Oracle Forms Developer

Form Builder **リファレンス**:Vol.1

リリース 6*i*

2000 **年** 4 **月**

部品番号 : J01127-01

ORACLE®

Oracle Forms Developer Form Builder リファレンス:Vol.1 リリース 6i

部品番号: J01127-01

原本名: Oracle Forms Developer: Form Builder Reference, Release 6i Volume 1

原本部品番号: A73074-01

目次

はじめに

Oracle Forms Developer Form Builderリファレンス、リリース6*i*にようこそ。

このリファレンス・ガイドでは、Oracle Forms Developer Form 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。

- Vol. 1
  - ビルトイン
  - トリガー
- Vol. 2
  - プロパティ
  - オプション
  - システム変数

ここでは、このガイドの構成を説明し、Oracle Forms Developer Form 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の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6*i*』	J00449-01

ビルトイン

ビルトインの概要

Form Builderには、トリガーおよび独自に作成したユーザー命名サブプログラムからコールできるビルトイン・サブプログラムが用意されています。ビルトインによって、ナビゲーション、インタフェース制御およびトランザクション処理などの標準的なアプリケーション機能のプログラムによる制御が可能になります。

この項では、次の項目に関する情報を提供します。

- ビルトイン構文
- ビルトイン名前付きパラメータ
- ビルトイン・コードの例
- ビルトイン・オブジェクトID
- 制限付きビルトイン・サブプログラム
- ビルトイン定数

ビルトイン構文

名前付きパラメータはイタリックのモノスペース・フォントで示されます。名前付きパラメータはどれも、実パラメータで置き換えることができます。実パラメータには定数、リテラル、バインド変数または数値を使用できます。

```
SET_TIMER(timer_name, milliseconds, iterate);
```

この例では、timer_nameはCHAR値なので、指定するタイマー名を引用符で囲む必要があります。millisecondsパラメータは数値として渡されるので、引用符で囲む必要はありません。iterateパラメータは定数として渡されるので、引用符で囲まずに、パラメータの説明に示されているとおりに入力する必要があります。大文字を使用してもかまいません。

いくつかのオプションの要素が変更可能なケースでは、さまざまな代替構文が提示されます。代替構文が提示されるのは、いくつもの複雑な構文規則を解読する必要をなくすためです。

特定のオブジェクト名を組み込むかわりに変数を使用する場合もあることに注意してください。そのような場合には、変数を引用符で囲みません。次の例は、SET_TIMERビルトインをコールして有効なタイマー名を含んでいる変数を参照するWhen-Timer-Expiredトリガーを示したものです。

```
DECLARE
   the_timer  CHAR := GET_APPLICATION_PROPERTY(TIMER_NAME);
```

```
BEGIN
   SET_TIMER(the_timer, 60000, REPEAT);
END;
```

ビルトイン名前付きパラメータ

名前付きパラメータの後には等号/不等号（=>）を続ける必要があります。これは、名前付きパラメータの後に続く実パラメータを示します。たとえば、SET_TIMERビルトインのmillisecondsを変更する場合は、次の構文を使用して直接そのパラメータを使用できます。

```
SET_TIMER(timer_name => 'my_timer', milliseconds => 12000,
               iterate => NO_REPEAT);
```

また、次の構文を使用して引き続きビルトインをコールすることもできます。

```
SET_TIMER('my_timer', 12000, NO_REPEAT);
```

ビルトイン・コードの例

ビルトイン・サブプログラムの例が用意されています。例のなかには、構文を示したものもあります。それ以外の例では、単独または他のビルトインとの連携によるビルトインの使用方法を詳細に示してあります。例の構文に関して留意すべきポイントには、次のものがあります。

- 例は、そのとおりに入力できる形で示されます。
- 大文字およびイタリック体は読みやすくするために使用されているものであり、無視してかまいません。
- ビルトイン名、IF、THEN、ELSE、BEGINおよびENDなどのその他のPL/SQL予約語は、読みやすくするために大文字で表示されます。
- 名前付きパラメータは、イタリック体で示されます。名前付きパラメータを使用する場合は、引用符で囲まずに、表示されているとおりにパラメータ名を入力し、その後に等号/不等号（=>）を続けてください。
- CHAR型の引数は、引用符で囲む必要があります。
- その他のデータ型の引数は、引用符で囲まないでください。
- 引用符（''）、カンマ（,）、カッコ、アンダースコア（_）およびセミコロン（;）以外の特殊文字は無視してください。

ビルトイン・オブジェクトID

一部のビルトイン・サブプログラムは、実パラメータとしてオブジェクトIDを受け入れます。オブジェクトIDは、Form Builderで作成されるときに各オブジェクトに割り当てられる、内部の不透明ハンドルです。オブジェクトIDは内部で管理され、ユーザーが外的には表示できません。オブジェクトIDを取り出す唯一の方法は、ローカル変数またはグローバル変数を使用して、オブジェクトの戻り値をその変数に割り当てることです。

この割当ては、FIND_ビルトイン・ファンクションを介して行います。PL/SQLブロック内でFIND_を使用すれば、その変数を、そのブロックにあるままでオブジェクトIDとして使用できます。各オブジェクトの有効なPL/SQLタイプは、各パラメータの構文説明で示されています。FIND_BLOCKビルトインの説明に、オブジェクトIDを取得する方法の例が示されています。

ビルトイン・フォーム座標単位

多数のビルトイン・サブプログラムで、次のようなプロパティを使用してサイズおよび位置座標を指定できます。

- HEIGHT
- WIDTH
- DISPLAY_POSITION
- VIEWPORT_X_POS
- VIEWPORT_Y_POS
- VIEW_SIZE
- VIEWPORT_X_POS_ON_CANVAS
- VIEWPORT_Y_POS_ON_CANVAS

座標または幅および高さを指定するときは、「フォーム・モジュール」プロパティ・パレットで設定されている現行のフォーム座標システムの単位を使用します。フォームの座標システムでは、Form Builderでオブジェクトのサイズおよび位置座標を指定するための単位が定義されます。次に示すようなフォームの座標単位を設定するには、「座標システム」フォーム・モジュール・プロパティを使用します。

- 文字セルまたは単位:

  インチ

  センチメートル

  ピクセル

ポイント

文字セル座標システムで設計する場合は、オブジェクトの大きさおよび位置座標はすべて文字セルで表されるので、Form Builderではサイズおよび位置プロパティには整数のみが受け入れられます。

実単位（インチまたはセンチメートル、ポイント）を使用して設計する場合は、オブジェクトの大きさおよび位置座標はすべてユーザーが指定した単位で表されるので、Form Builderではサイズおよび位置プロパティに小数および整数が受け入れられます。実単位の精度は3桁なので、座標は小数点以下第3位まで指定できます。ピクセルまたは文字セルを使用する場合は、座標の小数部は切り捨てられて整数になります。

ビルトインの大文字の戻り値

GET_FORM_PROPERTYなどのGET_X_PROPERTYビルトインは、CHAR引数を大文字の値として戻します。これは、IF文で結果を比較する方法に影響します。

制限付きビルトイン・サブプログラム

制限付きビルトインは、フォームのナビゲーション（外部画面ナビゲーションまたは内部ナビゲーションの両方）に影響します。制限付きビルトインは、内部ナビゲーションが発生していないときにトリガーからのみコールできます。

制限付きビルトインはPreおよびPostトリガーからはコールできません。これらのトリガーは、Form Builderがあるオブジェクトから別のオブジェクトへナビゲートするときに起動します。

制限付きビルトインは、When-Button-PressedまたはWhen-Checkbox-Changedなどの、インタフェース項目に固有のWhenトリガーからコールできます。制限付きビルトインは任意のWhen-New-"object"-Instanceトリガーから、およびキー・トリガーからもコールできます。

制限なしビルトインは、論理ナビゲーションにも物理ナビゲーションにも影響を与えず、任意のトリガーからコールできます。

ビルトインの説明には、そのビルトインが制限付きか制限なしかを示す見出し「ビルトイン・タイプ」が含まれています。

ビルトイン定数

ビルトイン・サブプログラムの多くは、引数として数値を取り込みます。このような数値引数に定数が定義されていることがよくあります。定数は名前付きの数値です。定数をビルトインに渡すときは、定数値を引用符で囲まないでください。

定数は、式の中で演算子の右側にのみ現れます。

ビルトインが引数として取り込める定数が複数ある場合もあります。可能な定数は、各パラメータの説明にリストされています。

次の例で、BLOCK_SCOPEはパラメータ定数VALIDATION_UNITに指定できる定数です。説明にリストされているその他の定数には、FORM、RECORDおよびITEMがあります。

```
SET_FORM_PROPERTY('my_form', VALIDATION_UNIT, BLOCK_SCOPE);
```

個々のビルトインの説明

これ以降は、個々のビルトインについての説明です。各ビルトインは、適宜、次の書式で示されます。

構文

ビルトインの構文を説明します。ビルトインに複数の書式がある場合には、それらの書式が示されます。たとえば、ビルトインのターゲット・オブジェクトを名前またはオブジェクトIDによってコールできる場合には、両方の書式の構文が表示されます。

ビルトイン・タイプ

そのビルトインが制限付きか制限なしかを示します。

戻り値

ビルトイン・ファンクションの戻り値またはデータ型を示します。

問合せ入力モード

問合せ入力モード中にそのビルトインをコールできるかどうかを示します。

説明

そのビルトインの一般的な用途と使用方法を示します。

パラメータ

構文図に含まれているパラメータを説明します。通常、下線の付いているパラメータがデフォルトです。

個々のビルトイン説明の制限事項

制限事項を示します。

個々のビルトインの制限事項の例

そのビルトインの実際のコールを開発するために構文とともに使用できる実例を提供します。

ABORT_QUERYビルトイン

説明

現ブロックでオープンされている問合せをクローズします。

問合せは、SELECT文が発行された時からデータベースからすべての行がフェッチされるまでの間、オープンされています。特に、フォームが問合せ入力モードの場合には、SELECT文がまだ発行されていないので、問合せはオープンされていません。

構文

```
PROCEDURE ABORT_QUERY;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

使用上の注意

ABORT_QUERYは「問合せ取消」デフォルト・メニューと等価ではありません。ABORT_QUERYはデータベースからの初期フェッチを防止するのではなく、フェッチ処理に割り込むことによって後続のフェッチを防止します。

ABORT_QUERYの制限事項

ABORT_QUERYは、次のトリガーでは使用しないでください。

- On-Fetch。On-Fetchトリガーは、Oracle以外のデータ・ソースに対して実行するときにデフォルトのForm Builderファンクションの代わりに使用するために、トランザクション・トリガーを使用するアプリケーション用に用意されているものです。On-Fetchトリガーが行のフェッチを終わらせるには、CREATE_QUERIED_RECORDビルトインを発行せずにOn-Fetchトリガーを終了します。

■ Pre-Query。Pre-Queryトリガーは問合せがオープンされる前に起動するので、クローズすべき問合せはなく、ABORT_QUERYは無視されます。問合せ入力モードをプログラムで取り消すには、次のように、When-New-Record-Instanceトリガーを使用してフラグをチェックして、ビルトインEXIT_FORMをコールします。

```
IF (:global.cancel_query = 'Y'
   and :system.mode = 'ENTER-QUERY')
THEN
    Exit_Form;
    :global.cancel_query = 'N';
END IF;
```

次に、Pre-QueryトリガーまたはFRM-40301のエラーが発生した場合のOn-Errorトリガーの処理のどちらかから、フラグを'TRUE'に設定します。

ACTIVATE_SERVERビルトイン

説明

指定された型の列を所定のレコード・グループに追加します。

構文

```
PROCEDURE ACTIVATE_SERVER
  (item_id  Item);
PROCEDURE ACTIVATE_SERVER
  (item_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。

使用上の注意

■ OLEコンテナにはOLEオブジェクトを含み、OLEサーバーはアクティブになっている必要があ

ります。

ACTIVATE_SERVERの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効。

ACTIVATE_SERVERの例

```
/*
** Built-in: ACTIVATE_SERVER
** Example:  Activates the OLE server associated with the object
**          in the OLE container.
** Trigger:  When-Button-Pressed
*/
DECLARE
 item_id  ITEM;
 item_name VARCHAR(25) := 'OLEITM';
BEGIN
 item_id := Find_Item(item_name);
 IF Id_Null(item_id) THEN
  message('No such item:'||item_name);
 ELSE
  Forms_OLE.Activate_Server(item_id);
 END IF;
END;
```

ADD_GROUP_COLUMNビルトイン

説明

指定された型の列を所定のレコード・グループに追加します。

構文

```
FUNCTION ADD_GROUP_COLUMN
  (recordgroup_id    RecordGroup,
   groupcolumn_name  VARCHAR2,
   column_type       NUMBER);
FUNCTION ADD_GROUP_COLUMN
  (recordgroup_name  VARCHAR2,
   groupcolumn_name  VARCHAR2,
   column_type       NUMBER);
FUNCTION ADD_GROUP_COLUMN
  (recordgroup_id,   RecordGroup
```

```
   groupcolumn_name  VARCHAR2,
   column_type       NUMBER,
   column_width      VARCHAR2);
FUNCTION ADD_GROUP_COLUMN
  (recordgroup_name  VARCHAR2,
   groupcolumn_name  VARCHAR2,
   column_type       NUMBER,
   column_width      VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

戻り値

GroupColumn

パラメータ

recordgroup_id	Form Builderがグループを作成するときに割り当てる一意のID。そのIDのデータ型はRECORDGROUPです。
recordgroup_name	ユーザーがレコード・グループを作成するときに付けた名前。その名前のデータ型はVARCHAR2です。
groupcolumn_name	列の名前を指定します。列名のデータ型はVARCHAR2です。
column_type	列のデータ型を指定します。許容される値は、次の定数です。 VARCHAR_COLUMN その列がVARCHAR2データのみを受け入れられる場合、これを指定します。 DATE_COLUMN その列がDATEデータのみを受け入れられる場合、これを指定します。 LONG_COLUMN その列がLONGデータのみを受け入れられる場合、これを指定します。 NUMBER_COLUMN その列がNUMBERデータのみを受け入れられる場合、これを指定します。
column_width	column_typeとしてCHAR_COLUMNを指定した場合は、データの最大長を指示する必要があります。COLUMN_WIDTHは2000を超えてはならず、整数としてす必要があります。

エラー条件:

次の条件のもとではエラーが戻されます。

- 存在しないレコード・グループの名前を入力しました。
- グループまたは列の名前に、Oracleの標準命名規則に従っていない名前を指定しました。
- VARCHAR2、NUMBER、DATEまたはLONG以外の列型を入力しました。

ADD_GROUP_COLUMNの制限事項

- グループへの列の追加は、行を追加する前に行う必要があります。
- すでに行を持っているグループには列を追加できません。その場合は、DELETE_GROUP_ROWで行を削除してから列を追加します。
- グループへの列の追加は、CREATE_GROUPのコールを使用してグループを作成した後でしか実行できません。
- 列がデータベースの列に対応する場合、VARCHAR2型の列の幅は対応するデータベースの列の幅より小さくしないでください。
- 列がデータベースの列に対応する場合、VARCHAR2型の列の幅は対応するデータベースの列の幅より大きくてもかまいません。
- widthパラメータを必要とするのは、VARCHAR2型の列のみです。
- レコード・グループに多数の列があると、パフォーマンスに影響します。
- LONG列は、1つのレコード・グループに1つのみ存在できます。

ADD_GROUP_COLUMNの例

```
/*
** Built-in:  ADD_GROUP_COLUMN
** Example: Add one Number and one Char column to a new
**          record group.
*/
PROCEDURE Create_My_Group IS
  rg_name VARCHAR2(15) := 'My_Group';
  rg_id   RecordGroup;
  gc_id   GroupColumn;
BEGIN
  /*
  ** Check to see if Record Group already exists
  */
  rg_id := Find_Group( rg_name );
  /*
  ** If Not, then create it with one number column and one          ** Char
```

