
OG_UNKNOWN_FONTWEIGHT	不明。フォントの太さをこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってその太さが判別されない場合は、この値が戻されます。																					
CG	width	<p>フォントの幅。すべての幅をすべてのシステムで利用できるわけではありません。詳細は、システム管理者に問い合わせるか、またはシステムのマニュアルを参照してください。このフィールドには、次のビルトイン定数を指定できます。</p> <table><tr><td>OG_DENSE_FONTWIDTH</td><td>狭くする。</td></tr><tr><td>OG_EXPAND_FONTWIDTH</td><td>広くする。</td></tr><tr><td>OG_EXTRADENSE_FONTWIDTH</td><td>さらに狭くする。</td></tr><tr><td>OG_EXTRAEXPAND_FONTWIDTH</td><td>さらに広くする。</td></tr><tr><td>OG_NORMAL_FONTWIDTH</td><td>標準にする。</td></tr><tr><td>OG_SEMIDENSE_FONTWIDTH</td><td>やや広い狭さにする。</td></tr><tr><td>OG_SEMIEXPAND_FONTWIDTH</td><td>やや狭い広さにする。</td></tr><tr><td>OG_ULTRADENSE_FONTWIDTH</td><td>最も狭くする。</td></tr><tr><td>OG_ULTRAEXPAND_FONTWIDTH</td><td>最も広くする。</td></tr><tr><td>OG_UNKNOWN_FONTWIDTH</td><td>不明。フォントの幅をこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってその幅が判別されない場合は、この値が戻されます。</td></tr></table>	OG_DENSE_FONTWIDTH	狭くする。	OG_EXPAND_FONTWIDTH	広くする。	OG_EXTRADENSE_FONTWIDTH	さらに狭くする。	OG_EXTRAEXPAND_FONTWIDTH	さらに広くする。	OG_NORMAL_FONTWIDTH	標準にする。	OG_SEMIDENSE_FONTWIDTH	やや広い狭さにする。	OG_SEMIEXPAND_FONTWIDTH	やや狭い広さにする。	OG_ULTRADENSE_FONTWIDTH	最も狭くする。	OG_ULTRAEXPAND_FONTWIDTH	最も広くする。	OG_UNKNOWN_FONTWIDTH	不明。フォントの幅をこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってその幅が判別されない場合は、この値が戻されます。
OG_DENSE_FONTWIDTH	狭くする。																					
OG_EXPAND_FONTWIDTH	広くする。																					
OG_EXTRADENSE_FONTWIDTH	さらに狭くする。																					
OG_EXTRAEXPAND_FONTWIDTH	さらに広くする。																					
OG_NORMAL_FONTWIDTH	標準にする。																					
OG_SEMIDENSE_FONTWIDTH	やや広い狭さにする。																					
OG_SEMIEXPAND_FONTWIDTH	やや狭い広さにする。																					
OG_ULTRADENSE_FONTWIDTH	最も狭くする。																					
OG_ULTRAEXPAND_FONTWIDTH	最も広くする。																					
OG_UNKNOWN_FONTWIDTH	不明。フォントの幅をこの値に設定できません。ただし、フォントを取得したときに、Graphics Builderによってその幅が判別されない場合は、この値が戻されます。																					

CG	Kerning	<p>フォントをカーニングするかどうかを指定する。カーニングとは、テキストを読みやすくするために、隣接する文字間の空白を調整することです。このフィールドの値として、次のいずれかを指定できます。</p> <p>TRUE フォントをカーニングする。 FALSE フォントをカーニングしない。</p>
C	nearest	<p>指定したフォントが見つからない場合に、Graphics Builderによってそれにもっとも近いフォントが代用されるようにするかどうかを指定する。近似フォントを見つけるときの優先順位は、書体、サイズ、スタイル、フォントの太さ、幅の順です（つまり、Graphics Builderによって、まず指定された書体が検索され、次にサイズが検索され、という順番で検索されていきます）。この属性の値として、次のいずれかを指定できます。</p> <p>TRUE 最も近いフォントをかわりに使用する。 FALSE 最も近いフォントをかわりに使用しない。</p>
C	synthesize	<p>指定したフォントが見つからない場合、Graphics Builderによって近似フォントを変形させて指定フォントを合成するかどうかを指定します。このフィールドの値として、次のいずれかを指定できます。</p> <p>TRUE フォントを合成する。 FALSE フォントを合成しない。</p>

CG	charset	<p>フォントのキャラクタ・セット。このフィールドの値には、U.S. ASCII および漢字、アラビア文字などのキャラクタ・セットを指定します。このフィールドの有効な値のリストについては、使用しているオペレーティング・システムのBuilderのマニュアルを参照してください。</p> <p> OG_US7ASCII_CHARSET OG_WE8DEC_CHARSET OG_WE8HP_CHARSET OG_US8PC437_CHARSET OG_WE8EBDIC37_CHARSET OG_WE8EBDIC500_CHARSET OG_WE8PC850_CHARSET OG_D7DEC_CHARSET OG_F7DEC_CHARSET OG_S7DEC_CHARSET OG_E7DEC_CHARSET OG_SF7ASCII_CHARSET OG_NDK7DEC_CHARSET OG_I7DEC_CHARSET OG_NL7DEC_CHARSET OG_CH7DEC_CHARSET OG_SF7DEC_CHARSET OG_WE8ISO8859P1_CHARSET OG_EE8ISO8859P2_CHARSET OG_SE8ISO8859P3_CHARSET OG_NEE8ISO8859P4_CHARSET OG_CL8ISO8859P5_CHARSET OG_AR8ISO8859P6_CHARSET OG_EL8ISO8859P7_CHARSET OG_IW8ISO8859P8_CHARSET OG_WE8ISO8859P9_CHARSET OG_AR8ASMO708PLUS_CHARSET OG_AR7ASMO449PLUS_CHARSET OG_WE8MACROMAN8_CHARSET OG_JVMS_CHARSET OG_JEUC_CHARSET OG_JDEC_CHARSET OG_SJIS_CHARSET OG_JDBCS_CHARSET OG_JHP_CHARSET OG_KSC5601_CHARSET </p>
----	---------	---

CG	charset (続き)	OG_KIBM5540_CHARSET OG_KDBCS_CHARSET OG_CGB231380_CHARSET OG_CDBCS_CHARSET OG_BIG5_CHARSET OG_CNS1164386_CHARSET
----	-------------------	---

複数テキスト要素属性レコード

複数テキスト要素属性レコードは、複数テキスト要素にのみ使用可能な属性で構成されます。

```
TYPE og_cmptext_attr IS RECORD
(mask          NUMBER(1,0),
 stcount       NUMBER(10,0)
);
Mask Constants:
OG_STCOUNT_CMPTEXTA
OG_ALL_CMPTEXTA
OG_NONE_CMPTEXTA
```

	属性	説明
G	stcount	複数テキスト要素を構成する単一テキスト要素の数。

単一テキスト要素属性レコード

単一テキスト要素属性レコードは、単一テキスト要素にのみ使用可能な属性で構成されます。

```
TYPE og_smptext_attr IS RECORD
(mask          NUMBER(1,0),
 str           VARCHAR2(2000)
 font         og_font_attr,
 color        VARCHAR2(255)
);
Mask Constants:
OG_STR_SMPTEXTA
OG_FONT_SMPTEXTA
OG_COLOR_SMPTEXTA
OG_ALL_SMPTEXTA
OG_NONE_SMPTEXTA
```

	属性	説明
CSG	str	単一テキスト要素の実際のテキストが含まれている文字列。
CSG	font	文字列のテキストを表示するフォント。使用されるフォント属性を、フォント属性レコードのマスク属性（複数の場合もあり）の値で指定します。マスクを設定していない属性レコードのフィールドには、影響はありません。
CSG	color	文字列のテキストを表示するカラー。これはテキスト自体のカラーであることに注意してください。テキスト・オブジェクトの線カラーや塗りカラーを設定する場合は、そのテキスト・オブジェクトの図形属性を変更してください。

例

このプロシージャでは、座標(1,1)に"Message"という名前のテキスト・オブジェクトが作成され、次の2行のテキストが12ポイントのTimesフォントで入っています。

This is line 1.