```
column
  */
  IF Id_Null(rg_id) THEN
    rg_id := Create_Group( rg_name );
    gc_id := Add_Group_Column(rg_id, 'NumCol',NUMBER_COLUMN);
    gc_id := Add_Group_Column(rg_id, 'CharCol',VARCHAR2_COLUMN,15);
  END IF;
END;
```

ADD_GROUP_ROWビルトイン

説明

所定のレコード・グループに行を追加します。

構文

```
PROCEDURE ADD_GROUP_ROW
  (recordgroup_id  RecordGroup,
   row_number      NUMBER);
PROCEDURE ADD_GROUP_ROW
  (recordgroup_name  VARCHAR2,
   row_number        NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

recordgroup_id	Form Builderがグループを作成するときに割り当てる一意のID。そのIDのデータ型はRECORDGROUPです。
recordgroup_name	ユーザーがレコード・グループを作成するときに付けた名前。その名前のデータ型はVARCHAR2です。
row_number	グループ内の行を指定する整数。グループの最後を除く任意の位置に行を追加すると、その行より後の行はすべて、論理的に再番号付けされます。グループの終わりに行を追加するには、END_OF_GROUP定数を使用します。

エラー条件:

次のいずれかの条件のもとでは、ランタイム・エラーが戻されます。

- 存在しないレコード・グループの名前を入力した場合。
- 範囲外または無効な行番号（アルファベット文字など）を指定した場合。

ADD_GROUP_ROWの制限事項

- グループは、0以上の行から構成できます。
- グループへの行の追加は、グループが作成され、列が追加された後でのみ実行できます。
- グループ内にすでにある行の数よりも大きい（または負数の）行番号を指定すると、その行はグループの終わりに挿入されます。
- 問合せなしで静的グループに対して行を追加することはできません。

ADD_GROUP_ROWの例

```
/*
** Built-in:  ADD_GROUP_ROW
** Example:Add ten rows to a new record group and populate.
*/
PROCEDURE Populate_My_Group IS
  rg_name  VARCHAR2(20) := 'My_Group';
  rg_col1  VARCHAR2(20) := rg_name||'.NumCol';
  rg_col2  VARCHAR2(20) := rg_name||'.CharCol';
  rg_id    RecordGroup;
  gc_id    GroupColumn;
  in_words VARCHAR2(15);
BEGIN
  /*
  ** Check to see if Record Group already exists
  */
  rg_id := Find_Group( rg_name );
  /*
  ** If it does, then clear all the rows from the group and
  ** populate ten rows with the numbers from 1..10 along
  ** with the equivalent number in words.
  **
  **    Row#    NumCol    CharCol
  **    ----    ------    -------
  **      1       1       one
```

```
  **      2       2       two
  **      ::      :        :
  **     10      10       ten
  */
  IF NOT Id_Null(rg_id) THEN
    Delete_Group_Row( rg_id, ALL_ROWS );
    FOR i IN 1..10 LOOP
      /*
      ** Add the i-th Row to the end (bottom) of the
      ** record group, and set the values of the two cells
      */
      in_words := TO_CHAR(TO_DATE(i,'YYYY'),'year');
      Add_Group_Row( rg_id, END_OF_GROUP );
      Set_Group_Number_Cell( rg_col1, i, i);
      Set_Group_Char_Cell( rg_col2, i, in_words);
    END LOOP;
  END IF;
END;
```

ADD_LIST_ELEMENTビルトイン

説明

リスト項目に1つの要素を追加します。

構文

```
PROCEDURE ADD_LIST_ELEMENT
  (list_name   VARCHAR2,
   list_index, NUMBER
   list_label  VARCHAR2,
   list_value  NUMBER);
PROCEDURE ADD_LIST_ELEMENT
  (list_id     ITEM,
   list_index  VARCHAR2,
   list_label  VARCHAR2,
   list_value  NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

list_id	Form Builderがリスト項目を作成するときに割り当てる一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
list_name	ユーザーがリスト項目を作成するときに付けた名前。その名前のデータ型はVARCHAR2です。
list_index	リスト索引値を指定します。リスト索引は1を基礎としています。
list_label	リスト要素のラベルとして表示するVARCHAR2文字列を指定します。
list_value	リスト項目に追加する実際のリスト要素値。

ADD_LIST_ELEMENTの制限事項

Form Builderでは、「リスト形式」プロパティが「ポップリスト」または「Tlist」に設定されている実表リストの場合、ブロックに問合せまたは変更されたレコードが含まれているときにはその他の値要素を追加できません。これを実行すると、エラーが発生します。こうした状況は、設計時に「ほかの値のマッピング」リスト項目プロパティで指定された「その他の値」を、DELETE_LIST_ELEMENTまたはCLEAR_LISTを使用してすでに削除しているときに発生する場合があります。

注意: 問合せされたレコードがブロックに含まれている場合、ブロック・ステータスはQUERYです。挿入または更新されたレコードがブロックに含まれている場合、ブロック・ステータスはCHANGEDです。

ADD_LIST_ELEMENTの例

```
/*
**  Built-in:  ADD_LIST_ELEMENT
**  Example:   Deletes index value 1 and adds the value "1994" to
**           the list item called years when a button is pressed.
**  Trigger:  When-Button-Pressed
*/
BEGIN
   Delete_List_Element('years',1);
   Add_List_Element('years', 1, '1994', '1994');
END;
```

ADD_OLEARGSビルトイン

説明

OLEオブジェクトのメソッドに渡される引数の型と値を確立します。

構文

```
PROCEDURE ADD_OLEARG
  (newvar NUMBER, vtype VT_TYPE := VT_R8);

PROCEDURE ADD_OLEARG
  (newvar VARCHAR2, vtype VT_TYPE := VT_BSTR);

PROCEDURE ADD_OLEARG
  (newvar OLEVAR, vtype VT_TYPE := VT_VARIANT);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

Newvar	**この引数の値。型は（NUMBER、VARCHAR2またはOLEVAR）、FORMSデータ型またはPL/SQLデータ型です。**
vtype	**OLEメソッドによって解釈される引数の型。** **NUMBER引数の場合、デフォルトはVT_TYPE := VT_R8。** **VARCHAR2引数の場合、デフォルトはVT_TYPE := VT_BSTR。** **OLEVAR引数の場合、デフォルトはVT_TYPE := VT_VARIANT。**

使用上の注意

渡される引数ごとに、個別のADD_OLEARGコールが必要です。このコールは、最初の引数から順番に行う必要があります。

サポートされるOLE VT_TYPEのリストは、OLEバリアント型に記載されています。

ADD_PARAMETERビルトイン

説明

パラメータ・リストにパラメータを追加します。各パラメータは、キー、そのタイプおよび対応付けられた値から構成されます。

構文

```
PROCEDURE ADD_PARAMETER
  (list      VARCHAR2,
   key       VARCHAR2,
   paramtype  VARCHAR2,
   value     VARCHAR2);
PROCEDURE ADD_PARAMETER
  (name      VARCHAR2,
   key       VARCHAR2,
   paramtype  VARCHAR2,
   value     VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

list*または*name	**パラメータを割り当てるパラメータ・リストを指定します。実際のパラメータには、型PARAMLISTのパラメータ・リストIDまたはパラメータ・リストのVARCHAR2名のどちらかを指定できます。**
key	**パラメータの名前。キーのデータ型はVARCHAR2です。**
paramtype	**次の2つの型のどちらか1つを指定します。** **TEXT_PARAMETER　VARCHAR2文字列リテラル。** **DATA_PARAMETER　現フォームで定義されているレコード・グループの名前を指定するVARCHAR2文字列。Form BuilderがReport BuilderまたはGraphics Builderにデータ・パラメータを渡すと、Report BuilderまたはGraphics Builderがレポートまたは表示を行うために通常実行する問合せの代わりに、指定されたレコード・グループ内のデータを使用できます。**
value	**コールされたモジュールに渡す実際の値。テキスト・パラメータを渡す場合、最大長は64K文字です。値のデータ型はVARCHAR2です。**

ADD_PARAMETERの制限事項

- パラメータ・リストは、0（ゼロ）以上のパラメータから構成できます。
- すでにパラメータ・リストが存在する場合は、パラメータ・リストは作成できません。作成しようとするとエラーが発生します。このエラーを回避するには、パラメータ・リストを作成する前に、ID_NULLを使用してパラメータ・リストがすでに存在しているかどうかをチェッ

クします。パラメータ・リストがすでに存在する場合は、DESTROY_PARAMETER_LISTを使用してそのパラメータ・リストを削除した後、新しいリストを作成してください。

- パラメータ・リストが別のフォームに渡される場合、DATA_PARAMETER型のパラメータを追加することはできません。

ADD_PARAMETERの例

```
/*
** Built-in:  ADD_PARAMETER
** Example: Add a value parameter to an existing Parameter
**          List 'TEMPDATA', then add a data parameter to
**          the list to associate named query 'DEPT_QUERY'
**          with record group 'DEPT_RECORDGROUP'.
*/
DECLARE
  pl_id ParamList;
BEGIN
  pl_id := Get_Parameter_List('tempdata');
  IF NOT Id_Null(pl_id) THEN
    Add_Parameter(pl_id,'number_of_copies',TEXT_PARAMETER,'19');

    Add_Parameter(pl_id, 'dept_query', DATA_PARAMETER,
        'dept_recordgroup');
  END IF;
END;
```

ADD_TREE_DATAビルトイン

説明

階層化された3つの項目の指定されたノードの下に1つのデータ・セットを追加します。

構文

```
PROCEDURE ADD_TREE_DATA
  (item_id ITEM,
   node FTREE.NODE,
   offset_type NUMBER,
   offset NUMBER,
   data_source NUMBER,
   data VARCHAR2);
PROCEDURE ADD_TREE_DATA
  (item_name VARCHAR2,
```

```
   node FTREE.NODE,
   offset_type NUMBER,
   offset NUMBER,
   data_source NUMBER,
   data VARCHAR2);
PROCEDURE ADD_TREE_DATA
  (item_name VARCHAR2,
   node FTREE.NODE,
   offset_type NUMBER,
   offset NUMBER,
   data_source NUMBER,
   data RECORDGROUP);
PROCEDURE ADD_TREE_DATA
  (item_id ITEM,
   node FTREE.NODE,
   offset_type NUMBER,
   offset NUMBER,
   data_source NUMBER,
   data RECORDGROUP);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
node	有効なノードを指定します。
offset_type	ノードのオフセットのタイプを指定します。指定できる値は次のとおりです。 PARENT_OFFSET SIBLING_OFFSET *offset_type*がPARENT_OFFSETである場合は、*offset*で示される子の中にある指定のノードの1つ下のレベルにデータ・サブセットを追加します。 *offset_type*がSIBLING_OFFSETの場合、新規のデータを兄弟として指定のノードに追加します。

offset	新しいノードの位置を示します。 *offset_type*がPARENT_OFFSETの場合、*offset*は1-nまたはLAST_CHILDのいずれかとなります。 *offset_type*がSIBLING_OFFSETの場合、*offset*はNEXT_NODEまたはPREVIOUS_NODEのいずれかとなります。
data_source	データ・ソースのタイプを示します。指定できる値は次のとおりです。 RECORD_GROUP QUERY_TEXT
data	追加するデータを指定します。データ･ソースがQUERY_TEXTの場合、データは問合せのテキストとなります。データ・ソースがRECORD_GROUPの場合、データはタイプRECORDGROUPの項目またはレコード・グループの名前となります。

ADD_TREE_DATAの例

```
/*
** Built-in:  ADD_TREE_DATA
*/

-- This code copies a set of values from a record group
-- and adds them as a top level node with any children
-- nodes specified by the structure of the record group.
-- The new top level node will be inserted as the last
-- top level node.
DECLARE
   htree        ITEM;
   rg_data      RECORDGROUP;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Find the record group.
   rg_data := FIND_GROUP('new_data_rg');

   -- Add the new node at the top level and children.
   Ftree.Add_Tree_Data(htree,
                       Ftree.ROOT_NODE,
                       Ftree.PARENT_OFFSET,
                       Ftree.LAST_CHILD,
                       Ftree.RECORD_GROUP,
                       rg_data);
END;
```

ADD_TREE_NODEビルトイン

説明

階層化されたツリー項目に1つのデータ要素を追加します。

構文

```
FUNCTION ADD_TREE_NODE
  (item_name VARCHAR2,
   node FTREE.NODE,
   offset_type NUMBER,
   offset NUMBER,
   state NUMBER,
   label VARCHAR2,
   icon VARCHAR2,
   value VARCHAR2);
FUNCTION ADD_TREE_NODE
  (item_id ITEM,
   node FTREE.NODE,
   offset_type NUMBER,
   offset NUMBER,
   state NUMBER,
   label VARCHAR2,
   icon VARCHAR2,
   value VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

戻り値

NODE

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。

node	有効なノードを指定します。
offset_type	ノードのオフセットのタイプを指定します。指定できる値は次のとおりです。 PARENT_OFFSET SIBLING_OFFSET
offset	新しいノードの位置を示します。 *offset_type*がPARENT_OFFSETの場合は、*offset*は1-nまたはLAST_CHILDのいずれかとなります。 *offset_type*がSIBLING_OFFSETの場合、*offset*はNEXT_NODEまたはPREVIOUS_NODEのいずれかとなります。
state	ノードの状態を指定します。指定できる値は次のとおりです。 COLLAPSED_NODE EXPANDED_NODE LEAF_NODE
label	ノードに表示されたテキスト。
icon	ノードのアイコンのファイル名。
value	ノードのVARCHAR2値を指定します。

ADD_TREE_NODEの例

```
/*
** Built-in:  ADD_TREE_NODE
*/

-- This code copies a value from a Form item and
-- adds it to the tree as a top level node.The
-- value is set to be the same as the label.

DECLARE
   htree        ITEM;
   top_node     FTREE.NODE;
   new_node     FTREE.NODE;
   item_value   VARCHAR2(30);
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Copy the item value to a local variable.
   item_value := :wizard_block.new_node_data;

   -- Add an expanded top level node to the tree
   -- with no icon.
```

```
  new_node := Ftree.Add_Tree_Node(htree,
                  Ftree.ROOT_NODE,
                  Ftree.PARENT_OFFSET,
                  Ftree.LAST_CHILD,
                  Ftree.EXPANDED_NODE,
                  item_value,
                  NULL,
                  item_value);
END;
```

APPLICATION_PARAMETERビルトイン

説明

「パラメータ値入力」ダイアログ・ボックス内に、現メニューに関係するすべてのパラメータおよびそれらの現行の設定値を表示します。

構文

```
PROCEDURE APPLICATION_PARAMETER;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

エラー: 現行のメニューに対してパラメータが指定されないと、Form Builderは「FRM-10201: パラメータは必要ありません。」というエラー・メッセージを出します。

BELLビルトイン

説明

次に端末画面がフォームの内部状態との同期をとるときに、端末のブザーを鳴らすように設定します。この同期は、内部処理の結果またはSYNCHRONIZEビルトイン・サブプログラムへのコールの結果として起こります。

構文

```
PROCEDURE BELL;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

BELLの例

次の例では、ブザーを3回鳴らします。

```
FOR i in 1..3 LOOP
  BELL;
  SYNCHRONIZE;
END LOOP;
```

BLOCK_MENUビルトイン

説明

フォーム内の有効なブロックの順序番号と名前を含む値リスト（LOV）を表示します。Form Builderは、ユーザーが値リストから選択するブロック内の最初の入力可能項目に入力フォーカスを設定します。

構文

```
PROCEDURE BLOCK_MENU;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可。ただし、問合せ入力モードで現ブロック外に移動することは無効です。

パラメータ

なし

BLOCK_MENUの例

```
/*
** Built-in:  BLOCK_MENU
** Example:   Calls up the list of blocks in the form when the
**            user clicks a button, and prints a message if
**            the user chooses a new block out of the list to
**            which to navigate.
*/
DECLARE
  prev_blk  VARCHAR2(40) := :System.Cursor_Block;
BEGIN
  BLOCK_MENU;
  IF :System.Cursor_Block <> prev_blk THEN
    Message('You successfully navigated to a new block!');
  END IF;
END;
```

BREAKビルトイン

説明

現行のフォームがデバッグ・モードで実行されているときに、フォームの実行を停止し、デバッガを表示します。デバッガからの選択により、グローバル変数とシステム変数の値を表示できます。BREAKビルトインは、主に、トリガーの実行時にフォームの状態を調べる場合に役立ちます。

構文

```
PROCEDURE BREAK;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

BREAKの制限事項

現行のフォームがデバッグ・モードで実行されていない場合、BREAKビルトイン・サブプログラムのコールを発行しても、何の効果もありません。

BREAKの例

```
/*
** Built-in:  BREAK
** Example:   Brings up the debugging window for a particular
**            value of the 'JOB' item anytime the user
**            changes records.
** Trigger:   When-New-Record-Instance
*/
BEGIN
  IF :Emp.Job = 'CLERK' THEN
    Break;
    Call_Form('clerk_timesheet');
    Break;
  END IF;
END;
```

CALL_FORMビルトイン

説明

親フォームをアクティブにしたままで、表示されているフォームを実行します。Form Builderでは、これと同じ「Runform」設定項目のあるコール先フォームが親フォームとして実行されます。コール先フォームが終了すると、CALL_FORMへのコールを開始したポイントからForm Builderの処理がコール側フォームで再開されます。

構文

```
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2);
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2,
   display          NUMBER);
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2,
   display          NUMBER,
   switch_menu      NUMBER);
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2,
   display          NUMBER,
   switch_menu      NUMBER,
```

```
   query_mode      NUMBER);
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2,
   display         NUMBER,
   switch_menu      NUMBER,
   query_mode       NUMBER,
   data_mode        NUMBER);
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2,
   display         NUMBER,
   switch_menu      NUMBER,
   query_mode       NUMBER,
   paramlist_id     PARAMLIST);
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2,
   display         NUMBER,
   switch_menu      NUMBER,
   query_mode       NUMBER,
   paramlist_name   VARCHAR2);
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2,
   display         NUMBER,
   switch_menu      NUMBER,
   query_mode       NUMBER,
   data_mode        NUMBER,
   paramlist_id     PARAMLIST);
PROCEDURE CALL_FORM
  (formmodule_name  VARCHAR2,
   display         NUMBER,
   switch_menu      NUMBER,
   query_mode       NUMBER,
   data_mode        NUMBER,
   paramlist_name   VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

formmodule_name	コール先フォームの名前（引用符（'）で囲む）。データ型はVARCHAR2です。
display	HIDE（デフォルト） コール側フォームを非表示にしてから、コール先フォームを描画します。 NO_HIDE コール側フォームを非表示にしないで、コール先フォームを表示します。

switch_menu	NO_REPLACE（デフォルト） Form Builderは、コール先フォームのために、コール側フォームのデフォルト・メニュー・モジュールをアクティブの状態に保ちます。 DO_REPLACE　Form Builderはコール側フォームのデフォルト・メニュー・モードをコール先フォームのデフォルト・メニュー・モジュールに置換します。
query_mode	NO_QUERY_ONLY（デフォルト） Form Builderは、指示されたフォームを通常モードで実行します。この場合、エンド・ユーザーは、コール先フォーム内で挿入、更新、および削除ができます。 QUERY_ONLY　Form Builderは、指示されたフォームを問合せ専用モードで実行します。この場合、エンド・ユーザーは、問合せを実行できますが、レコードの挿入、更新、削除はできません。
data_mode	NO_SHARE_LIBRARY_DATA（デフォルト） Form Builderは（設計時に）同一のライブラリが付加されているフォームの間で実行時にデータを共有します。 SHARE_LIBRARY_DATA 実行時に、Form Builderは（設計時に）同一のライブラリが付加されているフォームの間でデータを共有します。
paramlist_id	Form Builderがパラメータ・リストを作成するときに割り当てる一意のID。オプションで、コール先フォームへの最初の入力としてパラメータ・リストを組み込むことができます。データ型はPARAMLISTです。
paramlist_name	ユーザーがパラメータ・リスト・オブジェクトを定義するときに付けた名前。データ型はVARCHAR2です。

CALL_FORMの制限事項

- コール側フォームがQUERY_ONLYモードで実行されているときは、Form Builderはquery_modeパラメータを無視します。コール先フォームをNO_QUERY_ONLY（標準）モードで実行するように、CALL_FORM構文で指定している場合でも、Form Builderでは、QUERY_ONLYフォームからコールされたすべてのフォームをQUERY_ONLYフォームとして実行します。

- CALL_FORMを介してフォームに渡されるパラメータ・リストには、型DATA_PARAMETERのパラメータを含むことはできません。CALL_FORMで渡すことができるのは、テキスト・パラメータのみです。

- CALL_FORMに割り当てられたメモリのなかには、Runformセッションが終了するまでその割当てを解除できないものがあります。コール先フォームの大きなスタックを作成する場合に、警告を出します。?

- CALL_FORMをPre-Logon、On-LogonまたはPost-Logonトリガーで実行する場合は、常に、DO_REPLACEパラメータを指定してコール側フォームのメニューをコール先フォームのメニューと置き換えます。DO_REPLACEを指定できないと、コール側フォームに対してメニュー

が表示されません??（代わりの解決策としては、REPLACE_MENUビルトインをコール先フォーム内のWhen-New-Form-Instanceトリガーからコールする方法があります）。

CALL_FORMの例

```
/* Example 1:
** Call a form in query-only mode.
*/
BEGIN
  CALL_FORM('empbrowser', no_hide, no_replace, query_only);
END;

/* Example 2:
** Call a form, pass a parameter list (if it exists)
*/
DECLARE
  pl_id       PARAMLIST;
  theformname  VARCHAR2(20);
BEGIN
  theformname := 'addcust';

  /* Try to lookup the 'TEMPDATA' parameter list */
  pl_id := GET_PARAMETER_LIST('tempdata');
  IF ID_NULL(pl_id) THEN
    CALL_FORM(theformname);
  ELSE
    CALL_FORM(theformname,
              hide,
              no_replace,
              no_query_only,
              pl_id);
  END IF;

  CALL_FORM('lookcust', no_hide, do_replace, query_only);
END;
```

CALL_INPUTビルトイン

説明

エンド・ユーザーからファンクション・キーの入力を受け付け、処理します。CALL_INPUTを終了すると、Form BuilderはCALL_INPUTへのコールが発生した時点から処理を再開します。

構文

```
PROCEDURE CALL_INPUT;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

CALL_INPUTの制限事項

CALL_INPUTは、前のバージョンとの互換性を保つために組み込まれています。新規のアプリケーションにはこのビルトインを組み込まないでください。

CALL_OLEビルトイン

説明

識別されたOLEオブジェクトのメソッドに制御を渡します。

構文

```
PROCEDURE CALL_OLE
(obj OLEOBJ, memberid PLS_INTEGER);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

obj	OLEオブジェクトの名前。
memberid	実行するメソッドのメンバーID。

使用上の注意

- このコールを発行する前に、INIT_OLEARGSプロシージャおよびADD_OLEARGSプロシージャを使用して、引数の番号、タイプおよび値を設定しておく必要があります。
- プロシージャ・コールでは、値は戻されません。メソッドから戻り値を取得するには、この

コール（CALL_OLE_CHAR、_NUM、_OBJまたは_VAR）のいずれかのファンクションを使用します。

- このメソッドにより、FORM_OLE_FAILURE例外が発生することがあります。その場合は、ファンクションLAST_OLE_EXCEPTIONを使用して、詳細情報を取得できます。

CALL_OLE_<returntype>ビルトイン

説明

識別されたOLEオブジェクトのメソッドに制御を渡します。指定したタイプの戻り値を受け取ります。

ファンクションには4つのバージョンがあり（returntypeの値によって示される）、引数の型CHAR、NUM、OBJおよびVARについて1つずつあります。

構文

```
FUNCTION CALL_OLE_CHAR
  (obj OLEOBJ, memberid PLS_INTEGER)
RETURN returnval VARCHAR2;

FUNCTION CALL_OLE_NUM
  (obj OLEOBJ, memberid PLS_INTEGER)
RETURN returnval NUMBER;

FUNCTION CALL_OLE_OBJ
  (obj OLEOBJ, memberid PLS_INTEGER)
RETURN returnval OLEOBJ;

FUNCTION CALL_OLE_VAR
  (obj OLEOBJ, memberid PLS_INTEGER)
RETURN returnval OLEVAR;
```

ビルトイン・タイプ

制限なしファンクション

Returns

指定した書式によるメソッドの戻り値。

パラメータ

obj	OLEオブジェクトの名前。
memberid	オブジェクトのメソッドのメンバーID。

使用上の注意

- コールを発行する前に、INIT_OLEARGSプロシージャおよびADD_OLEARGSプロシージャを使用して、引数の番号、タイプおよび値を設定する必要があります。
- メソッドにより、FORM_OLE_FAILURE例外が発生することがあります。その場合は、ファンクションLAST_OLE_EXCEPTIONを使用して、詳細情報を取得できます。

CANCEL_REPORT_OBJECTビルトイン

説明

長時間実行されている非同期レポートを取り消します。REPORT_OBJECT_STATUSによりレポートの状態チェックして、レポートが取り消されたか確認してください。

構文

```
PROCEDURE CANCEL_REPORT_OBJECT
  (report_id VARCHAR2
);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

report_id	**RUN_REPORT_OBJECTビルトインから戻されるVARCHAR2値。この値はローカル・レポート・サーバーまたはリモート・レポート・サーバーのいずれかで現在実行中のレポートを個々に識別します。**

使用上の注意

- CANCEL_REPORT_OBJECTは、レポートが非同期で実行されている場合にのみ有効です。同期モードで実行されているレポートを取り消すことはできません。

CHECKBOX_CHECKEDビルトイン

説明

CHECKBOX_CHECKEDファンクションへのコールによって、指定のチェックボックスの状態を示すBOOLEAN値が戻されます。項目がチェックボックスでない場合、Form Builderは次のエラーを戻します。
FRM-41038:項目<item_name>はチェックボックスではありません。

構文

```
FUNCTION CHECKBOX_CHECKED
  (item_id  ITEM);
FUNCTION CHECKBOX_CHECKED
  (item_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

BOOLEAN

問合せ入力モード

可

CHECKBOX_CHECKEDをコールする前に、GET_ITEM_PROPERTY(*item_name*, ITEM_TYPE)へのコールを使用して、項目タイプを確認できます。

プログラムでチェックボックスの値を設定するには、標準バインド変数構文を使用して、チェックボックスに有効な値を割り当てます。

パラメータ

item_id	項目作成時にForm Builderによってその項目に割り当てられた一意のIDを指定します。このIDのデータ型はITEMです。
item_name	ユーザーが設計時に項目の名前として定義した文字列を指定します。その名前のデータ型はVARCHAR2です。

CHECKBOX_CHECKEDの制限事項

CHECKBOX_CHECKEDビルトインは、指定されたチェックボックスの状態に関するBOOLEAN値を戻します。CHECKBOX_CHECKEDビルトインは、チェックボックスの実際の値も、「その他の値のマッピング」プロパティに対して指定された値も戻しません。

CHECKBOX_CHECKEDの例

```
/*
** Built-in:  CHECKBOX_CHECKED
** Example:   Sets the query case-sensitivity of the item
**            whose name is passed as an argument, depending
**            on an indicator checkbox item.
*/
PROCEDURE Set_Case_Sensitivity( it_name VARCHAR2) IS
  indicator_name VARCHAR2(80) := 'control.case_indicator';
  it_id         Item;
BEGIN
  it_id := Find_Item(it_name);

  IF Checkbox_Checked(indicator_name) THEN
    /*
    ** Set the item whose name was passed in to query case-
    ** sensitively (i.e., Case Insensitive is False)
    */
    Set_Item_Property(it_id, CASE_INSENSITIVE_QUERY,
                   PROPERTY_FALSE );
ELSE
    /*
    ** Set the item whose name was passed in to query case-
    ** insensitively (ie Case Insensitive True)
    */
    Set_Item_Property(it_id,CASE_INSENSITIVE_QUERY,PROPERTY_TRUE);
  END IF;
END;
```

CHECK_RECORD_UNIQUENESSビルトイン

説明

On-Check-Uniqueトリガーからコールされると、レコードの主キーの一意性をチェックするデフォルトのForm Builder処理が開始されます。

このビルトインは主に、Oracle以外のデータ・ソースに対して実行されるアプリケーション用に組み込まれています。

構文

```
PROCEDURE CHECK_RECORD_UNIQUENESS;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

CHECK_RECORD_UNIQUENESSの制限事項

On-Check-Uniqueトリガー内でのみ有効です。

CHECK_RECORD_UNIQUENESSの例

```
/*
** Built-in:  CHECK_RECORD_UNIQUENESS
** Example:   Perform Form Builder record uniqueness checking
**            from the fields in the block that are marked as
**            primary keys based on a global flag setup at
**            startup by the form, perhaps based on a
**            parameter.
** Trigger:   On-Check-Unique
*/
BEGIN
  /*
  ** Check the global flag we set during form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    User_Exit('chkuniq block=EMP');
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Check_Record_Uniqueness;
  END IF;
END;
```

CLEAR_BLOCKビルトイン

説明

現ブロックからすべてのレコードを削除、つまり、"フラッシュ"します。

構文

```
PROCEDURE CLEAR_BLOCK;
PROCEDURE CLEAR_BLOCK
  (commit_mode  NUMBER);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

ポスト転記もコミットもされていない現ブロック内のレコードに対してエンド・ユーザーが変更を加えた場合、Form Builderはcommit_modeパラメータに入力された引数によって示された指示に従ってそのレコードを処理します。

commit_mode	**オプションのアクション・パラメータは、次の定数を引数としてとることができます。**
	ASK_COMMIT — **CLEAR_BLOCK処理の中で、Form Builderからエンド・ユーザーに対し、コミットを行うよう指示が出されます。**
	DO_COMMIT — **Form Builderにより、変更点の有効化およびコミット処理が行われ、現行ブロックがフラッシュされます。この一連の処理の中で、エンド・ユーザーへの指示はありません。**
	NO_COMMIT — **Form Builderにより、変更点が有効化され、現行ブロックがフラッシュされます。この一連の処理の中で、変更点のコミットは行われず、エンド・ユーザーへの指示もありません。**
	NO_VALIDATE — **Form Builderでは、現ブロックをフラッシュします。このとき、変更点の有効化やコミット処理は行われず、エンド・ユーザーへの指示もありません。**

CLEAR_BLOCKの例

```
/*
** Built-in:  CLEAR_BLOCK
** Example:   Clears the current block without validation, and
**            deposits the primary key value which the user
**            has typed into a global variable which a
```

```
**            Pre-Query trigger will use to include it as a
**            query criterion.
** Trigger:   When-New-Item-Instance
*/
BEGIN
  IF :Emp.Empno IS NOT NULL THEN
    :Global.Employee_Id := :Emp.Empno;
    Clear_Block(No_Validate);
  END IF;
END;
/*
** Trigger:Pre-Query
*/
BEGIN
  Default_Value(NULL, 'Global.Employee_Id');
  IF :Global.Employee_Id IS NOT NULL THEN
    :Emp.Empno := :Global.Employee_Id;
  END IF;
END;
```

CLEAR_EOLビルトイン

説明

現在のカーソル位置から行の終わりまでの現行テキスト項目の値を消去します。

構文

```
PROCEDURE CLEAR_EOL;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

CLEAR_EOLの例

```
/*
** Built-in:  CLEAR_EOL
** Example:   Clears out the contents of any number field when
**            the end user navigates to it.
```