And now line 2.

各複数テキスト要素は、テキスト・オブジェクトのちょうど1行のテキストを表すことを覚えておいてください。

```
PROCEDURE make_text IS
    text_obj    og_object;
    text_rec    og_text_ca;
    smp_rec     og_smp_text_attr;
    font_rec    og_font_attr;
BEGIN
    /* Set text object's name and origin attributes */
    text_rec.text_caob.name:='Message';
    text_rec.text_caot.origin.x:=OG_INCH;
    text_rec.text_caot.origin.y:=OG_INCH;
    text_rec.text_caob.mask:=OG_NAME_GENERICA;
    text_rec.text_caoh.mask:=OG_NONE_GRAPHICA;
    text_rec.text_caot.mask:=OG_ORIGIN_TEXTA;
    /* Make the text object */
    text_obj:=og_make(text_rec);
    /* Insert new compound text element into the text object at
       index 0 */
    og_insert_cmptext(text_obj, 0);
    /* Set font record's typeface and point size attributes */
    font_rec.typeface:='times';
    font_rec.ptsize:=12;
    font_rec.mask:=OG_TYPEFACE_FONTA+
                   OG_PSIZE_FONTA;
    /* Set simple text record for text string and font */
    smp_rec.str:='This is line 1.';
    smp_rec.font:=font_rec;
    smp_rec.mask:=OG_STR_SMPTEXTA+
                  OG_FONT_SMPTEXTA;
    /* Insert a new simple text element at index 0 in text
       object's compound text element at index 0, using
       defined simple text record */
    og_insert_smp_text(text_obj, smp_rec, 0, 0);
    /* Insert new compound text element into the text object at
       index 1 */
    og_insert_cmptext(text_obj, 1);
    /* Change the simple text record's text string */
    smp_rec.str:='And now';
    /* Insert a new simple text element at index 0 in text
```

```
object's compound text element at index 1, using
defined simple text record */
og_insert_smptext(text_obj, smp_rec, 1, 0);
/* Change the simple text record's text string */
smp_rec.str:=' line 2.';
/* Insert a new simple text element at index 1 in text
object's compound text element at index 1, using
defined simple text record */
og_insert_smptext(text_obj, smp_rec, 1, 1);
END;
```

例

このファンクションでは、テキスト・フィールド・オブジェクトのハンドルを引数として使用し、このフィールドに入っているテキストを戻します。複数テキスト要素にしかアクセスしていないので、フェッチされるのはテキスト・フィールドの1行目のテキストのみです。

```
FUNCTION get_text(text_obj IN og_object) RETURN VARCHAR2 IS
    smp_rec    og_smptext_attr;
BEGIN
    /* Set the simple text record's mask, indicating that the text string
    is the only attribute to get */
    smp_rec.mask:=OG_STR_SMPTEXTA;
    /* Get the 0th simple text element in the text object's
    0th compound text element, and store the results in
    the simple text record */
    og_get_smptext(text_obj, 0, 0, smp_rec);
    /* Return the text string attribute of the simple text
    record */
    RETURN(smp_rec.str);
END;
```

タイマー属性レコード

タイマー属性レコードは、タイマーにのみ使用可能な属性で構成されます。

TYPE og_timer_attr IS RECORD
(mask NUMBER(2,0),
name VARCHAR2(255),
interval NUMBER(10,3),
timerproc VARCHAR2(255),
active BOOLEAN
);

Mask Constants:
OG_NAME_TIMERA
OG_INTERVAL_TIMERA
OG_TIMERPROC_TIMERA
OG_ACTIVE_TIMERA
OG_ALL_TIMERA
OG_NONE_TIMERA

	属性	説明
CSG	name	タイマーの名前。

CSG	interval	タイマー・プロシージャの実行間隔の秒数。
CSG	timerproc	タイマーの起動時に実行するプロシージャの名前。

ウィンドウ属性レコード

ウィンドウの位置とサイズは、「画面解像度単位」、つまり一般的にピクセルと呼ばれる単位で表します。画面解像度の水平値と垂直値は、OG_APPと呼ばれるビルトイン・グローバル・レコードに提供されています。このレコードは、OG_APP_ATTRタイプであり、これについては、本章の「アプリケーション属性レコード」の項で詳しく説明しています。

実際の数値のかわりにこのグローバル変数を使用して、アプリケーションが、画面解像度の異なるシステム上で一貫した表示を維持できるようにしてください。

ウィンドウ属性レコードは、ウィンドウにのみ使用可能な属性で構成されます。

```
TYPE og_window_attr is RECORD
(mask  NUMBER(2,0),
 position    og_point,
 width NUMBER(5,0),
 height      NUMBER(5,0),
 name  VARCHAR2(255),
 scrollbars  BOOLEAN,
 helptarget  VARCHAR2(255)
);
```

Mask Constants:

OG_POSITION_WINDOWA
OG_SIZE_WINDOWA
OG_SIZE_WINDOWA
OG_NAME_WINDOWA
OG_SCROLLBARS_WINDOWA
OG_HELPTARGET_WINDOWA

OG_ALL_WINDOWA
OG_NONE_WINDOWA

	属性	説明
CSG	position	ウィンドウの左上角のx座標とy座標（画面解像度単位）。
CSG	width	ウィンドウの幅（画面解像度単位）。
CSG	height	ウィンドウの高さ（画面解像度単位）。
CSG	name	ウィンドウの名前。実行時には、レイアウト・ウィンドウのデフォルト名が「Main Layout」となります。
C	Scrollbars	ウィンドウにスクロール・バーを表示するかどうかを指定します。この属性の値として、次のいずれかを指定できます。 TRUE ウィンドウにスクロール・バーを表示する。 FALSE ウィンドウにスクロール・バーを表示しない。
CSG	helptarget	ウィンドウがアクティブの間に「ヘルプ・システム」を起動した時に表示される、ランタイム・ヘルプ内のハイパーテキスト・ターゲット。

グローバル変数

ビルトイン・グローバル変数

```
OG_App
OG_Inch
OG_Null_Axis
OG_Null_Buttonproc
OG_Null_Display
OG_Null_Ftemp
OG_Null_Layer
OG_Null_Object
OG_Null_Query
OG_Null_Reflines
OG_Null_Sound
OG_Null_Template
OG_Null_Timer
OG_Null_Window
```

OG_App

説明

最初にGraphics BuilderのビルトインPL/SQL構成体が実行されたときの、アプリケーションの属性値のスナップショットが入ります。

構文

```
OG_App  OG_App_Attr;
```

注意: このグローバル変数は、ある時点における値のスナップショットであるため、アプリケーションの属性を変更しても、この変数に影響はありません。たとえば、データベース接続を変更しても、*username*、*password*および*connection*の属性は自動的に更新されません。

OG_Inch

説明

1インチに相当するレイアウト単位数。

構文

```
OG_Inch  NUMBER;
```

OG_Null_Axis

説明

チャート軸のNULLハンドル。

構文

```
OG_Null_Axis  OG_Axis;
```

OG_Null_Buttonproc

説明

ボタン・プロシージャのNULLハンドル。

構文

```
OG_Null_Buttonproc  OG_Buttonproc;
```

OG_Null_Display

説明

画面のNULLハンドル。

構文

```
OG_Null_Display  OG_Display;
```

OG_Null_Ftemp

説明

フィールド・テンプレートのNULLハンドル。

構文

```
OG_Null_Ftemp  OG_Ftemp;
```

OG_Null_Layer

説明

レイヤーのNULLハンドル。

構文

```
OG_Null_Layer  OG_Layer;
```

OG_Null_Object

説明

グラフィック・オブジェクトのNULLハンドル。

構文

```
OG_Null_Object  OG_Object;
```

OG_Null_Query

説明

問合せのNULLハンドル。

構文

```
OG_Null_Query  OG_Query;
```


OG_Null_Refline

説明

参照線のNULLハンドル。

構文

```
OG_Null_Refline  OG_Refline;
```

OG_Null_Sound

説明

サウンドのNULLハンドル。

構文

```
OG_Null_Sound  OG_Sound;
```

OG_Null_Template

説明

チャート・テンプレートのNULLハンドル。

構文

```
OG_Null_Template  OG_Template;
```

OG_Null_Timer

説明

タイマーのNULLハンドル。

構文

```
OG_Null_Timer  OG_Timer;
```

OG_Null_Window

説明

ウィンドウのNULLハンドル。

構文

```
OG_Null_Window  OG_Window;
```

D

Do_Sql, 93
DO_SQL, 92

K

Keep_Password作業環境, 213, 469

L

landscapeプロパティ, 387

N

NULLハンドル, 20

O

OG_Activate_Layer, 87, 89
OG_Activate_Timer, 188
OG_App, 516
og_app_attr属性レコード, 469
OG_Append_Directory, 93
OG_Append_File, 94
OG_Append_Row, 138
og_arc_attr属性レコード, 470
OG_Center, 96

og_chart_attr属性レコード, 478
og_chelement_attr属性レコード, 479
OG_Clear_Query, 139
OG_Clone (オブジェクト), 27
OG_Clone (テンプレート), 173
OG_Close_Display, 17
og_cmptext_attr属性レコード, 510
OG_Connect, 13
og_contaxis_attr属性レコード, 471
OG_Damage (領域), 97
OG_Damage (オブジェクト), 27
OG_Data_Changed, 140
OG_Data_Queried, 141
OG_Deactivate_Timer, 189
OG_Delete_Child, 28
OG_Delete_Cmptext, 30
OG_Delete_Column, 3
OG_Delete_Field, 9
OG_Delete_Ftemp, 174
OG_Delete_Param, 130
OG_Delete_Point, 31
OG_Delete_Property, 32
OG_Delete_Refline, 175
OG_Delete_Smptext, 33
OG_Destroy, 204
OG_Destroy (Query), 142
OG_Destroy (サウンド), 165
OG_Destroy (問合せ), 142
OG_Destroy (ウィンドウ), 204
OG_Destroy (オブジェクト), 34
OG_Destroy (サウンド), 165
OG_Destroy (タイマー), 190
OG_Destroy (テンプレート), 175
og_display_attr, 480
OG_Draw, 35
OG_Execute_Query, 143

OG_Export_Drawing (ウィンドウ), 40
OG_Export_Drawing (オブジェクト/レイヤー), 38
OG_Export_Drawing (図表), 36
OG_Export_Image, 42
OG_Export_Sound, 166
OG_Export_Template, 176
og_font_attr属性レコード, 505
OG_Generate_Display, 18
og_generic_attr属性レコード, 488
OG_Get_Attr (アプリケーション), 98
OG_Get_Attr (ウィンドウ), 108
OG_Get_Attr (グラフィック・オブジェクト), 102
OG_Get_Attr (サウンド), 106
OG_Get_Attr (参照線), 106
OG_Get_Attr (軸), 98
OG_Get_Attr (図表), 99
OG_Get_Attr (タイマー), 107
OG_Get_Attr (問合せ), 105
OG_Get_Attr (フィールド・テンプレート), 100
OG_Get_Attr (プリンタ), 104
OG_Get_Attr (枠テンプレート), 101
OG_Get_Axis, 177
OG_Get_Buttonproc, 109
OG_Get_Char_Param, 131
OG_Get_Char_Property, 44
OG_Get_Charcell, 144
OG_Get_Chart_Element
OG_Get_Child, 45
OG_Get_Cmptext, 46
OG_Get_Column, 6
og_get_ctcount, 428
OG_Get_Date_Param, 132
OG_Get_Date_Property, 47
OG_Get_Datecell, 146
OG_Get_Display, 19
OG_Get_Field, 7
OG_Get_Ftemp, 178
OG_Get_Layer, 88
OG_Get_Newrows, 147
OG_Get_Num_Param, 133
OG_Get_Num_Property, 48
OG_Get_Numcell, 148
OG_Get_Object, 49
OG_Get_Param_Type, 134
OG_Get_Point, 50
OG_Get_Query, 149, 150
OG_Get_Refline, 179
OG_Get_Row, 8
OG_Get_Schema, 150
OG_Get_Smptext, 51
OG_Get_Sound, 167
OG_Get_Template, 180
OG_Get_Timer, 190
OG_Get_Window, 205
og_graphic_attr属性レコード, 490
og_group_attr属性レコード, 493
OG_Help, 110
OG_Hide_Layer, 89
OG_Hide_Window, 206
OG_Host, 110
og_image_attr属性レコード, 493
OG_Import_Drawing, 53
OG_Import_Image, 54
OG_Import_Sound, 167
OG_Import_Template, 181
OG_Inch, 517
OG_Insert_Child, 56
OG_Insert_Cmptext, 57
OG_Insert_Column, 151
OG_Insert_Field, 8
OG_Insert_Ftemp, 182
OG_Insert_Point, 59
OG_Insert_Refline, 183
OG_Insert_Smptext, 60
OG_Isnull, 20
og_line_attr属性レコード, 495
OG_Logged_On, 14
OG_Logoff, 14
OG_Logon, 15
OG_Make_Chart, 9
OG_Make_Ellipse, 61
OG_Make_Group, 62
OG_Make_Image, 63
OG_Make_Line, 64
OG_Make_Poly, 65
OG_Make_Query, 152