```
** Trigger:   When-New-Item-Instance
*/
BEGIN
  IF Get_Item_Property(:System.Trigger_Item, DATATYPE) = 'NUMBER' THEN
     Clear_Eol;
  END IF;
END;
```

CLEAR_FORMビルトイン

説明

現行のフォームのすべてのレコードを削除、つまりフラッシュして、最初のブロックの最初の項目に入力フォーカスを置きます。

構文

```
POROCEDURE CLEAR_FORM;
PROCEDURE CLEAR_FORM
  (commit_mode  NUMBER);
PROCEDURE CLEAR_FORM
  (commit_mode    NUMBER,
   rollback_mode  NUMBER);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

ポストもコミットもされていない現行のフォームまたは任意のコール先フォーム内のレコードに対してエンド・ユーザーが変更を加えた場合、Form Builderは次のパラメータに入力された引数によって示された指示に従ってそのレコードを処理します。

commit_mode	ASK_COMMIT	LEAR_FORM処理の中で、Form Builderからエンド・ユーザーに対し、コミットを行うよう指示が出されます。
	DO_COMMIT	Form Builderにより、変更点の有効化およびコミット処理が行われ、現行フォームがフラッシュされます。この一連の処理の中で、エンド・ユーザーへの指示はありません。
	NO_COMMIT	Form Builderにより、変更点が有効化され、現行フォームがフラッシュされます。この一連の処理の中で、変更点のコミットは行われず、エンド・ユーザーへの指示もありません。
	NO_VALIDATE	Form Builderでは、現フォームをフラッシュします。このとき、変更点の有効化やコミット処理は行われず、エンド・ユーザーへの指示もありません。
rollback_mode	TO_SAVEPOINT	Form Builderでは（ポスト済みのものを含む）コミットされていないすべての変更内容が現行フォームのセーブポイントにロールバックされます。
	FULL_ROLLBACK	Form Builderでは、現ランフォーム・セッション中に加えられたコミットされていないすべての変更内容（ポスト済みのものを含む）がロールバックされます。ポスト専用モードで実行されているフォームからFULL_ROLLBACKを指定することはできません。（フォームにポストしていないのレコードがあるときに別のフォームをコールすると、ポスト専用モードになる可能性があります。コール側フォームが発行したロックの損失を防ぐために、Form Builderはコール先フォーム内のコミット処理をすべて妨げます。

CLEAR_FORMの制限事項

非同期ブロックまたはユーザー定義サブプログラムでPL/SQL ROLLBACK文を使用すると、Form Builderはその文をパラメータのないCLEAR_FORMビルトイン・サブプログラムとして解釈します。

CLEAR_FORMの例

```
/*
** Built-in:  CLEAR_FORM
** Example:   Clear any changes made in the current form,
**            without prompting to commit.
*/
BEGIN
  Clear_Form(No_Validate);
END;
```

CLEAR_ITEMビルトイン

説明

現在のカーソル位置に関係なく現行のテキスト項目から値を消去して、テキスト項目値をNULLに変更します。

構文

```
PROCEDURE CLEAR_ITEM;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

CLEAR_ITEMの例

```
/*
** Built-in:  CLEAR_ITEM
** Example:   Clear the current item if it does not represent
**            the first day of a month.
** Trigger:When-New-Item-Instance
*/
BEGIN
  IF TO_CHAR(:Emp.Hiredate,'DD') <> '01' THEN
    Clear_Item;
    Message('This date must be of the form 01-MON-YY');
  END IF;
END;
```

CLEAR_LISTビルトイン

説明

リスト項目からすべての要素を消去します。Form Builderがリストを消去すると、項目の「必須」プロパティに関係なく、そのリストには1つの要素のみ（NULL要素）が含まれるようになります。

構文

```
PROCEDURE CLEAR_LIST
  (list_id  ITEM);
```

```
PROCEDURE CLEAR_LIST
  (list_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

list_id	Form Builderがリスト項目を作成するときに割り当てる一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
list_name	ユーザーがリスト項目を作成するときに付けた名前。その名前のデータ型はVARCHAR2です。

使用上の注意

■ 「ほかの値のマッピング」プロパティが定義されていて、ブロック中に問合せレコードがある場合は、CLEAR_LISTのビルトイン処理を実行しないでください。このような場合にCLEAR_LISTビルトインを使用すると、すでにフェッチされたレコードを表示できなくなる場合があります。

たとえば、値A、BおよびCを含んでいるリスト項目があり、「ほかの値のマッピング」プロパティが定義されているとします。また、これらの値はデータベースからフェッチされたものであるとします（問合せはオープンしている）。この時点でCLEAR_LISTが指定されたリストを消去すると、それまでにフェッチされた値（A、BおよびC）を表示する試みがForm Builderによってなされても、リストが消去されたために表示できず、エラーが発生します。

リストを消去する前に、オープンされている問合せをクローズします。オープンされた問合せのクローズには、ABORT_QUERYビルトインを使用します。

注意: 問合せされたレコードがブロックに含まれている場合、ブロック・ステータスはQUERYです。挿入または更新されたレコードがブロックに含まれている（問合せされたレコードが修正された）場合、ブロック・ステータスはCHANGEDです。

CLEAR_LISTの制限事項

■ ポップ・リストまたはTLISTスタイルのリスト項目については、初期値要素または他の値要素を削除するために指定されたDELETE_LIST_ELEMENTの条件が満たされない場合、CLEAR_LISTはそれらの要素を消去しません。

初期値または他の値要素を削除できない場合、CLEAR_LISTはリスト内のこれらの要素をそのままにして、それ以外のすべての要素を消去します。詳細は、DELETE_LIST_ELEMENTの制限事項を参照してください。

CLEAR_LISTの例

```
/*
**  Built-in:  CLEAR_LIST
**  Example:See POPULATE_LIST
*/
```

CLEAR_MESSAGEビルトイン

説明

画面メッセージ領域から現行のメッセージを削除します。

構文

```
PROCEDURE CLEAR_MESSAGE;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

CLEAR_MESSAGEの例

```
/*
** Built-in:  CLEAR_MESSAGE
** Example:   Clear the message from the message line.
*/
BEGIN
  Message('Working...',No_Acknowledge);
  SELECT current_tax
    INTO :Emp.Tax_Rate
    FROM tax_table
   WHERE state_abbrev = :Emp.State;
  Clear_Message;
END;
```

CLEAR_RECORDビルトイン

説明

妥当性チェックなしでブロックから現行のレコードを削除、つまりフラッシュします。問合せがブロック内でオープンしていて、現行レコードの削除後にレコード・スペースが埋められていない場合、Form Builderは次のレコードをフェッチしてそのブロックを再び埋めます。

消去されたデータベース・レコードは、次の「トランザクションのポストおよびコミット」プロセスにより削除として処理されません。

デフォルトのマスター/ディテール・ブロック・リレーションでは、マスター・レコードを消去すると、妥当性チェックなしで、そのレコードに対応するディテール・レコードがすべて消去されます。

構文

```
PROCEDURE CLEAR_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

CLEAR_RECORDの例

```
/*
** Built-in:  CLEAR_RECORD
** Example:   Clear the current record if it's not the last
**            record in the block.
*/
BEGIN
  IF :System.Last_Record = 'TRUE' AND :System.Cursor_Record = '1' THEN
    Message('You cannot clear the only remaining entry.');
    Bell;
  ELSE
    Clear_Record;
  END IF;
END;
```

CLOSE_FORMビルトイン

説明

複数のフォームを持つアプリケーションで、指示されたフォームをクローズします。指示されたフォームが現行フォームである場合、CLOSE_FORMはEXIT_FORMと同等の意味を持ちます。

構文

```
PROCEDURE CLOSE_FORM
  (form_name  VARCHAR2);
PROCEDURE CLOSE_FORM
  (form_id  FORMMODULE);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

form_name	VARCHAR2としてクローズするフォームの名前を指定します。
form_id	実行時にインスタンス化されたときにフォームに動的に割り当てられる、一意のID。 FIND_FORMビルトインは、適切な型の変数に使用します。このフォームIDのデータ型はFORMMODULEです。

CLOSE_FORMの制限事項

- CALL_FORMを発行してコール先モーダル・フォームをコールすることにより現在使用不可にされているフォームは、クローズできません。
- コール先フォームがコール側のフォームをクローズすることはできません。たとえば、FORM_AがFORM_Bをコールした場合、FORM_BはFORM_Aをクローズすることはできません。

CLOSE_SERVERビルトイン

説明

OLEコンテナに関連付けらたOLEサーバーを非アクティブ化します。OLEサーバーとOLEコンテナの接続を終了します。

構文

```
PROCEDURE CLOSE_SERVER
  (item_id  Item);
PROCEDURE CLOSE_SERVER
  (item_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。

CLOSE_SERVERの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効。

CLOSE_SERVERの例

```
/*
** Built-in: CLOSE_SERVER
** Example:  Deactivates the OLE server associated with the object
**           in the OLE container.
** Trigger:  When-Button-Pressed
*/
DECLARE
 item_id  ITEM;
 item_name VARCHAR(25) := 'OLEITM';
BEGIN
 item_id := Find_Item(item_name);
 IF Id_Null(item_id) THEN
  message('No such item:'||item_name);
 ELSE
  Forms_OLE.Close_Server(item_id);
 END IF;
END;
```

COMMIT_FORMビルトイン

説明

データベース内のデータを更新して、フォーム内のデータと一致させます。Form Builderは最初に、フォームの妥当性チェックを行ってから、フォーム内の各ブロックごとにデータベースに対して削除、挿入および更新を行い、データベースをコミットします。データベースがコミットされることにより、データベースのすべての行および表のロックが解除されます。

エンド・ユーザーが現行のRunformセッション中にデータベースにデータを転記した場合、COMMIT_FORMビルトインへのコールによってこのデータがデータベースに対してコミットされます。

コミット操作に続いて、Form Builderはすべての実表ブロック内のすべてのレコードをデータベースからの問合せレコードのように扱います。Form Builderは、コミット処理時に起動するトリガー内に発生した変更を認識しません。

構文

```
PROCEDURE COMMIT_FORM;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

COMMIT_FORMの制限事項

PL/SQL COMMIT文を非同期ブロックまたはフォーム・レベル・プロシージャで使用する場合、Form BuilderはCOMMIT_FORMビルトインへのコールとして解釈します。

COMMIT_FORMの例

例1

```
/*
** Built-in:  COMMIT_FORM
** Example:   If there are records in the form to be
**            committed, then do so.Raise an error if the
**            commit was not successful.
*/
BEGIN
  /*
```

```
  ** Force validation to happen first
  */
  Enter;
  IF NOT Form_Success THEN
    RAISE Form_Trigger_Failure;
  END IF;
  /*
  ** Commit if anything is changed
  */
  IF :System.Form_Status = 'CHANGED' THEN
    Commit_Form;
    /*
    ** A successful commit operation sets Form_Status back
    ** to 'QUERY'.
    */
    IF :System.Form_Status <> 'QUERY' THEN
      Message('An error prevented your changes from being
              committed.');
      Bell;
      RAISE Form_Trigger_Failure;
    END IF;
  END IF;
END;
```

例2

```
/*
** Built-in:  COMMIT_FORM
** Example:    Perform Form Builder database commit during commit
**             processing.Decide whether to use this Built-in
**             or a user exit based on a global flag setup at
**             startup by the form, perhaps based on a
**
** Trigger:    On-Commit
*/
BEGIN
  /*
  ** Check the global flag we set during form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    User_Exit('my_commit');
  /*
  ** Otherwise, do the right thing.
  */
```

```
  ELSE
    Commit_Form;
  END IF;
END;
```

CONVERT_OTHER_VALUEビルトイン

説明

チェックボックス、ラジオ・グループまたはリスト項目の現行値を、現行のチェックボックスの状態（「チェックあり」または「チェックなし」）または現行のラジオ・グループ要素またはリスト項目要素に関連付けられた値に変換します。

構文

```
PROCEDURE CONVERT_OTHER_VALUE
  (item_id  ITEM);
PROCEDURE CONVERT_OTHER_VALUE
  (item_name  VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

item_id	Form Builderが項目を作成するときに割り当てる一意のIDを指定します。このIDのデータ型はITEMです。
item_name	設計時に項目の名前として定義したVARCHAR2文字列を指定します。

CONVERT_OTHER_VALUEの制限事項

項目がチェックボックス、ラジオ・グループおよびリスト項目のいずれでもない場合、「FRM-41026: 項目が操作を認識しません」というエラー・メッセージが表示されます。このエラーを回避するには、CONVERT_OTHER_VALUE.をコールする前にGET_ITEM_PROPERTY（item_name, ITEM_TYPE）のコールを発行して項目タイプを判別します。

CONVERT_OTHER_VALUEの例

```
/*
** Built-in:  CONVERT_OTHER_VALUE
** Example:   Ensure that a particular checkbox's value
**            represents either the checked or unchecked
```

```
**            value before updating the record.
** Trigger:   Pre-Update
*/
BEGIN
  Convert_Other_Value('Emp.Marital_Status');
END;
```

COPYビルトイン

説明

項目または変数から別の項目またはグローバル変数に値をコピーします。NAME_INビルトインにより参照される項目に値を書き込むために特別に使用します。COPYは次の2つの理由で存在します。

- 参照項目を値に等しく設定する場合、標準PL/SQL構文は使用できません。
- フォームが問合せ入力モードのときに、関係演算子などの文字を「NUMBER」フィールドや「DATE」フィールドにプログラムによって書き込むことが必要となる場合があります。

構文

```
PROCEDURE COPY
  (source      VARCHAR2,
   destination  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

source	***source*はリテラル値です。**
destination	***destination* は、テキスト項目または別のグローバル変数のいずれかです。**

使用上の注意

- DATE_FORMAT_COMPATIBILITY_MODE propertyが5.0に設定されている場合は、日付値を指定してCOPYを使用すると、BUILTIN_DATE_FORMAT propertyで定義された値が使用されます。このプロパティが4.5に設定されている場合、COPYではデフォルトの米国書式を使用して日付文字列を設定します。

■ 参照元としてテキスト項目を使用するには、次のコードを使用します。

```
COPY(NAME_IN(source), destination);
```

COPYの制限事項

テキスト項目にコピーされた値については妥当性チェックは行われません。ただし、それ以外のタイプの項目については、コピーされた値に関してもすべて標準の妥当性チェックが行われます。

COPYの例

例1

```
/*
** Built-in:  COPY
** Example:   Force a wildcard search on the EmpNo item during
**            query.
** Trigger:   Pre-Query
*/
DECLARE
  cur_val VARCHAR2(40);
BEGIN
  /*
  ** Get the value of EMP.EMPNO as a string
  */
  cur_val := Name_In('Emp.Empno');
  /*
  ** Add a percent to the end of the string.
  */
  cur_val := cur_val || '%';
  /*
  ** Copy the new value back into the item so Form Builder
  ** will use it as a query criterion.
  */
  Copy( cur_val, 'Emp.Empno' );
END;
```

例2

```
/*
** Built-in:  COPY
** Example:   Set the value of a global variable whose name is
**           dynamically constructed.
*/
DECLARE
```

```
  global_var_name  VARCHAR2(80);
BEGIN
  IF :Selection.Choice = 5 THEN
    global_var_name := 'Storage_1';
  ELSE
    global_var_name := 'Storage_2';
  END IF;
  /*
  ** Use the name in the 'global_var_name' variable as the
  ** name of the global variable in which to copy the
  ** current 'Yes' value.
  */
  COPY( 'Yes', 'GLOBAL.'||global_var_name );
END;
```

COPY_REGIONビルトイン

説明

画面からテキスト項目またはイメージ項目の選択した領域をコピーし、別の選択領域をカットまたはコピーするまで、その領域をペースト・バッファに格納します。

構文

```
PROCEDURE COPY_REGION;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

使用上の注意

COPY_REGIONおよびその他の編集機能は、テキスト項目やイメージ項目に対してのみ使用してください。カット・アンド・コピー機能では選択済み領域はシステム・クリップボードに転送され、ペースト・ターゲットが指定されるまでシステム・クリップボード内にあります。ペー