OG_Make_Rect, 66
OG_Make_Rrect, 67
OG_Make_Sound, 169
OG_Make_Symbol, 68
OG_Make_Template, 184
OG_Make_Text, 69
OG_Make_Timer, 191
OG_Make_Window, 206
OG_Move, 70
OG_Next_Row, 154
OG_Numcols, 155
OG_Numrows, 156
OG_Open_Display, 22
OG_Param_Exists, 135
OG_Pause, 111
OG_Play_Sound, 170
OG_Point_In, 71
OG_Point_Near, 73
og_poly_attr属性レコード, 496
OG_Print, 112
OG_Property_Exists, 75
og_query_attr属性レコード, 497
OG_Quit, 112
OG_Record_Sound, 171
og_rect_attr属性レコード, 499
OG_Root_Object, 113
OG_Rotate, 76
og_rrect_attr属性レコード, 500
OG_Same, 77
OG_Save_Display, 23
OG_Scale, 79
OG_Set_Attr (アプリケーション), 114
OG_Set_Attr (ウィンドウ), 126
OG_Set_Attr (オブジェクト), 120
OG_Set_Attr (サウンド), 125
OG_Set_Attr (参照線), 124
OG_Set_Attr (軸), 115
OG_Set_Attr (図表), 117
OG_Set_Attr (タイマー), 125
OG_Set_Attr (チャート要素), 116
OG_Set_Attr (問合せ), 123
OG_Set_Attr (フィールド・テンプレート), 118

OG_Set_Attr (フィールド・テンプレート)その他のビルトイン, 118
OG_Set_Attr (プリンタ), 122
OG_Set_Attr (枠テンプレート), 119
OG_Set_Charcell, 158
OG_Set_Datecell, 159
OG_Set_Edgecolor, 81
OG_Set_Fillcolor, 82
OG_Set_Numcell, 160
OG_Set_Param, 135
OG_Set_Property, 83
OG_Set_Schema, 161
OG_Show_Layer, 90
OG_Show_Window, 207
og_smptext_attr属性レコード, 510
og_sound_attr属性レコード, 500
OG_Start_From, 162
OG_Stop_Sound, 171
og_symbol_attr属性レコード, 501
OG_Synchronize, 84
og_text_attr属性レコード, 502
og_timer_attr属性レコード, 512
OG_Translate_Envvar, 127
OG_Update_Bbox, 85
OG_Update_Chart, 11
OG_User_Exit, 128

P

PL/SQLビルトイン概要, 2
PL/SQLボタン・プロシージャ
オブジェクトと対応付ける
PL/SQLを介した, 345
オブジェクトとの関連付け
PL/SQLを使用した, 488
ハンドルの取得, 109

R

RAWデータ型, 144

RUN_PRODUCTビルトイン, 200

T

TOOL_INTのビルトイン・パッケージ

- ADD_PARAMETER, 193
- CREATE_PARAMETER_LIST, 194
- DELETE_PARAMETER, 195
- DESTROY_PARAMETER, 196
- GET_PARAMETER_ATTR, 197
- GET_PARAMETER_LIST, 198
- ISNULL, 199
- RUN_PRODUCT, 200
- SET_PARAMETER_ATTR, 202

TOOLS_INTビルトイン, 193

あ

アクティブ・プロパティ, 452

圧縮

- イメージの, 42, 43

アプリケーション・プロパティ, 210

- カーソル, 211
- 垂直画面解像度, 217
- 垂直レイアウト解像度, 216
- 水平画面解像度, 213
- 接続文字列, 210
- パスワード, 213
- プラットフォーム, 214
- ユーザー名, 215

アプリケーション属性

- 取得, 98

アプリケーション属性レコード, 469, 471

粗さプロパティ, 375

い

位置プロパティ, 248, 377, 459

一般属性, 463

一般属性レコード, 488

一般プロパティ, 345

- イベント, 347
- 内側枠ボックス, 351
- オブジェクト・タイプ, 353
- オブジェクトを隠す, 350
- 親, 355
- 外側枠ボックス, 354
- 問合せ実行, 348
- 名前, 352
- パラメータ設定, 356
- フォーマット・トリガー, 349
- ボタン・プロシージャ, 346
- 列, 346, 347

イベント・プロパティ, 272, 347

イメージ

- 圧縮, 42, 43
- インポート, 54

イメージ・プロパティ, 374

- 粗さ, 375
- 位置, 377
- クリップする四角形, 374
- 高さ, 376
- 幅, 379
- 品質, 378

イメージ結合属性レコード, 493

イメージ属性レコード, 493, 494

印刷, 112

印刷プロパティ, 386

- landscape, 387
- 開始ページ, 390
- 終了ページ, 387
- 出力ファイル, 390
- 名前, 388
- 部数, 386
- ページ・サイズ, 389

インポート

- イメージ, 54, 55
- サウンド, 167, 168
- 描画, 53, 54

う

ウィンドウ

- クローズ, 204
- 名前, 458, 513
- ハンドルの取得, 205
- 非表示, 206
- 表示, 207, 208

ウィンドウ・ビルトイン, 204

- OG_Destroy, 204
- OG_Get_Window, 205
- OG_Hide_Window, 206
- OG_Make_Window, 206
- OG_Show_Window, 208

ウィンドウ・プロパティ, 456

- 位置, 459
- 高さ, 456
- 名前, 458
- 幅, 459, 460
- ヘルプ・ターゲット, 457

ウィンドウの非表示, 206

ウィンドウの表示, 207

内側枠ボックス・プロパティ, 351

え

円グラフ

- 円弧を切り離して突出させる, 273

円グラフ・チャート

- 円弧を切り離して突出させる, 479

円グラフの円弧

- 切り離して突出させる, 273
- 切り離して突出させる, 8

円グラフの円弧を切り離して突出させる, 8, 479

円グラフ枠結合属性レコード, 485

円グラフ枠属性レコード, 485

円弧結合属性レコード, 470

円弧属性レコード, 470

円弧プロパティ, 218

- 基本円弧, 218
- 閉じ形状, 219

塗り形状, 220

お

オーバーラップ・プロパティ, 341

オーバーラップなしプロパティ, 312

オープン

- 図表, 22

オープン・トリガー・プロパティ, 288

「オープン時実行」プロパティ, 395

オブジェクト

- 移動, 70
- 回転, 76, 77
- グループからの削除, 28
- グループへの追加, 56
- サイズ変更, 79
- 削除, 34
- ハンドルの取得, 49
- 複製, 27

オブジェクト・タイプ・プロパティ, 353

オブジェクト固有の属性, 463

オブジェクトの移動, 70

オブジェクトのグループへの追加, 56

オブジェクトのサイズ変更, 79

オブジェクトのスケーリング, 79

オブジェクトの複製, 27

オブジェクトを隠すプロパティ, 350

親プロパティ, 355

か

カーソル

- 最初の行の設定, 162

- 進む, 154

- 問合せの作成, 143

カーソル (マウス)