スト・ターゲットが指定されると、カットまたはコピーされた内容がターゲット位置にペーストされます。

COPY_REPORT_OBJECT_OUTPUTビルトイン

説明

レポートの出力をファイルにコピーします。

構文

```
PROCEDURE COPY_REPORT_OBJECT_OUTPUT
  (report_id VARCHAR2(20),
   output_file VARCHAR2
);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

report_id	RUN_REPORT_OBJECTビルトインから戻されるVARCHAR2値。この値によって、現在ローカルで実行されているまたはリモートのレポート・サーバー上で実行されているレポートが一意に識別されます。
output_file	レポート出力のコピー先ファイルの名前。

使用上の注意

- 「レポート宛先タイプ」プロパティを使用して、出力ファイルの形式を指定します。
- リモート・マシンからレポートの出力をコピーするには、「レポート宛先タイプ」プロパティを「Cache」に設定する必要があります。

COPY_REPORT_OBJECT_OUTPUTの例

```
DECLARE
   repid REPORT_OBJECT;
   v_rep  VARCHAR2(100);
   rep_status varchar2(20);
BEGIN
   repid := find_report_object('report4');
   v_rep := RUN_REPORT_OBJECT(repid);
```

```
    rep_status := report_object_status(v_rep);

    if rep_status = 'FINISHED' then
       message('Report Completed');
       copy_report_object_output(v_rep,'d:¥temp¥local.pdf');
       host('netscape d:¥temp¥local.pdf');
    else
       message('Error when running report.');
    end if;
END;
```

COUNT_QUERYビルトイン

説明

On-Countトリガーで、現ブロックに関して問合せで取り出す行数を識別するためのデフォルトのForm Builder処理を行って、現ブロックを消去します。ブロック内にコミットすべき変更がある場合、Form BuilderはCOUNT_QUERY処理時にエンド・ユーザーに対してその変更をコミットするよう要求します。COUNT_QUERYに対して有効なコールが行われると、Form Builderは、次のメッセージを戻します。

FRM-40355:問合せによって<レコード件数>件のレコードが取り出されます。

このビルトインは主に、Oracle以外のデータ・ソースに対して実行されるアプリケーション用に組み込まれているものです。

構文

```
PROCEDURE COUNT_QUERY;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

COUNT_QUERYの制限事項

制限付きビルトインを許可するトリガーでのみ有効です。

COUNT_QUERYの例

例1

```
/*
** Built-in:  COUNT_QUERY
** Example:   Display the number of records that will be retrieved
**            by the current query.
*/
BEGIN
  Count_Query;
END;
```

例2

```
/*
** Built-in:  COUNT_QUERY
** Example:   Perform Form Builder count query hits processing.
**            Decide whether to use this Built-in or a user
**            exit based on a global flag setup at startup by
**            the form, perhaps based on a parameter.
** Trigger:   On-Count
*/
BEGIN
  /*
  ** Check the global flag we set during form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    /*
    ** User exit returns query hits count back into the
    ** CONTROL.HITS item.
    */
    User_Exit('my_count');
    /*
    ** Deposit the number of query hits in the appropriate
    ** block property so Form Builder can display its normal
    ** status message.
    */
    Set_Block_Property(:System.Trigger_Block,QUERY_HITS,
                  :control.hits);
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Count_Query;
```

```
  END IF;
END;
```

CREATE_GROUPビルトイン

説明

指定の名前を持つ非問合せレコード・グループを作成します。ADD_GROUP_COLUMNビルトイン、ADD_GROUP_ROWビルトインおよびPOPULATE_GROUP_WITH_QUERYビルトインを使用して明示的に行や列を追加するまでは、新規に作成されたレコードには列も行もありません。

構文

```
FUNCTION CREATE_GROUP
  (recordgroup_name  VARCHAR2,
   scope             NUMBER,
   array_fetch_size  NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

RECORDGROUP

問合せ入力モード

可

パラメータ

recordgroup_name	ユーザーが設計時にレコード・グループの名前として定義した文字列。Form Builderは、レコード・グループ・オブジェクトを作成するときに、オブジェクトに型RECORDGROUPの一意のIDも割り当てます。レコード・グループまたはレコード・グループ列ビルトイン・サブプログラムの後続のコールにおいて、名前またはIDでレコード・グループをコールすることができます。

scope	レコード・グループが現行のフォーム内のみで使用できるか、または複数のアプリケーション内のすべてのフォーム内で使用できるかを指定します。引数として次の定数をとります。 FORM_SCOPE　レコード・グループが現行のフォーム内のみで使用できることを示します。これがデフォルト値です。 GLOBAL_SCOPE　レコード・グループがグローバルであり、アプリケーション内のすべてのフォームで使用できることを示します。グローバル・レコード・グループはいったん作成されると、ランタイム・セッションが終わるまで持続します。
array_fetch_size	配列フェッチ・サイズを指定します。デフォルトの配列サイズは20です。

CREATE_GROUPの例

```
/*
** Built-in:  CREATE_GROUP
** Example:   Creates a record group and populates its values
**            from a query.
*/
DECLARE
  rg_name  VARCHAR2(40) := 'Salary_Range';
  rg_id    RecordGroup;
  gc_id    GroupColumn;
  errcode  NUMBER;
BEGIN
  /*
  ** Make sure the record group does not already exist.
  */
  rg_id := Find_Group(rg_name);
  /*
  ** If it does not exist, create it and add the two
  ** necessary columns to it.
  */
  IF Id_Null(rg_id) THEN
    rg_id := Create_Group(rg_name);
    /* Add two number columns to the record group */
    gc_id := Add_Group_Column(rg_id, 'Base_Sal_Range',
                              NUMBER_COLUMN);
    gc_id := Add_Group_Column(rg_id, 'Emps_In_Range',
                              NUMBER_COLUMN);
  END IF;
  /*
  ** Populate group with a query
  */
  errcode := Populate_Group_With_Query( rg_id,
```

```
                  'SELECT SAL-MOD(SAL,1000),COUNT(EMPNO) '
                ||'FROM EMP '
                ||'GROUP BY SAL-MOD(SAL,1000) '
                ||'ORDER BY 1');
END;
```

CREATE_GROUP_FROM_QUERYビルトイン

説明

レコード・グループを所定の名前で作成します。レコード・グループには、ユーザーが問合せの選択リストに入れる各列を表す列があります。POPULATE_GROUPビルトインによりレコード・グループに行を追加します。

注意: SELECT文の列に仮列名（別名）を渡さないと、Form Builderはダミー・カウンタ<NUM>でICRGGQを作成します。これは、列名が無効である場合には必ず行われます。最初のダミー名カウンタは常に1です。たとえば、問合せSELECT 1 + 1 FROM DUALの結果は、ICRGGQ_1とい名前の列になります。

構文

```
FUNCTION CREATE_GROUP_FROM_QUERY
  (recordgroup_name  VARCHAR2,
   query             VARCHAR2,
   scope             NUMBER,
   array_fetch_size  NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

RECORDGROUP

問合せ入力モード

可

パラメータ

ecordgroup_name	**レコード・グループの名前。Form Builderは、レコード・グループ・オブジェクトを作成するときに、オブジェクトに型RECORDGROUPの一意のIDも割り当てます。**

query	引用符（'）に囲まれた有効なSQL SELECT文。問合せの結果として取り出された列はすべて、表の中の列のデータ型をとります。問合せを表内の列のサブセットに制限すると、レコード・グループにはそれらの列のみが作成されます。
scope	レコード・グループが現行のフォーム内のみで使用できるか、または複数のアプリケーション内のすべてのフォーム内で使用できるかを指定します。引数として次の定数をとります。 FORM_SCOPE　レコード・グループが現行のフォーム内のみで使用できることを示します。これがデフォルト値です。 GLOBAL_SCOPE　レコード・グループがグローバルであり、アプリケーション内のすべてのフォームで使用できることを示します。グローバル・レコード・グループはいったん作成されると、ランタイム・セッションが終わるまで持続します。
array_fetch_size	配列フェッチ・サイズを指定します。デフォルトの配列サイズは20です。

CREATE_GROUP_FROM_QUERYの制限事項

- フォームAを実行中にグローバル・レコード・グループが問合せから作成され（または移入され）、フォームAが実行を終了するときにAに対してローカルなバインド変数参照（:block.itemまたは:PARAMETER.param）が問合せ文字列に含まれている場合、グローバル問合せレコード・グループはグローバル非問合せレコード・グループに変換されます（データは保持しますが、POPULATE_GROUPに対する以降のコールはエラーとみなされます）。

CREATE_GROUP_FROM_QUERYの例

```
/*
** Built-in:  CREATE_GROUP_FROM_QUERY
** Example:   Create a record group from a query, and populate it.
*/
DECLARE
  rg_name  VARCHAR2(40) := 'Salary_Range';
  rg_id    RecordGroup;
  errcode  NUMBER;
BEGIN
  /*
  ** Make sure group doesn't already exist
  */
  rg_id := Find_Group( rg_name );
  /*
  ** If it does not exist, create it and add the two
  ** necessary columns to it.
  */
  IF Id_Null(rg_id) THEN
    rg_id := Create_Group_From_Query( rg_name,
               'SELECT SAL-MOD(SAL,1000) BASE_SAL_RANGE,'
              ||'COUNT(EMPNO) EMPS_IN_RANGE '
```

```
            ||'FROM EMP '
            ||'GROUP BY SAL-MOD(SAL,1000) '
            ||'ORDER BY 1');
  END IF;
  /*
  ** Populate the record group
  */
  errcode := Populate_Group( rg_id );
END;
```

CREATE_OLEOBJビルトイン

説明

最初のフォームで、OLEオブジェクトを作成し、そのオブジェクトの永続性を確立します。2番目のフォームで、前にインスタンス生成されたOLEオブジェクトの永続性を変更します。

構文

```
FUNCTION CREATE_OLEOBJ
  (name OLEOBJ, persistence_boolean := TRUE)
RETURN objpointer OLEOBJ;
```

...または...

```
FUNCTION CREATE_OLEOBJ
  (localobject VARCHAR2,
   persistence_boolean := TRUE)
RETURN objpointer OLEOBJ;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

OLEオブジェクトのポインタ

パラメータ

name	OLEオブジェクトのサーバーのプログラムID。
localobject	ステータスを非永続から永続に変更するOLEオブジェクトへのポインタ。

persistence_boolean	ブール値TRUEによって、オブジェクトが永続として設定されます。これはオプションのパラメータです。指定されない場合のデフォルト値は非永続です。

使用上の注意

永続オブジェクトはトリガー起動中は存在します。非永続オブジェクトは、コールを起動したトリガーが実行されている間のみ存在します。

CREATE_PARAMETER_LISTビルトイン

説明

パラメータ・リストを所定の名前で作成します。パラメータ・リストを作成した時点ではその中にパラメータは含まれていません。ADD_PARAMETERビルトイン・サブプログラムを使用してリストにパラメータを追加する必要があります。パラメータ・リストは、引数としてCALL_FORM、NEW_FORM、OPEN_FORMおよびRUN_PRODUCTビルトイン・サブプログラムに渡すことができます。

構文

```
FUNCTION CREATE_PARAMETER_LIST
  (name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

ParamList

問合せ入力モード

可

パラメータ

name	パラメータ・リスト・オブジェクトのVARCHAR2名を指定します。Form Builderは、オブジェクトを生成するときに、そのオブジェクトにPARAMLIST型の一意のIDを割り当てます。パラメータ・リストに関係するビルトイン・サブプログラムへの後続のコールにおいて、名前またはIDでパラメータ・リストをコールすることができます。

CREATE_PARAMETER_LISTの制限事項

■ DEFAULTという名前のパラメータ・リストを作成することはできません。DEFAULTは、Form

Builderがランタイム・セッションの開始時に作成するパラメータ・リストのために保持されます。

- すでにパラメータ・リストが存在する場合は、パラメータ・リストは作成できません。作成しようとするとエラーが発生します。このエラーを回避するには、パラメータ・リストを作成する前に、ID_NULLを使用してパラメータ・リストがすでに存在しているかどうかをチェックします。パラメータ・リストがすでに存在する場合は、そのパラメータ・リストを削除してから、新しいリストを作成してください。

CREATE_PARAMETER_LISTの例

```
/*
** Built-in:  CREATE_PARAMETER_LIST
** Example:   Create a parameter list named 'TEMPDATA'.First
**            make sure the list does not already exist, then
**            attempt to create a new list.Signal an error
**            if the list already exists or if creating the
**            list fails.
*/
DECLARE
  pl_id   ParamList;
  pl_name VARCHAR2(10) := 'tempdata';
BEGIN
  pl_id := Get_Parameter_List(pl_name);
  IF Id_Null(pl_id) THEN
    pl_id := Create_Parameter_List(pl_name);
    IF Id_Null(pl_id) THEN
      Message('Error creating parameter list '||pl_name);
      RAISE Form_Trigger_Failure;
    END IF;
  ELSE
    Message('Parameter list '||pl_name||' already exists!');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

CREATE_QUERIED_RECORDビルトイン

説明

On-Fetchトリガーからコールされるときに、そのブロックの待ちリスト上にレコードを作成します。待ちリストとは、中間的なレコード・バッファであり、この中にはデータ・ソースからフェッチされたが、まだアクティブ・レコードのブロック・リストには入れられていないレコー

ドが含まれています。このビルトインは主に、トランザクション・トリガーを使用してOracle以外のデータ・ソースに対して実行されるアプリケーション用に組み込まれているものです。

待ちリストからレコードを削除する方法はないので注意してください。したがって、このようなアプリケーションではプログラムによってレコードに移入するために利用できるデータを確保しておく必要があります。

構文

```
PROCEDURE CREATE_QUERIED_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

CREATE_QUERIED_RECORDの制限事項

- 多数のレコードを持つブロックでは、このプロシージャによってディスクI/Oまたはメモリーの割当て、あるいはその両方に副作用がもたらされる可能性があります。

CREATE_QUERIED_RECORDの例

```
/*
** Built-in:  CREATE_QUERIED_RECORD
** Example:   Fetch the next N records into this block.Record
**            count kept in Global.Record_Count.
** Trigger:   On-Fetch
*/
DECLARE
  fetch_count NUMBER;
  FUNCTION The_Next_Seq
  RETURN NUMBER IS
    CURSOR next_seq IS SELECT uniq_seq.NEXTVAL FROM DUAL;
    tmp NUMBER;
  BEGIN
    OPEN next_seq;
    FETCH next_seq INTO tmp;
    CLOSE next_seq;
    RETURN tmp;
```

```
  END;
BEGIN
  /*
  ** Determine how many records Form Builder is expecting us to
  ** fetch
  */
  fetch_count := Get_Block_Property('MYBLOCK',RECORDS_TO_FETCH);
  FOR i IN 1..fetch_count LOOP
    /*
    ** Create the Queried Record into which we'll deposit
    ** the values we're about to fetch;
    */
    Create_Queried_Record;
    :Global.Record_Count := NVL(:Global.Record_Count,0)+1;
    /*
    ** Populate the item in the queried record with a
    ** sequence function we declared above
    */
   :myblock.numbercol := the_next_seq;
  END LOOP;
END;
```

CREATE_RECORDビルトイン

説明

現ブロック内の現行のレコードの後に新規のレコードを作成します。次に、Form Builderは新規のレコードに移動します。

構文

```
PROCEDURE CREATE_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

CREATE_RECORDの例