- 外観の変更, 211, 469

カーソル・プロパティ, 211

カーニング・プロパティ, 434

カーブ調整プロパティ, 337

開始行プロパティ, 282
開始ページ・プロパティ, 390
回転
 オブジェクト, 76, 77
回転角度プロパティ, 369
概要
 属性レコード, 462
隠しレイヤー, 89
カスタム間隔プロパティ, 429
カスタム書式プロパティ, 226
カスタム数値書式プロパティ, 295
「カスタム問合せプロシージャ」プロパティ, 393
カスタム日付書式プロパティ, 294
角の丸い四角形結合属性レコード, 500
角の丸い四角形属性レコード, 500
カラー・プロパティ, 413, 427
カラー回転プロパティ, 331
間隔プロパティ, 439, 453
環境変数, 127
 評価, 127

き

記号結合属性レコード, 501
記号サイズ・プロパティ, 420
記号属性レコード, 501
記号プロパティ, 418
 記号サイズ, 420
 索引, 419
 中央, 418
基本円弧プロパティ, 218
基本四角形プロパティ, 404, 410
基本線軸プロパティ, 291
基本線の値プロパティ, 292
「キャッシュ・タイプ」プロパティ, 392
キャラクタ・セット・プロパティ, 425
近似プロパティ, 435

く

グラフィック・オブジェクト・ビルトイン, 25
 OG_Clone, 27
 OG_Damage, 27
 OG_Delete_Child, 29
 OG_Delete_Cmptext, 30
 OG_Delete_Point, 31
 OG_Delete_Property, 32
 OG_Delete_Smptext, 33
 OG_Destroy, 34
 OG_Draw, 35
 OG_Export_Drawing, 36
 OG_Export_Drawing (ウィンドウ), 40
 OG_Export_Drawing (オブジェクト/レイヤー), 38
 OG_Export_Image, 42
 OG_Get_Char_Property, 44
 OG_Get_Child, 45
 OG_Get_Cmptext, 46
 OG_Get_Date_Property, 47
 OG_Get_Num_Property, 48
 OG_Get_Object, 49
 OG_Get_Point, 50
 OG_Get_Smptext, 51
 OG_Import_Drawing, 53
 OG_Import_Image, 54
 OG_Insert_Child, 56
 OG_Insert_Cmptext, 57
 OG_Insert_Point, 59
 OG_Insert_Smptext, 60
 OG_Make_Ellipse, 61
 OG_Make_Group, 62
 OG_Make_Image, 63
 OG_Make_Line, 64
 OG_Make_Poly, 65
 OG_Make_Rect, 66
 OG_Make_Rrect, 67
 OG_Make_Symbol, 68
 OG_Make_Text, 69
 OG_Move, 70
 OG_Point_In, 71

- OG_Point_Near, 73
- OG_Property_Exists, 76
- OG_Rotate, 76
- OG_Same, 77, 78
- OG_Scale, 79
- OG_Set_Edgecolor, 81
- OG_Set_Fillcolor, 82
- OG_Set_Property, 83
- OG_Synchronize, 84
- OG_Update_Bbox, 85
- クリップ・フラグ・プロパティ, 372
- クリップする四角形プロパティ, 374
- グループ
 - からのオブジェクトの削除, 28
 - 子オブジェクトのハンドルの取得, 45
 - へのオブジェクトの挿入, 56
 - レイヤーとして扱う, 88, 90
 - レイヤーとして使用, 87
- グループ・プロパティ, 372
 - クリップ・フラグ, 372
 - 子の数, 372
- グループ結合属性レコード, 493
- グループ属性レコード, 493
- クローズ
 - ウィンドウ, 204
- クローズ・トリガー・プロパティ, 286
- グローバル変数
 - OG_App, 516

け

- 結合形式プロパティ, 368
- 結合属性レコード, 464
 - 説明, 464
- 原点プロパティ, 436

こ

- 子
 - グループから削除, 28

- グループへの挿入, 56
- 合計値プロパティ, 319
- 格子カウント・プロパティ, 325
- 合成プロパティ, 443
- 項目軸結合属性レコード, 475
- 項目軸属性レコード, 475
- 項目数値書式プロパティ, 310
- 項目幅プロパティ, 293
- 項目日付書式プロパティ, 309
- 項目ラベル・プロパティ, 241, 308
- 固定バウンディング・ボックス・プロパティ, 430
- 子の数プロパティ, 372

さ

- 最小行数プロパティ, 328
- 最小項目数プロパティ, 257
- 最小値プロパティ, 232, 264
- サイズと位置プロパティ, 281
- 最大行数プロパティ, 327
 - 「最大行フラグ」プロパティ, 398
 - 「最大行」プロパティ, 397
- 最大項目数プロパティ, 256
- 最大値プロパティ, 231, 263
- サウンド
 - インポート, 167, 168
 - 再生, 170
 - 再生中の停止, 171
 - 作成
 - PL/SQLを使用した, 169
 - データベースからの選択, 169
 - ハンドルの取得, 167
 - 録音, 171
- サウンド・ビルトイン, 165
 - OG_Destroy, 165
 - OG_Export_Sound, 166
 - OG_Get_Sound, 167
 - OG_Import_Sound, 168
 - OG_Make_Sound, 169
 - OG_Play_Sound, 170
 - OG_Record_Sound, 171

- OG_Stop_Sound, 172
- サウンド・プロパティ, 417
 - 名前, 417
- サウンド属性レコード, 500
- サウンドの再生, 170
- サウンドの録音, 171
- 索引プロパティ, 419
- 削除
 - オブジェクト, 34
 - グループの子, 29
 - 多角形の点, 31
- 作成
 - サウンド
 - PL/SQLを使用した, 169
 - レイヤー, 56
- 作成可能な属性, 468
- 作成方法
 - PL/SQLを使用してのテキスト・オブジェクトの, 502
- 参照線
 - チャートからの削除, 175
- 参照線カウント・プロパティ, 296
- 参照線属性レコード, 499
- 参照線プロパティ, 406
 - 軸, 406
 - 数値, 409
 - 日付値, 407
 - ラベル, 408

し

- 四角形結合属性レコード, 499
- 四角形属性レコード, 499
- 四角形プロパティ, 404
 - 基本四角形, 404
- 軸（一般）プロパティ, 239
 - 位置, 248
 - 項目ラベル, 241
 - 軸タイプ, 240
 - 軸ラベル, 239
 - 主目盛きざみ, 244, 245

- 主目盛格子, 243
- 方向, 242, 243
- 補助目盛きざみ, 246
- 補助目盛格子, 245
- 補助目盛りきざみ数, 247
- 目盛きざみ位置, 251
- 目盛きざみラベル, 250
- 目盛きざみラベル回転, 249
- 軸（項目）プロパティ, 253
 - 最小項目数, 257
 - 最大項目数, 256
 - 自動最小値, 254
 - 自動最大値, 253
 - 数値書式, 258
 - 日付書式, 255
- 軸（値）プロパティ, 260
 - 最小値, 264
 - 最大値, 263
 - 自動最小値, 261
 - 自動最大値, 260
 - 自動主目盛間隔, 262
 - 数値書式, 265, 266
 - スケール, 268, 269
 - ステップ, 269
 - パーセント基準, 267
 - パーセント元, 266
- 軸（日付）プロパティ
 - カスタム書式, 226
 - 最小値, 232
 - 最大値, 231
 - 自動最小値, 223
 - 自動最大値, 222
 - 自動主目盛間隔, 224
 - 四半期書式, 234
 - 週末をスキップ, 235
 - 曜日書式, 227
 - ステップ, 236
 - 月書式, 233
 - 年の最初の月, 228
 - 年書式, 237
 - ラベル, 229, 230
- 軸属性レコード, 476
- 軸タイプ・プロパティ, 240