```
/*
** Built-in:  CREATE_RECORD
** Example:   Populate new records in a block based on return
**            values from a query
*/
PROCEDURE Populate_Rows_Into_Block( projid NUMBER)  IS
  CURSOR tempcur( cp_projid NUMBER ) IS
    SELECT milestone_name, due_date
      FROM milestone
     WHERE project_id = cp_projid
    ORDER BY due_date;
BEGIN
  /* Add these records to the bottom of the block */
  Last_Record;
  /* Loop thru the records in the cursor */
  FOR rec IN tempcur( projid ) LOOP
    /*
    ** Create an empty record and set the current row's
    ** Milestone_Name and Due_Date items.
    */
    Create_Record;
  : Milestone.Milestone_Name := rec.milestone_name;
  : Milestone.Due_Date       := rec.due_date;
  END LOOP;
  First_Record;
END;
```

CREATE_TIMERビルトイン

説明

新しいタイマーを所定の名前で作成します。間隔およびタイマーを時間切れのたびに繰り返すか、一回のみ実行するかを指示できます。タイマーが時間切れになると、Form BuilderはWhen-Timer-Expiredトリガーを起動します。

構文

```
FUNCTION CREATE_TIMER
  (timer_name    VARCHAR2,
   milliseconds  NUMBER,
   iterate       NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

timer

問合せ入力モード

可

パラメータ

timer_name	最高30文字までの英数字でタイマー名を指定します。名前はアルファベット文字で始める必要があります。その名前のデータ型はVARCHAR2です。
milliseconds	タイマーの持続時間をミリ秒単位で指定します。このパラメータに指定できる値の範囲は1～2147483648ミリ秒です。2147483648を超える値は、2147483648に切り捨てられます。指定できるのは正の整数のみであることに注意してください。このパラメータのデータ型はNUMBER型です。詳細は、制限事項を参照してください。
iterate	時間切れの時にタイマーを繰り返すかどうかを指定します。引数として次の定数をとります。 REPEAT 時間切れの時にタイマーを繰り返すことを示します。これがデフォルトです。 NO_REPEAT 再び明示的にコールされるまで、時間切れの時にタイマーを繰り返さず、1回のみ使用することを示します。

CREATE_TIMERの制限事項

- 2147483648を超える値は、2147483648に統一されます。
- ミリ秒を10進値で表すことはできません。
- 大文字か小文字かに関係なく、2つのタイマーが同じフォーム・インスタンスで同じ名前を共有することはできません。
- タイマーの実行時に定義されたWhen-Timer-Expiredトリガーがない場合は、エラーが戻されます。
- タイマーの実行時に定義されたWhen-Timer-Expiredトリガーがなく、タイマーが繰返しタイマーの場合、その後の繰返しは取り消されますが、タイマーは保持されます。
- タイマーの実行時にWhen-Timer-Expiredトリガーが定義されておらず、繰返しタイマーでない場合、そのタイマーは削除されます。

CREATE_TIMERの例

次の例では、EMP_TIMERという名前のタイマーを作成し、そのタイマーを60秒にセットし、反復値をNO_REPEATに設定しています。

```
DECLARE
    timer_id Timer;
    one_minute NUMBER(5) := 60000;
BEGIN
    timer_id := CREATE_TIMER('emp_timer', one_minute, NO_REPEAT);
END;
```

CREATE_VARビルトイン

説明

空の、名前のないバリアントを作成します。

スカラー用と配列用に2種類のバージョンのファンクションがあります。

構文

```
FUNCTION CREATE_VAR
  (persistence BOOLEAN)
RETURN newvar OLEVAR;
```

```
FUNCTION CREATE_VAR
  (bounds OLE_SAFEARRAYBOUNDS,
   vtype VT_TYPE,
   persistence BOOLEAN)
RETURN newvar OLEVAR;
```

ビルトイン・タイプ

制限なしファンクション

戻し値

作成されたOLEバリアント

パラメータ

persistence	作成後のバリアントの永続性を制御します。ブール値がTRUEの場合、永続としてバリアントが作成されます。値がFALSEの場合、非永続してバリアントが作成されます。 これはオプションのパラメータです。指定されない場合のデフォルト値は非永続です。
bounds	作成された配列に指定されるディメンションを指定するPL/SQL表。 このパラメータの内容およびレイアウトならびに型OLE_SAFEARRAYBOUNDSの詳細は、OLE用配列型のサポートを参照してください。
vtype	作成された配列内の要素のOLEバリアント・タイプ（VT_TYPE）。配列内に異なる要素タイプが混在している場合、VT_VARIANTを指定します。

使用上の注意

- 作成されたバリアントは、配列でない場合は、型指定されていません。その場合、その要素はユーザーが指定する型を持ちます。
- 作成されたバリアントには値もありません。SET_VARファンクションを使用して、初期値と型をバリアントに割り当てます。
- 永続バリアントはトリガー起動中は存在します。非永続バリアントは、コールを起動したトリガーが実行されている間のみ存在します。DESTROY_VARIANTも参照してください。

CUT_REGIONビルトイン

説明

画面からテキスト項目またはイメージ項目の選択した領域を削除し、別の選択領域をカットまたはコピーするまで、その領域をペースト・バッファに格納します。

構文

```
PROCEDURE CUT_REGION;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

使用上の注意

CUT_REGIONおよびその他の編集機能は、テキスト項目およびイメージ項目に対してのみ使用してください。カット・アンド・コピー機能では選択済み領域はシステム・クリップボードに転送され、ペースト・ターゲットが指定されるまでシステム・クリップボード内にあります。ペースト･ターゲットが指定されると､カットまたはコピーされた内容がターゲット位置にペーストされます。

DBMS_ERROR_CODEビルトイン

説明

検出された最新のデータベース・エラーのエラー番号を戻します。

構文

```
FUNCTIONDBMS_ERROR_CODE;
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

パラメータ

なし

使用上の注意

再帰的エラーの場合、このビルトイン処理では、スタックの先頭にあるメッセージ・コードが戻されます。したがって、エラー・テキストを解析する際は、これ以降のメッセージ番号について解析してください。

DBMS_ERROR_CODEの例

```
/*
** Built-in:DBMS_ERROR_CODE,DBMS_ERROR_TEXT
** Example: Reword certain Form Builder error messages by
**          evaluating the DBMS error code that caused them
** Trigger: On-Error
*/
DECLARE
  errcode    NUMBER       := ERROR_CODE;
  dbmserrcode NUMBER;
  dbmserrtext VARCHAR2(200);
BEGIN
  IF errcode = 40508 THEN
    /*
    ** Form Builder had a problem INSERTing, so
```

```
    ** look at the Database error which
    ** caused the problem.
    */
    dbmserrcode := DBMS_ERROR_CODE;
    dbmserrtext := DBMS_ERROR_TEXT;

    IF dbmserrcode = -1438 THEN
      /*
      ** ORA-01438 is "value too large for column"
      */
      Message('Your number is too large.Try again.');
    ELSIF dbmserrcode = -1400 THEN
      /*
      ** ORA-01400 is "Mandatory column is NULL"
      */
      Message('You forgot to provide a value.Try again.');
    ELSE
      /*
      ** Printout a generic message with the database
      ** error string in it.
      */
      Message('Insert failed because of '||dbmserrtext);
    END IF;
  END IF;
END;
```

DBMS_ERROR_TEXTビルトイン

説明

データベース・エラーのメッセージ番号（ORA-01438など）とメッセージ・テキストを戻します。

構文

```
FUNCTION DBMS_ERROR_TEXT;
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

パラメータ

なし

使用上の注意

例外処理ルーチン中のデータベース・エラー・メッセージをテストする場合に、このファンクションを使用してください。

再帰的エラーの場合、DBMS_ERROR_TEXTでは、発生した順にすべてのエラーが戻されます。

DBMS_ERROR_TEXTの例

```
/*
** Built-in:DBMS_ERROR_CODE,DBMS_ERROR_TEXT
** Example: Reword certain Form Builder error messages by
**          evaluating the DBMS error code that caused them
** Trigger: On-Error
*/
DECLARE
  errcode    NUMBER       := ERROR_CODE;
  dbmserrcode NUMBER;
  dbmserrtext VARCHAR2(200);
BEGIN
  IF errcode = 40508 THEN
    /*
    ** Form Builder had a problem INSERTing, so
    ** look at the Database error which
    ** caused the problem.
    */
    dbmserrcode := DBMS_ERROR_CODE;
    dbmserrtext := DBMS_ERROR_TEXT;

    IF dbmserrcode = -1438 THEN
      /*
      ** ORA-01438 is "value too large for column"
      */
      Message('Your number is too large.Try again.');
    ELSIF dbmserrcode = -1400 THEN
      /*
      ** ORA-01400 is "Mandatory column is NULL"
```

```
      */
      Message('You forgot to provide a value.Try again.');
    ELSE
      /*
      ** Printout a generic message with the database
      ** error string in it.
      */
      Message('Insert failed because of '||dbmserrtext);
    END IF;
  END IF;
END;
```

DEBUG_MODEビルトイン

説明

メニューのデバック・モードをトグル式でオン・オフに切り替えます。メニューのデバック・モードがオンの状態で、メニュー項目のコマンドを実行すると、Form Builderからその処理に応じたメッセージが発行されます。

構文

```
PROCEDURE DEBUG_MODE;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

DEBUG_MODEの制限事項

DEBUG_MODEは、メニュー・モジュールに限り使用できます。このビルトイン処理では、フォームをデバッグ・モードにすることはできません。

DEFAULT_VALUEビルトイン

説明

指定された変数の現在の値がNULLならば、指定された値をその変数にコピーします。変数の現在の値がNULLでなければ、DEFAULT_VALUEは何もしません。したがって、テキスト項目についてこのビルトイン処理を実行すると、NULL項目に対してCOPYビルトイン処理を実行する場合と同様の処理が行われます。処理の対象となる変数が未定義のグローバル変数の場合は、Form Builderによりその変数が作成されます。

構文

```
PROCEDURE DEFAULT_VALUE
  (value_string  VARCHAR2,
   variable_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

value_string	有効なVARCHAR2型の文字列や変数、または有効な文字列をデータとして持つテキスト項目。
variable_name	有効な変数またはグローバル変数、テキスト項目の名前。variable_nameのデータ型はVARCHAR2です。このビルトインの引数として渡されるオブジェクトは、一重引用符で囲む必要があります。

DEFAULT_VALUEの制限事項

DEFAULT_VALUEのビルトイン処理は、「初期値」項目のプロパティを処理するものではありません。

DEFAULT_VALUEの例

```
/*
** Built-in:DEFAULT_VALUE
** Example: Make sure a Global variable is defined by
**          assigning some value to it with Default_Value
*/
BEGIN
  /*
  ** Default the value of GLOBAL.Command_Indicator if it is
```

```
  ** NULL or does not exist.
  */
  Default_Value('***','global.command_indicator');
  /*
  ** If the global variable equals the string we defaulted
  ** it to above, then it must have not existed before
  */
  IF :Global.Command_Indicator = '***' THEN
    Message('You must call this screen from the Main Menu');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

DELETE_GROUPビルトイン

説明

プログラム処理の中で自動的に作成されたレコード・グループを削除します。

構文

```
PROCEDURE DELETE_GROUP
  (recordgroup_id  RecordGroup);
PROCEDURE DELETE_GROUP
  (recordgroup_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

recordgroup_id	**Form Builderがグループを作成するときに割り当てる一意のID。そのIDのデータ型はRECORDGROUPです。**
recordgroup_name	**ユーザーがレコード・グループを作成するときに付けた名前。その名前のデータ型はVARCHAR2です。**

DELETE_GROUPの制限事項

このビルトイン処理では、設計時に作成されたレコード・グループを削除することはできません。

DELETE_GROUPの例

```
/*
** Built-in:DELETE_GROUP
** Example: Delete a programmatically created record group
*/
PROCEDURE Remove_Record_Group( rg_name VARCHAR2 ) IS
  rg_id RecordGroup;
BEGIN
  /*
  ** Make sure the Record Group exists before trying to
  ** delete it.
  */
  rg_id := Find_Group( rg_name );
  IF NOT Id_Null(rg_id) THEN
    Delete_Group( rg_id );
  END IF;
END;
```

DELETE_GROUP_ROWビルトイン

説明

指定されたレコード・グループの指定行またはすべての行を削除します。削除された行以降の行では、Form Builderによって自動的に行番号が繰り上げられます。行が削除されると、その行が格納されていたメモリーが空きます。この空きメモリーは、Form Builderで使用します。

ALL_ROWS定数を使用してグループの全行削除を選択すると、Form Builderによりすべての行が削除されますが、そのグループ自体は、DELETE_GROUPサブプログラムを実行しないかぎり削除されずに残ります。

個別の行削除を行うと、削除行の前後の行が連番になるように、後続の行番号が振り直されます。

構文

```
PROCEDURE DELETE_GROUP_ROW
  (recordgroup_id  RecordGroup,
   row_number      NUMBER);
PROCEDURE DELETE_GROUP_ROW
  (recordgroup_name  VARCHAR2,
   row_number        NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

recordgroup_id	Form Builderにより、グループ作成の際に各グループに一意に割り当てられるID。そのIDのデータ型はRECORDGROUPです。
recordgroup_name	レコード・グループを作成するときに指定したグループの名前。その名前のデータ型はVARCHAR2です。
row_number	レコード・グループから削除する行を指定します。行には1からnまでの番号が自動的に付けられています。row numberパラメータのデータ型はNUMBERです。 ALL_ROWS　Form Builderに全行を削除させるよう指定します。このとき、レコード・グループは削除されません。 ALL_ROWSは定数です。

DELETE_GROUP_ROWの制限事項

このビルトイン処理では、静的レコード・グループの行削除はできません。

DELETE_GROUP_ROWの例

```
/*
** Built-in:DELETE_GROUP_ROW
** Example: Delete certain number of records from the tail
**          of the specified record group.
*/
PROCEDURE Delete_Tail_Records( recs_to_del NUMBER,
                           rg_name VARCHAR2 ) IS
  rg_id    RecordGroup;
  rec_count NUMBER;
BEGIN
  /*
  ** Check to see if Record Group exists
  */
  rg_id := Find_Group( rg_name );
  /*
  ** Get a count of the records in the record group
  */
  rec_Count := Get_Group_Row_Count( rg_id );
  IF rec_Count < recs_to_del THEN
    Message('There are only '||TO_CHAR(rec_Count)||
```

```
        ' records in the group.');
    RAISE Form_Trigger_Failure;
  END IF;
  /*
  ** Loop thru and delete the last 'recs_to_del' records
  */
  FOR j IN 1..recs_to_del LOOP
    Delete_Group_Row( rg_id, rec_Count - j + 1 );
  END LOOP;
END;
```

DELETE_LIST_ELEMENTビルトイン

説明

リスト項目から、特定のリスト要素を削除します。

構文

```
PROCEDURE DELETE_LIST_ELEMENT
  (list_name   VARCHAR2,
   list_index  NUMBER);
PROCEDURE DELETE_LIST_ELEMENT
  (list_id    ITEM,
   list_index  NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

list_id	**Form Builderがリスト項目を作成するときに割り当てる一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。**
list_name	**ユーザーがリスト項目を作成するときに付けた名前。その名前のデータ型はVARCHAR2です。**
list_index	**リスト索引値を指定します。リスト索引は1を基礎としています。**

使用上の注意

■ 「ほかの値のマッピング」プロパティが定義されていて、ブロック中に問合せレコードがあ

る場合は、DELETE_LIST_ELEMENTのビルトイン処理を実行しないでください。このような場合にCLEAR_LISTビルトインを使用すると、すでにフェッチされたレコードを表示できなくなる場合があります。

たとえば、値AおよびB、Cを含んでいるリスト項目があり、「ほかの値のマッピング」プロパティが定義されているとします。また、これらの値はデータベースからフェッチされたものであるとします（問合せはオープンしている）。この場合に、DELETE_LIST_ELEMENTによってリストからBを削除すると、エラーが起こります。これは、Form Builderが前回フェッチした値（A、B、C）を表示する際に、Bがリストから削除されているために値の表示が不可能となって発生するエラーです。

リスト要素の削除は、オープンしている問合せをすべてクローズしてから行ってください。オープンされた問合せのクローズには、ABORT_QUERYビルトインを使用します。

注意: 他の値要素が、設計時に全く指定されていなかったり、実行時にリストから自動的に削除された場合、そのリストには他の値要素は存在しません。

DELETE_LIST_ELEMENTの制限事項

ポップ・リストまたはTLISTスタイルのリスト項目では、ユーザーがデフォルトの値要素を削除しようとすると、Form Buildeから「FRM-41331: <list_item>から要素を削除できませんでした」というエラー・メッセージが戻ります。

デフォルト値要素とは、設計時に、「初期値」プロパティの設定でラベルや値を指定した要素のことです。

コンボボックス形式のリスト項目については、「初期値」プロパティに要素ラベルではなく実際の値が設定されている場合に限り、デフォルト値要素を削除できます。

実表の「ポップリスト」または「Tlist」項目に対して、次のことを行うと、Form Builderから「FRM-41331: <list_item>から要素を削除できませんでした」というエラー・メッセージが戻ります。

- ブロック中に問合せレコードや変更レコードがある状態で、他の値要素を削除しようとした場合。
- ブロック中に問合せレコードや変更レコードがある状態で、他の値要素を持たないリストから要素を削除しようとした場合。

  注意: 問合せされたレコードがブロックに含まれている場合、ブロック・ステータスはQUERYです。挿入または更新されたレコードがブロックに含まれている（問合せされたレコードが修正された）場合、ブロック・ステータスはCHANGEDです。

DELETE_LIST_ELEMENTの例