軸の方向プロパティ, 242
軸フィールド・テンプレート結合属性レコード, 481
軸フィールド・テンプレート属性レコード, 481
軸プロパティ, 336, 406
軸ラベル・プロパティ, 239
軸枠結合属性レコード, 483
軸枠属性レコード, 483
始点プロパティ, 382
自動更新プロパティ, 276
自動最小値プロパティ, 223, 254, 261, 323
自動最大値プロパティ, 222, 253, 260, 322
自動主目盛間隔プロパティ, 224, 262
四半期書式プロパティ, 234
シャドウ・サイズ・プロパティ, 306
シャドウ方向プロパティ, 305
終点プロパティ, 381
週末をスキップ・プロパティ, 235
終了, 112
終了行プロパティ, 277
終了ページ・プロパティ, 387
出力ファイル・プロパティ, 390
取得可能な属性, 468
主目盛きざみプロパティ, 244
主目盛格子プロパティ, 243
曜日書式プロパティ, 227
使用例, 89
ショートカット・ビルトイン, 468
 説明, 468
書体プロパティ, 444

す

垂直画面解像度プロパティ, 217
垂直原点プロパティ, 446
垂直格子プロパティ, 329
垂直文字揃えプロパティ, 445
垂直レイアウト解像度プロパティ, 216
水平画面解像度プロパティ, 213
水平原点プロパティ, 432
水平格子プロパティ, 326
水平文字揃えプロパティ, 431

水平レイアウト解像度プロパティ, 212
数値書式プロパティ, 258, 265, 334
数値プロパティ, 409
図形結合属性レコード, 490
図形属性, 463
図形属性レコード, 490
図形プロパティ, 357
 回転角度, 369
 結合形式, 368
 線種スタイル, 361, 362
 凹凸スタイル, 359, 360
 転送モード, 370
 塗りパターン, 365
 端形式, 360, 361
 バックグラウンド線カラー, 357, 358
 バックグラウンド塗りカラー, 358
 フォアグラウンド塗りカラー, 367
 フォアグラウンド枠, 366
 枠パターン, 363
 枠幅, 364
スケーラブル・フォント・プロパティ, 438
スケーラブルな境界線プロパティ, 437
スケール・プロパティ, 268
スタイル・プロパティ, 441
ステップ・プロパティ, 236, 269
図表
 オープン
 PL/SQLによる, 22
 出力, 112
 のハンドルの取得, 19
 保存, 23
図表属性レコード, 480
図表の保存, 23
図表ビルトイン, 17
 OG_Close_Display, 17
 OG_Generate_Display, 18
 OG_Get_Display, 19
 OG_Isnull, 20, 21
 OG_Open_Display, 22
 OG_Save_Display, 23
図表プロパティ, 286
 オープン・トリガー, 288
 クローズ・トリガー, 286

高さ, 287
幅, 289
日付書式, 287

せ

接続

データベース, 15

接続文字列プロパティ, 210

切断

データベースから ~, 14

設定可能な属性, 468

セル

値の取得, 144, 146, 148

線結合属性レコード, 495

線種スタイル・プロパティ, 361

線属性レコード, 495

線の形式プロパティ, 339

線プロパティ, 380

始点, 382, 383

終点, 381

矢印スタイル, 380

そ

属性, 465, 466, 467, 468

一般, 463, 464

オブジェクト固有, 463

作成可能, 468

取得, 98

取得可能, 468

ショートカット・ビルトイン, 468

図形, 463

設定可能, 468

マスク, 465

マスク定数, 465, 466, 467

属性クラス, 463

属性レコード, 462, 464, 465, 469, 470, 471, 472, 475,
476, 478, 479, 480, 481, 482, 483, 484, 485, 487, 488,

490, 493, 494, 495, 496, 497, 499, 500, 501, 502, 505,
510, 512

アプリケーション, 469

一般, 488

イメージ, 493, 494, 495

イメージ結合, 493

円グラフ枠, 485

円グラフ枠結合, 485

円弧, 470, 471

円弧結合, 470

概要, 462

角の丸い四角形, 500

角の丸い四角形結合, 500

記号, 501, 502

記号結合, 501

クラス, 463

グループ, 493

グループ, 493

グループ結合, 493

結合, 464, 465

項目軸, 475

項目軸結合, 475

サウンド, 500, 501

参照線, 499, 500

四角形, 499

四角形, 499

四角形結合, 499

軸, 476, 477

軸 (値), 471

軸フィールド・テンプレート, 481

軸フィールド・テンプレート結合, 481

軸枠, 483

軸枠結合, 483

図形, 490

図形結合, 490

図表, 480

線, 495, 496

線結合, 495

タイマー, 512, 513

多角形, 496, 497

多角形結合, 496

単一テキスト, 510

値軸, 471

値軸結合, 471
 チャート, 478
 チャート, 478, 479
 チャート結合, 478
 チャート要素, 479, 480
 チャート要素結合, 479
 テキスト, 502, 503, 504, 505
 テキスト概要, 502
 テキスト結合, 502
 問合せ, 497, 498, 499
 日付軸, 472
 日付軸結合, 472
 表枠, 487
 表枠結合, 487
 フィールド・テンプレート, 482, 483
 フォント, 505, 506, 507, 508, 509
 複数テキスト要素, 510
 プリンタ, 497
 マスク, 465
 枠, 484
 外側枠ボックス・プロパティ, 354
 その他のビルトイン, 91
 OG_Append_Directory, 93
 OG_Append_File, 94
 OG_Center, 96
 OG_Damage, 97
 OG_Get_Attr, 98
 OG_Get_Attr (ウィンドウ), 108
 OG_Get_Attr (グラフィック・オブジェクト), 102
 OG_Get_Attr (サウンド), 106
 OG_Get_Attr (参照線), 106
 OG_Get_Attr (軸), 98
 OG_Get_Attr (図表), 99
 OG_Get_Attr (タイマー), 107
 OG_Get_Attr (問合せ), 105
 OG_Get_Attr (フィールド・テンプレート), 100
 OG_Get_Attr (プリンタ), 104
 OG_Get_Attr (枠テンプレート), 101
 OG_Get_Buttonproc, 109
 OG_Help, 110
 OG_Host, 110
 OG_Pause, 111

OG_Quit, 112
 OG_Root_Object, 113
 OG_Set_Attr (アプリケーション), 114
 OG_Set_Attr (ウィンドウ), 126
 OG_Set_Attr (オブジェクト), 120
 OG_Set_Attr (サウンド), 125
 OG_Set_Attr (参照線), 124
 OG_Set_Attr (軸), 115
 OG_Set_Attr (図表), 117
 OG_Set_Attr (タイマー), 125
 OG_Set_Attr (チャート要素), 116
 OG_Set_Attr (問合せ), 123
 OG_Set_Attr (プリンタ), 122
 OG_Set_Attr (枠テンプレート), 119
 OG_Translate_Envvar, 127
 OG_User_Exit, 128
 その他プロパティ, 313