```
/*
```

```
**  Built-in: DELETE_LIST_ELEMENT
**  Example:  See ADD_LIST_ELEMENT
*/
```

DELETE_PARAMETERビルトイン

説明

指定のキーを持つパラメータを、パラメータ・リストから削除します。

構文

```
PROCEDURE DELETE_PARAMETER
  (list  VARCHAR2,
   key   VARCHAR2);
PROCEDURE DELETE_PARAMETER
  (name  VARCHAR2,
   key   VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

list*または*name	**パラメータ・リストを示す（リストIDまたはリスト名で指定）。実際のパラメータには、PARAMLIST型のパラメータ・リストIDまたはパラメータ・リストのVARCHAR2名のどちらかを指定できます。**
key	**パラメータの名前。キーのデータ型はVARCHAR2です。**

DELETE_PARAMETERの制限事項

- パラメータ・リストに最後に残ったパラメータを削除しても、その削除処理でリスト自体が自動的に削除されることはありません。パラメータ・リストを削除する場合は、DESTROY_PARAMETER_LISTサブプログラムをコールしてください。

DELETE_PARAMETERの例

```
/*
** Built-in: DELETE_PARAMETER
** Example:  Remove the 'NUMBER_OF_COPIES' parameter from the
**           already existing 'TEMPDATA' parameter list.
*/
```

```
BEGIN
  Delete_Parameter('tempdata','number_of_copies');
END;
```

DELETE_RECORDビルトイン

説明

On-Deleteトリガー以外で使用する場合は、ブロックから現レコードを削除し、そのレコードに削除マークを設定します。このビルトイン処理でレコードを削除すると、1回に1レコードずつ削除されるのではなく、削除対象レコードが削除レコード・リストに追加されます。次に有効なコミット・プロセスを行うときにこの削除リスト中のレコードがまとめて削除されます。

削除するレコードがデータベース中の行に対応している場合、Form Builderによりそのレコードをロックしてから、レコードの削除と削除マークの設定が行われます。

オープンしている問合せがブロック中に存在する場合、Form Builderにより、必要に応じてレコードがフェッチされ、ブロック内に取り込まれます。CLEARRECCLEAR_RECORDビルトイン・サブプログラムの説明も参照してください。

On-Deleteトリガーの中でDELETE_RECORDを実行すると、「トランザクションのポストおよびコミット」プロセスの中で、Form Builderのデフォルトのレコード削除処理が起動します。このコーディング例を、この後の例2に示します。

構文

```
PROCEDURE DELETE_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

DELETE_RECORDの例

例1

```
/*
```

```
** Built-in: DELETE_RECORD
** Example:  Mark the current record in the current block for
**           deletion.
*/
BEGIN
  Delete_Record;
END;
```

例2

```
/*
** Built-in: DELETE_RECORD
** Example:  Perform Form Builder delete record processing
**           during commit-time.Decide whether to use this
**           Built-in or a user exit based on a global flag
**           setup at startup by the form, perhaps based on
**           a parameter.
** Trigger:  On-Delete
*/
BEGIN
  /*
  ** Check the global flag we set during form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    User_Exit('my_delrec block=EMP');
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Delete_Record;
  END IF;
END;
```

DELETE_TIMERビルトイン

説明

フォームから指定のタイマーを削除します。

構文

```
PROCEDURE DELETE_TIMER
  (timer_id  Timer);
```

```
PROCEDURE DELETE_TIMER
  (timer_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

timer_id	特に、CREATE_TIMERビルトインへの正常なコールに対する応答としてタイマーを作成するときにForm Builderにより割り当てられる一意のIDを指定します。FIND_TIMERビルトインを使用して、そのIDを適切に入力された変数に戻します。このIDのデータ型はTIMERです。
timer_name	タイマーを定義する際につけたタイマーの名前。timer_nameのデータ型はVARCHAR2です。

DELETE_TIMERの制限事項

- タイマーを削除する場合、タイマー・オブジェクトの可用性チェックのためのID_NULLをコールしますが、このとき、ID_NULLのコールの前にFIND_TIMERコールを発行してください。たとえば、次の例は誤りです。DELETE_TIMERへのコールによってIDの値が設定されないからです。つまり、タイマーは削除されますが、IDは依然存在しており、存在しないタイマーを指しているため、NULLにはなりません。

```
-- Invalid Example
   timer_id := Find_Timer('my_timer');
   Delete_Timer(timer_id);
   IF (ID_Null(timer_id))...
```

DELETE_TIMERの例

```
/*
** Built-in: DELETE_TIMER
** Example:  Remove a timer after first checking to see if
**           it exists
*/
PROCEDURE Cancel_Timer( tm_name VARCHAR2 ) IS
  tm_id Timer;
BEGIN
  tm_id := Find_Timer( tm_name );

  IF NOT Id_Null(tm_id) THEN
    Delete_Timer(tm_id);
```

```
  ELSE
    Message('Timer '||tm_name||' has already been cancelled.');
  END IF;
END;
```

DELETE_TREE_NODEビルトイン

説明

ツリーからデータ要素を削除します。

構文

```
PROCEDURE DELETE_TREE_NODE
   (item_name VARCHAR2,
node NODE);
PROCEDURE DELETE_TREE_NODE
   (item_id ITEM,
node NODE);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
node	有効なノードを指定します。

使用上の注意

ブランチ・ノードを削除すると、そのすべての子ノードも削除されます。

DELETE_TREE_NODEの例

```
/*
** Built-in:DELETE_TREE_NODE
*/
```

```
-- This code finds a node with the label "Zetie"
-- and deletes it and all of its children.

DECLARE
   htree          ITEM;
   delete_node    FTREE.NODE;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Find the node with a label of "Zetie".
   -- Start searching from the root of the tree.
   delete_node := Ftree.Find_Tree_Node(htree,
                                       'Zetie',
                                        Ftree.FIND_NEXT,
                                        Ftree.NODE_LABEL,
                                        Ftree.ROOT_NODE,
                                        Ftree.ROOT_NODE);

   -- Delete the node and all of its children.
   IF NOT Ftree.ID_NULL(delete_node) then
       Ftree.Delete_Tree_Node(htree, delete_node);
   END IF;
END;
```

DESTROY_PARAMETER_LISTビルトイン

説明

動的に作成されたパラメータ・リストと、そのリスト中のすべてのパラメータを削除します。

構文

```
PROCEDURE DESTROY_PARAMETER_LIST
  (list  VARCHAR2);
PROCEDURE DESTROY_PARAMETER_LIST
  (name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

*list*または*name*	パラメータ・リストを示す（リストIDまたはリスト名で指定）。実際のパラメータには、PARAMLIST型のパラメータ・リストIDまたはパラメータ・リストのVARCHAR2名のどちらかを指定できます。

使用上の注意

DESTROY_PARAMETER_LISTを使用してパラメータ・リストが破棄された場合、パラメータ・リスト・ハンドルはNULLに設定されません。

このIDをPARAMLIST型の変数に戻す場合は、GET_PARAMETER_LISTのビルトイン処理を実行してください。

DESTROY_PARAMETER_LISTの例

```
/*
** Built-in:  DESTROY_PARAMETER_LIST
** Example:  Remove the parameter list 'tempdata' after first
**         checking to see if it exists
*/
DECLARE
  pl_id ParamList;
BEGIN
  pl_id := Get_Parameter_List('tempdata');
  IF NOT Id_Null(pl_id) THEN
    Destroy_Parameter_List(pl_id);
  END IF;
END;
```

DESTROY_VARIANTビルトイン

説明

CREATE_VARファンクションによって作成されたバリアントを破棄します。

構文

```
PROCEDURE DESTROY_VARIANT (variant OLEVAR);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

variant	破棄されるOLEバリアント。

DISPATCH_EVENTビルトイン

説明

ActiveXコントロール・イベントのディスパッチ・モードを指定します。デフォルトで、ActiveXイベントに関連付けられているすべてのPL/SQLイベント・プロシージャは制限付きです。つまり、go_itemはプロシージャ内からコールすることができず、OUTパラメータは無視されることを意味します。ただし、同じイベントが複数の項目に適用され、go_itemが必要になるインスタンスがあります。これによって、イベントを制限なしでディスパッチする必要がでてきます。On-Dispatch-Eventトリガーを使用すると、DISPATCH_EVENTをコールして、ディスパッチ・モードを制限付きまたは制限なしに指定できます。ActiveXコントロール・イベントの操作方法の詳細は、オンライン・ヘルプの「ActiveXコントロール・イベントのイベントに応答」を参照してください。

構文

```
PROCEDURE DISPATCH_EVENT
  (sync NUMBER,
);
PROCEDURE DISPATCH_EVENT
);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

sync	ディスパッチ・モードを制限付き（RESTRICTED）または制限なし（RESTRICTED_ALLOWED）のいずれかに指定します。

DISPATCH_EVENTの例

```
/*
```

```
ON-DISPATCH-EVENT trigger
*/
BEGIN
    IF :SYSTEM.CUSTOM_ITEM_EVENT = 4294966696 THEN
       /*when event occurs, allow it to apply to different items */
       FORMS4W.DISPATCH_EVENT(RESTRICTED_ALLOWED);
    ELSE
       /*run the default, that does not allow applying any other
       item */
       FORMS4W.DISPATCH_EVENT(RESTRICTED);
    END IF;
END;
```

DISPLAY_ERRORビルトイン

説明

ログに書き込まれたエラーがある場合、「エラー表示」画面を表示します。オペレータが「エラー表示」画面を表示しながらファンクション・キーを押すと、Form Builderによってフォームが再表示されます。このビルトインのコール時に、表示するエラーがない場合は、Form Builderによってコールが無視され、「エラー表示」画面も表示されません。

構文

```
PROCEDURE DISPLAY_ERROR;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

DISPLAY_ITEMビルトイン

説明

下位互換性のために用意されています。新規のアプリケーションには、SET_ITEM_INSTANCE_PROPERTYビルトインを使用してください。DISPLAY_ITEMは、指定された表示属性を項目に割り当てて、その項目の外観を変更します。DISPLAY_ITEMには、変更済みインスタンスをミラーする任意の項目の外観も変更する副作用があります。SET_ITEM_INSTANCE_PROPERTYはミラー項目を変更しません。

現行フォーム内の項目はどれでも参照できます。

DISPLAY_ITEMのビルトイン処理で変更した内容は、次の場合に無効となります。

- 同じ項目のインスタンスが、別のDISPLAY_ITEMビルトインによって参照された場合。
- 同じ項目のインスタンスが、SET_ITEM_INSTANCE_PROPERTYビルトイン（VISUAL_ATTRIBUTEプロパティあり）によって参照された場合。
- 項目のインスタンスが削除された場合（CLEAR_RECORDまたは問合せなどによって）。
- レコードを変更して（ステータスはNEW）、いったんレコードから抜け出た後、再度レコードをアクティブにした場合。
- 現行フォームを終了した場合。

構文

```
PROCEDURE DISPLAY_ITEM
  (item_id    ITEM,
   attribute  VARCHAR2);
PROCEDURE DISPLAY_ITEM
  (item_name  VARCHAR2,
   attribute  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_id	Form Builderが項目を作成するときに割り当てる一意のIDを指定します。このIDのデータ型はITEMです。

item_name	項目を作成するときに項目名として指定したVARCHAR2型の文字列を指定します。
attribute	設定するユーザー命名の可視属性を指定します。その他に、Oracle*Terminalリソース・ファイルの中の有効な属性も指定できます。Form Builderでは、ユーザー命名の可視属性から先に検索されます。

注意: 項目にデフォルト属性を適用する方法としてNormalを指定できますが、フォームにNormalと呼ばれる可視属性または論理属性（キャラクタ・モードまたはそれ以外）が含まれていない場合に限られます。また、項目をその初期可視属性（デフォルト、カスタムまたはユーザー命名）に戻す方法として、NULLを指定できます。

DISPLAY_ITEMの例

```
/*
** Built-in: DISPLAY_ITEM
** Example:  Change the visual attribute of each item in the
**           current record.
*/
DECLARE
  cur_itm   VARCHAR2(80);
  cur_block VARCHAR2(80) := :System.Cursor_Block;
BEGIN
  cur_itm   := Get_Block_Property( cur_block, FIRST_ITEM );
  WHILE ( cur_itm IS NOT NULL ) LOOP
    cur_itm := cur_block||'.'||cur_itm;
    Display_Item( cur_itm, 'My_Favorite_Named_Attribute');
    cur_itm := Get_Item_Property( cur_itm, NEXTITEM );
  END LOOP;
END;
```

DOWNビルトイン

説明

次のレコードの、同じ項目のインスタンスにナビゲートします。必要に応じて、Form Builderによりレコードがフェッチされます。Form Builderによりレコードが新規作成された場合は、そのレコードの中で最初のナビゲート可能項目に移動します。

構文

```
PROCEDURE DOWN;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

DO_KEYビルトイン

説明

指定のビルトイン・サブプログラムに対応するキー・トリガーを実行します。対応するキー・トリガーが存在しない場合は、指定のサブプログラムを実行します。このビルトイン処理でサブプログラムを実行しても、対応するファンクション・キーを押して実行しても、得られる結果は同じです。

構文

```
PROCEDURE DO_KEY
  (built-in_subprogram_name  VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

built-in_subprogram_name	有効なビルトイン・サブプログラムの名前を指定します。

ビルトイン	キー・トリガー	対応付けられたファンクション・キー
BLOCK_MENU	Key-MENU	「ブロック・メニュー」
CLEAR_BLOCK	Key-CLRBLK	「ブロック消去」
CLEAR_FORM	Key-CLRFRM	「フォーム消去」
CLEAR_RECORD	Key-CLRREC	「レコード消去」
COMMIT_FORM	Key-COMMIT	「Commit」
COUNT_QUERY	Key-CQUERY	「問合せ件数」
CREATE_RECORD	Key-CREREC	「レコード作成」
DELETE_RECORD	Key-DELREC	「レコード削除」
DOWN	Key-DOWN	「下へ」