た

タイトル・プロパティ, 283
 第2-Y軸プロパティ, 296
 タイマー
 PL/SQLによるアクティブ化, 188
 ハンドルの取得, 190
 非アクティブ化, 189
 タイマー・ビルトイン, 188
 OG_Activate_Timer, 188
 OG_Deactivate_Timer, 189
 OG_Destroy, 190
 OG_Get_Timer, 190
 OG_Make_Timer, 191
 タイマー・プロパティ, 452
 アクティブ, 452
 間隔, 453
 名前, 454
 プロシージャ, 455
 タイマー属性, 512
 「タイマーで実行」プロパティ, 396
 タイマーの非アクティブ化, 189
 多角形

- から点の削除, 31
- の点の取得, 50
- への点の追加, 59
- 多角形結合属性レコード, 496
- 多角形属性レコード, 496
- 多角形プロパティ, 384
 - 閉じ形状, 384
 - ポイント数, 385
- 高さプロパティ, 287, 376, 456
- 単一テキスト
 - 取得, 51
 - 説明, 502
 - テキスト・オブジェクトからの削除, 33
 - テキスト・オブジェクトへの挿入, 60
- 単一テキスト・カウント・プロパティ, 285
- 単一テキスト・プロパティ, 413
 - カラー, 413
 - テキスト文字列, 415
 - フォント, 414
- 単一テキスト要素属性レコード, 510

ち

- 値軸結合属性レコード, 471
- 値軸属性レコード, 471
- 値書式プロパティ, 321
- チャート
 - 更新, 11, 12
- チャート・テンプレート
 - ハンドルの取得, 180
- チャート・ビルトイン
 - OG_Get_Column, 6
 - OG_Get_Field, 7
 - OG_Insert_Field, 8
 - OG_Make_Chart, 9
 - OG_Update_Chart, 11
- チャート・ビルトイン・プロシージャ, 3
- チャート・プロパティ, 276, 280, 283
 - 開始行, 282
 - サイズと位置, 281
 - 自動更新, 276

- 終了行, 277
- テンプレート, 282, 283
- 問合せ, 279
- チャート結合属性レコード, 478
- チャート属性レコード, 478
- チャートの更新, 11
- チャート要素結合属性レコード, 479
- チャート要素属性レコード, 479
- チャート要素プロパティ, 271
 - イベント, 272
 - 展開, 273
 - 名前, 274
- 中央プロパティ, 418

つ

- 月書式プロパティ, 233

て

- データ
 - 値の取得, 146
 - PL/SQLを介した, 144, 148
 - 問合せの実行, 143
- データ値プロパティ, 311
- データ範囲プロパティ, 280
- データベース
 - 接続, 15
 - 接続の検証, 14
 - 切断, 14
- データベース・ビルトイン
 - DO_SQL, 92
 - OG_Logged_On, 14
 - OG_Logoff, 14
 - OG_Logon, 15
- データベース・ビルトイン・プロシージャ, 13
- テキスト
 - 単一, 60, 61
 - 単一, 33, 51, 52, 502
 - フォント属性レコード, 505

複数, 30, 46, 57, 58, 502
変更, 33
テキスト・オブジェクト
 PL/SQLを使用した作成方法, 502
テキスト・フィールド
 読み込み
 PL/SQLを使用しての, 510
テキスト・プロパティ
 カーニング, 434
 カスタム間隔, 429
 カラー, 427
 間隔, 439, 440
 キャラクタ・セット, 425, 427
 近似, 435
 原点, 436
 合成, 443
 固定バウンディング・ボックス, 430
 書体, 444
 垂直原点, 446
 垂直文字揃え, 445
 水平原点, 432
 水平文字揃え, 431
 スケーラブル・フォント, 438
 スケーラブルな境界線, 437
 スタイル, 441
 バウンディング・ボックスの高さ, 423
 非表示, 433
 フォントの幅, 449
 フォントの太さ, 447, 448
 ポイント・サイズ, 437
 丸め, 451
テキスト結合属性レコード, 502
テキスト属性の概要, 502
テキスト属性レコード, 502, 503
テキスト文字列プロパティ, 415
凹凸スタイル・プロパティ, 359
点
 多角形からの削除, 31
 多角形での取得, 50
 多角形への追加, 59
展開プロパティ, 273
転送モード・プロパティ, 370
テンプレート・ビルトイン, 173

OG_Clone, 173
OG_Delete_Ftemp, 174
OG_Delete_Refline, 175
OG_Destroy, 176
OG_Export_Template, 176
OG_Get_Axis, 177
OG_Get_Ftemp, 178
OG_Get_Refline, 179
OG_Get_Template, 180
OG_Import_Template, 181
OG_Insert_Ftemp, 182
OG_Insert_Refline, 183
OG_Make_Template, 185
テンプレート・プロパティ, 282

と

問合せ

 行数の決定, 156
 実行, 143
 ハンドルの取得, 149
 列数の決定, 155
 「問合せ後トリガー・プロシージャ」プロパティ, 399
問合せ実行プロパティ, 348
 「問合せソース」プロパティ, 400
問合せ属性レコード, 497
 「問合せタイプ」プロパティ, 401
問合せの実行, 143
問合せプロパティ, 279, 392
 オープン時実行, 395
 カスタム問合せ, 393
 キャッシュ・タイプ, 392
 最大行, 397
 最大行フラグ, 398
 タイマーで実行, 396
 問合せ後トリガー・プロシージャ, 399
 問合せソース, 400
 問合せタイプ, 401
 名前, 399
 日付書式, 394, 395

統合

RUN_PRODUCTプロシージャ, 200
閉じ形状プロパティ, 219, 384
年書式プロパティ, 237
年の最初の月プロパティ, 228

な

名前プロパティ, 274, 303, 333, 352, 388, 399, 417, 454, 458

ぬ

塗り形状プロパティ, 220
塗りパターン・プロパティ, 365

は

パーセント基準プロパティ, 267
パーセント書式プロパティ, 314
パーセント値プロパティ, 315
パーセント元プロパティ, 266
バウンディング・ボックス
 更新, 85
バウンディング・ボックスの高さプロパティ, 423
バウンディング・ボックスの幅プロパティ, 424
端形式プロパティ, 360
パスワード
 現行ユーザーの, 213, 469
パスワード・プロパティ, 213
バックグラウンド線カラー・プロパティ, 357
バックグラウンド塗りカラー・プロパティ, 358
幅プロパティ, 289, 379, 459
パラメータ・ビルトイン,
 OG_Delete_Param, 130
 OG_Get_Char_Param, 131
 OG_Get_Date_Param, 132
 OG_Get_Num_Param, 133
 OG_Get_Param_Type, 134

 OG_Param_Exists, 135
 OG_Set_Param, 136
パラメータ設定プロパティ, 356
ハンドル
 NULLかどうかのチェック, 20
 PL/SQLボタン・プロシージャへ, 109
 ウィンドウの, 205
 オブジェクトに, 49
 グループの子, 45
 サウンドの, 167
 図表の, 19
 タイマーの, 190, 191
 チャート・テンプレートの, 180
 問合せに対する, 149
 比較, 77, 78, 79
 レイヤー, 88
凡例中の列数プロパティ, 302
凡例プロパティ, 301

ひ

日付軸結合属性レコード, 472
日付軸属性レコード, 472
日付書式プロパティ, 255, 287, 332
日付値プロパティ, 407
「日付書式」プロパティ, 394
非表示プロパティ, 433
描画
 インポート, 53, 54
表示位置プロパティ, 342
表示形式プロパティ, 343
表示順序プロパティ, 316
表示プロパティ, 318
表示枠プロパティ, 304
表枠結合属性レコード, 487
表枠属性レコード, 487
ビルトイン
 概要, 2
 ショートカット, 468, 469
 説明, 2
 の概要, 2

の説明, 2
ビルトイン・オブジェクト・プロパティ

OG_Print, 112

ビルトイン問合せ, 137

OG_Append_Row, 138

OG_Clear_Query, 139

OG_Data_Changed, 140

OG_Data_Queried, 141

OG_Destroy, 142

OG_Execute_Query, 143

OG_Get_Charcell, 144

OG_Get_Datecell, 146

OG_Get_Newrows, 147

OG_Get_Numcell, 148

OG_Get_Query, 149

OG_Get_Schema, 150

OG_Insert_Column, 151

OG_Make_Query, 152

OG_Next_Row, 154

OG_Numcols, 155

OG_Numrows, 156

OG_Set_Charcell, 158

OG_Set_Datecell, 159

OG_Set_Numcell, 160

OG_Set_Schema, 162

OG_Start_From, 162

品質プロパティ, 378

ふ

フィールド

チャートからの削除, 174

チャートに追加, 8

入手, 7

フィールド・テンプレート（一般）プロパティ, 331

カラー回転, 331

数値書式, 334

名前, 333

日付書式, 332

ルート, 335

フィールド・テンプレート（軸チャート）プロパティ, 336

オーバーラップ, 341

カーブ調整, 337, 338

軸, 336

線の形式, 340

表示位置, 342

表示形式, 343

ラベル回転, 338

フィールド・テンプレート・カウント・プロパティ, 299

フィールド・テンプレート属性レコード, 482

フィルタ・プロパティ, 278

フォアグラウンド線カラー・プロパティ, 366

フォアグラウンド塗りカラー・プロパティ, 367

フォーマット・トリガー・プロパティ, 349

フォント・プロパティ, 414

フォント属性レコード, 505

フォントの幅プロパティ, 449

フォントの太さプロパティ, 447

深さサイズ・プロパティ, 298

複数テキスト

取得, 46

説明, 502

テキスト・オブジェクトからの削除, 30

テキスト・オブジェクトへの挿入, 57

複数テキスト・プロパティ, 285

単一テキスト数, 285

複数テキスト数プロパティ, 428

複数テキスト数, 428

複数テキスト要素属性レコード, 510

部数プロパティ, 386

プラットフォーム・プロパティ, 214

プリンタ属性レコード, 497

プロシージャ・プロパティ, 455

プロパティ

アプリケーション, 210

一般, 345

イメージ, 374

印刷, 386

ウィンドウ, 456

円弧, 218

記号, 418

グループ, 372
サウンド, 417
参照線, 406
四角形, 404
軸（一般）, 239
軸（項目）, 253
軸（値）, 260
図形, 357
図表, 286
線, 380
タイマー, 452
多角形, 384
単一テキスト, 413
チャート, 276
チャート要素, 271
テキスト, 422
問合せ, 392
フィールド・テンプレート（一般）, 331
フィールド・テンプレート（軸チャート）, 336
複数テキスト, 285
丸い四角形, 410
枠（一般）, 298
枠（円グラフ）, 308
枠（軸チャート）, 291
枠（表チャート）, 322

へ

ページ・サイズ・プロパティ, 389
ヘルプ・ターゲット・プロパティ, 457

ほ

ポイント・サイズ・プロパティ, 437
ポイント数プロパティ, 385
補助きざみ数プロパティ, 246
補助目盛きざみプロパティ, 246
補助目盛格子プロパティ, 245
補助目盛りきざみ数プロパティ, 247
ボタン・プロシージャ・プロパティ, 271, 345

ま

マウス・イベント
設定
PL/SQLを介した, 347
PL/SQLを使用した, 488
マスク
属性レコード内の, 465
マスク属性, 465
マスク定数, 465
丸い四角形プロパティ, 410
基本四角形, 410
丸め半径, 411
丸め半径プロパティ, 411
丸めプロパティ, 451

め

目盛きざみ位置プロパティ, 251
目盛きざみプロパティ, 317
目盛きざみラベル・プロパティ, 250
目盛きざみラベル回転プロパティ, 249

や

矢印スタイル・プロパティ, 380

ゆ

ユーザー・イグジット
起動, 128
ユーザー名プロパティ, 215

ら

ラベル・プロパティ, 229, 408

ラベル回転プロパティ, 338

る

ルート・オブジェクト, 49, 56

 ハンドル, 113

ルート・プロパティ, 305, 335

れ

レイアウトの同期化, 84

レイヤー

 PL/SQLを使用した操作, 49

 アクティブ化, 87

 グループ・オブジェクトとして扱う, 88

 削除

 PL/SQLによる, 28

 操作, 56

 ハンドルの取得, 88

 表示, 90

レイヤー・ビルトイン, 87

 OG_Activate_Layer, 87

 OG_Get_Layer, 88

 OG_Hide_Layer, 89

 OG_Show_Layer, 90

レイヤーの隠し

 隠し, 89

レイヤーの表示, 90

列

 問合せへの追加, 151

列プロパティ, 346

列名プロパティ, 324

ろ

ログイン

 に使用するユーザー名の指定, 215

 に使用するユーザー名の指定, 469

わ

枠（一般）プロパティ, 298

 シャドウ・サイズ, 306

 シャドウ方向, 305

 名前, 303

 凡例, 301

 凡例中の列数, 302

 表示枠, 304

 フィールド・テンプレート・カウント, 299

 深さサイズ, 298

 ルート, 305

 枠タイプ, 300

枠（円グラフ）プロパティ, 308

 オーバーラップなし, 312

 合計値, 319

 項目数値書式, 310

 項目日付書式, 309

 項目ラベル, 308

 その他, 313

 値書式, 321

 データ値, 311, 312

 パーセント書式, 314

 パーセント値, 315

 表示, 319

 表示順序, 316

 目盛きざみ, 317

枠（軸チャート）プロパティ, 291

 カスタム数値書式, 295

 カスタム日付書式, 294

 基本線軸, 291

 基本線の値, 292

 項目幅, 293

 参照線カウント, 296

 第2-Y軸, 296

枠（表チャート）プロパティ, 322

 水平格子, 326

 格子カウント, 325

 最大行数, 327

 自動最小値, 323

 自動最大値, 322

 垂直格子, 329

列名, 324	枠パターン・プロパティ, 363
枠属性レコード, 484	枠幅プロパティ, 364
枠タイプ・プロパティ, 300	