DUPLICATE_ITEM	Key-DUP-ITEM	「項目の複製」
DUPLICATE_RECORD	Key-DUPREC	「レコードの複製」
EDIT_TEXTITEM	Key-EDIT	「編集」
ENTER	Key-ENTER	「Enter」
ENTER_QUERY	Key-ENTQRY	「問合せ入力」
EXECUTE_QUERY	Key-EXEQRY	「問合せ実行」
EXIT_FORM	Key-EXIT	「Exit/Cancel」
HELP	Key-HELP	「ヘルプ」
LIST_VALUES	Key-LISTVAL	「List」
LOCK_RECORD	Key-UPDREC	「レコード・ロック」
NEXT_BLOCK	Key-NXTBLK	「次のブロックへ」
NEXT_ITEM	Key-NEXT-ITEM	「次の項目へ」
NEXT_KEY	Key-NXTKEY	「Next Primary Key Fld」
NEXT_RECORD	Key-NXTREC	「次のレコードへ」
NEXT_SET	Key-NXTSET	「次のレコード・セットへ」
PREVIOUS_BLOCK	Key-PRVBLK	「前のブロックへ」
PREVIOUS_ITEM	Key-PREV-ITEM	「前の項目へ」
PREVIOUS_RECORD	Key-PRVREC	「前のレコードへ」
PRINT	Key-PRINT	「印刷」
SCROLL_DOWN	Key-SCRDOWN	「上方へスクロール」
SCROLL_UP	Key-SCRUP	「下方へスクロール」
UP	Key-UP	「上へ」

DO_KEYの制限事項

DO_KEYのパラメータに使用できるのはビルトイン名DO_KEY(ENTER_QUERY)のみです。特定キー名を受け入れるには、EXECUTE_TRIGGERビルトインEXECUTE_TRIGGER('KEY_F11')を実行します。

DO_KEYの例

```
/*
** Built-in: DO_KEY
** Example:  Simulate pressing the [Execute Query] key.
*/
BEGIN
  Do_Key('Execute_Query');
END;
```

DUMMY_REFERENCEビルトイン

説明

このビルトイン処理を使用すると、バインド変数を明示的に参照するよう設定できます。通常、バインド変数は計算式（または計算式からコールされるファンクションやプロシージャ）の中で間接的にしか参照できません。DUMMY_REFERENCEを使用すれば、Form Builderによって、計算項目（バインド変数の間接的な参照を含む）に再計算マークが設定されます。

式は、文字、数、または日付型のいずれでも構文いません。一般に式は、バインド変数へのシングル参照からなります。

注意: Form Builderに参照バインド変数を認識させるために、そのバインド変数に対してDUMMY_REFERENCEを実行する必要はありません（実行すると、関連する計算項目に再計算マークが付けられてしまいます）。

構文

```
PROCEDURE DUMMY_REFERENCE(式);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

DUMMY_REFERENCEの制限事項

なし

DUPLICATE_ITEMビルトイン

説明

現項目に、前のレコードの同一項目のインスタンスと同じ値を割り当てます。

構文

```
PROCEDURE DUPLICATE_ITEM;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

DUPLICATE_ITEMの制限事項

現行のセッションにおいて、必ず前のレコードが存在している必要があります。前のレコードがないと、Form Builderは「FRM-41803: 値をコピーする元の前レコードがありません。」というエラー・メッセージを返します。

DUPLICATE_RECORDビルトイン

説明

一つ前のレコードの各項目の値を、現レコードの対応する項目にコピーします。このとき、現レコードがデータベース中の行に対応しないでください。データベース中の行に対応していると、エラーが発生します。

注意: 複製レコードは、ソース・レコードのレコード・ステータスを継承しません。その代わりに、レコード・ステータスはINSERTになります。

構文

```
PROCEDURE DUPLICATE_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

DUPLICATE_RECORDの制限事項

現行のセッションにおいて、前のレコードが存在している必要があります。

DUPLICATE_RECORDの例

```
/*
** Built-in: DUPLICATE_RECORD;
** Example:  Make a copy of the current record and increment
**           the "line_sequence" item by one.
*/
DECLARE
  n NUMBER;
BEGIN
  /*
  ** Remember the value of the 'line_sequence' from the
  ** current record
  */
  n := :my_block.line_sequence;
  /*
  ** Create a new record, and copy all the values from the
  ** previous record into it.
  */
  Create_Record;
  Duplicate_Record;
  /*
  ** Set the new record's 'line_sequence' to one more than
  ** the last record's. */
  :my_block.line_sequence := n + 1;
END;
```

EDIT_TEXTITEMビルトイン

説明

ランフォームの項目エディタを起動して現行のテキスト項目を開き、フォームを編集モードにします。

構文

```
PROCEDURE EDIT_TEXTITEM;
PROCEDURE EDIT_TEXTITEM
  (x  NUMBER,
   y  NUMBER);
```

```
PROCEDURE EDIT_TEXTITEM
  (x      NUMBER,
   y      NUMBER,
   width  NUMBER,
   height  NUMBER);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

x	項目エディタのポップアップ・ウィンドウ表示位置のx座標を指定します。ここでの表示位置とは、画面上に表示するときのウィンドウの左上角の座標を指します。
y	項目エディタのポップアップ・ウィンドウ表示位置のy座標を指定します。ここでの表示位置とは、画面上に表示するときのウィンドウの左上隅の座標を指します。
width	エディタ・ウィンドウ全体（ボタンを含む）の横幅を指定します。
height	エディタ・ウィンドウ全体（ボタンを含む）の縦の長さを指定します。 縦の長さを6文字セル以下に指定すると、Form Builderによって縦の長さが6文字セルに設定されます。

項目エディタのポップアップ・ウィンドウ表示位置やウィンドウ・サイズを指定する場合に、このEDIT_TEXTITEMオプション・パラメータを使用します。このオプション・パラメータを指定しないと、Form Builderでは、デフォルトの表示位置とウィンドウ・サイズで項目エディタが起動されます。

EDIT_TEXTITEMの制限事項

- 横幅には、少なくともエディタ・ウィンドウ下部のボタンが表示できる長さを指定してください。

EDIT_TEXTITEMの例

```
/*
** Built-in:  EDIT_TEXTITEM
** Example  : Determine the x-position of the current item
**            then bring up the editor either on the left
**            side or right side of the screen so as to not
**            cover the item on the screen.
*/
DECLARE
```

```
  itm_x_pos NUMBER;
BEGIN
  itm_x_pos := Get_Item_Property(:System.Cursor_Item,X_POS);
  IF itm_x_pos > 40 THEN
    Edit_TextItem(1,1,20,8);
  ELSE
    Edit_TextItem(60,1,20,8);
  END IF;
END;
```

ENFORCE_COLUMN_SECURITYビルトイン

説明

データベースの列に対して、列のセキュリティ・チェックのデフォルト処理を実行します。Oracle以外のデータ・ソースに対して実行するアプリケーションにおいて、Form Builderのデフォルト・トランザクション処理の代わりにトランザクション・トリガーを実行する場合に、このビルトイン処理をインクルードするのが主な用途です。

列のセキュリティ・チェックのデフォルト処理では、Form Builderによって列レベルのセキュリティがかけられる際に発生したイベントが、発生順に参照されます。このセキュリティ・チェックは、「列セキュリティの強化」ブロック・ プロパティが「はい」に設定されているブロックに対して実行されます。列のセキュリティをかける前に、Form Builderではデータベースに問い合わせて、実表の列の中で現行フォームのオペレータが更新権限を持っている列を調べます。オペレータが更新権限を持っていない列では、Form Builderによってフォームの中の該当する実表項目が更新不可能となります。具体的には、「更新可」項目のプロパティを動的に「いいえ」に設定します。デフォルトでは、Form Builderによってフォームの起動時に各ブロックに対して順次この操作が行われます。

構文

```
PROCEDURE ENFORCE_COLUMN_SECURITY
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

使用上の注意

このビルトイン・サブプログラムをOn-Column-Securityトリガーにインクルードすると、このトリガーの処理を補強できます。ただし、同トリガーの処理そのものを代行させることはできません。詳細は、このマニュアル「トリガー」を参照してください。

ENFORCE_COLUMN_SECURITYの制限事項

このビルトイン処理は、On-Column-Securityトリガーの中でのみ使用できます。

ENTERビルトイン

説明

現行の妥当性チェック単位内でデータの妥当性チェックを行います（デフォルトの妥当性チェック単位は項目です）。

構文

```
PROCEDURE ENTER;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

ENTERの例

```
/*
** Built-in: ENTER
** Example:  Force Validation to occur before calling another
**           form
*/
BEGIN
  Enter;
  IF NOT Form_Success THEN
    RAISE Form_Trigger_Failure;
  END IF;
```

```
  Call_Form('newcust');
END;
```

ENTER_QUERYビルトイン

説明

ENTER_QUERYの動作は、指定のパラメータに応じて異なります。

構文

```
PROCEDURE ENTER_QUERY;
PROCEDURE ENTER_QUERY
  (keyword_one  VARCHAR2);
PROCEDURE ENTER_QUERY
  (keyword_two  VARCHAR2);
PROCEDURE ENTER_QUERY
  (keyword_one  VARCHAR2,
   keyword_two  VARCHAR2);
PROCEDURE ENTER_QUERY
  (keyword_one  VARCHAR2,
   keyword_two  VARCHAR2,
   locking      VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可（ブロックで最後に実行された問合せの結果、得られた例レコードを再表示する場合）

パラメータ

パラメータなし	**ENTER_QUERYと指定すると、現ブロックがフラッシュされ、フォームが問合せ入力モードになります。コミットすべき変更点がある場合は、ENTER_QUERYプロセスの中で、Form Builderからオペレータに対し、変更点をコミットするよう指示が出されます。**
keyword_one	**ENTER_QUERY(ALL_RECORDS)と指定すると、EXECUTE_QUERY起動時にForm Builderにより、選択済みレコードがすべてフェッチされます。それ以外のアクションは、ENTER_QUERYのみを指定した場合と同様です。**
keyword_two	**ENTER_QUERY(FOR_UPDATE)と指定すると、EXECUTE_QUERY起動時にForm Builderでは、選択済みレコードすべてを即時にロックしようとします。それ以外のアクションは、ENTER_QUERYのみを指定した場合と同様です。**

keyword_one/ keyword_two	ENTER_QUERY(ALL_RECORDS, FOR_UPDATE)と指定すると、EXECUTE_QUERY起動時にForm Builderでは、選択済みレコードすべてを即時にロックしようとし、さらにこれらのレコードをフェッチします。それ以外のアクションは、ENTER_QUERYのみを指定した場合と同様です。
locking	FOR_UPDATEパラメータを使用する場合に限り、このパラメータをNO_WAITに設定できます。NO_WAITと設定すると、即時更新用のロックがかからないレコードがあった場合に、オペレータにその旨を示すダイアログがForm Builderから表示されます。 NO_WAITパラメータを指定しない場合、Form Builderにより、ロックがかかるまでトライが続けられます。このとき、オペレータはこのプロセスを取り消すことはできません。 NO_WAITパラメータは、この機能をサポートするデータ・ソースに対して実行する場合に限り使用してください。

ENTER_QUERYの制限事項

ALL_RECORDSパラメータおよびFOR_UPDATEパラメータは注意して使用してください。多数の行のロックやフェッチでは、作業遂行上、大量のリソースを取得する必要があるため、処理に時間がかかることがあります。

ENTER_QUERYの例

```
/*
** Built-in: ENTER_QUERY
** Example:  Go Into Enter-Query mode, and exit the form if
**           the user cancels out of enter-query mode.
*/
BEGIN
  Enter_Query;
  /*
  ** Check to see if the record status of the first record
  ** is 'NEW' immediately after returning from enter-query
  ** mode.It should be 'QUERY' if at least one row was
  ** returned.
  */

  IF :System.Record_Status = 'NEW' THEN
    Exit_Form(No_Validate);
  END IF;
END;
```

ERASEビルトイン

説明

指定のグローバル変数を削除します(これにより、このグローバル変数は存在しなくなります)。また、そのグローバル変数が格納されていたメモリーを解放します。各グローバル変数には、それぞれ255バイトの記憶領域が割り当てられます。必要以上にパフォーマンスが低下するのを避けるため、不要なグローバル変数があれば消去してください。

構文

```
PROCEDURE ERASE
  (global_variable_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

global_variable_name	有効なグローバル変数の名前を指定します。

ERASEの例

```
ERASE('global.var');
```

ERROR_CODEビルトイン

説明

Form Builderで発生したエラーの番号を戻します。

構文

```
PROCEDURE ERROR_CODE;
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

パラメータ

なし

ERROR_CODEの例

```
/*
** Built-in: ERROR_CODE,ERROR_TEXT,ERROR_TYPE
** Example:   Reword certain FRM error messages by checking
**            the Error_Code in an ON-ERROR trigger
** Trigger:   On-Error
*/
DECLARE
  errnum NUMBER        := ERROR_CODE;
  errtxt VARCHAR2(80)  := ERROR_TEXT;
  errtyp VARCHAR2(3)   := ERROR_TYPE;
BEGIN
  IF errnum = 40301 THEN
    Message('Your search criteria identified no matches...
             Try Again.');
  ELSIF errnum = 40350 THEN
    Message('Your selection does not correspond to an employee.');
  ELSE
    /*
    ** Print the Normal Message that would have appeared
    **
    ** Default Error Message Text Goes Here
    */
    Message(errtyp||'-'||TO_CHAR(errnum)||':'||errtxt);
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

ERROR_TEXTビルトイン

説明

Form Builderで発生したエラーのメッセージ・テキストを戻します。

構文

```
FUNCTION ERROR_TEXT;
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

パラメータ

なし

使用上の注意

例外処理サブプログラムの実行中にエラー・メッセージをテストする場合にこのファンクションを指定してください。

ERROR_TEXTの例

```
/*
** Built-in: ERROR_CODE,ERROR_TEXT,ERROR_TYPE
** Example:  Reword certain FRM error messages by checking
**           the Error_Code in an ON-ERROR trigger
** Trigger:  On-Error
*/
DECLARE
  errnum NUMBER       := ERROR_CODE;
  errtxt VARCHAR2(80) := ERROR_TEXT;
  errtyp VARCHAR2(3)  := ERROR_TYPE;
BEGIN
  IF errnum = 40301 THEN
    Message('Your search criteria identified no matches...
            Try Again.');
  ELSIF errnum = 40350 THEN
    Message('Your selection does not correspond to an employee.');
  ELSE
    /*
    ** Print the Normal Message that would have appeared
    **
    ** Default Error Message Text Goes Here
    */
    Message(errtyp||'-'||TO_CHAR(errnum)||':'||errtxt);
```

```
  RAISE Form_Trigger_Failure;
END IF;
```

ERROR_TYPEビルトイン

説明

現行のRun Formセッションで最後に実行したアクションのエラー・メッセージ・タイプを返します。

構文

```
FUNCTION ERROR_TYPE;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

ERROR_TYPEは、エラー・メッセージ・タイプとして次のいずれかの値を戻します。

FRM　Form Builderで発生するエラーを示す値

ORA　Oracleで発生するエラーを示す値

問合せ入力モード

可

パラメータ

なし

使用上の注意

このファンクションは、次のような目的で使用します。

- キー入力などのユーザー・アクションの結果をテストし、On-Errorトリガー内の処理を確認するため。
- ビルトイン処理の結果をテストし、トリガーにおけるその後の処理を確認するため。

どちらのテストの場合も、対象となるアクションを実行したらすぐにテストします。テスト実行直前に対象外のアクションが入らないようにしてください。

ERROR_TYPEの例

```
/*
** Built-in: ERROR_CODE,ERROR_TEXT,ERROR_TYPE
** Example:  Reword certain FRM error messages by checking
**           the Error_Code in an ON-ERROR trigger
** Trigger:  On-Error
*/
DECLARE
  errnum NUMBER       := ERROR_CODE;
  errtxt VARCHAR2(80) := ERROR_TEXT;
  errtyp VARCHAR2(3)  := ERROR_TYPE;
BEGIN
  IF errnum = 40107 THEN
    Message('You cannot navigate to this non-displayed item...
         Try again.');
  ELSIF errnum = 40109 THEN
    Message('If you want to leave this block,
         you must first cancel Enter Query mode.');
  ELSE
    /*
    ** Print the Normal Message that would have appeared
    **
    ** Default Error Message Text Goes Here
    */
    Message(errtyp||'-'||TO_CHAR(errnum)||':'||errtxt);
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

EXEC_VERBビルトイン

説明

OLEサーバーが動詞名や動詞索引で識別した動詞を実行します。OLEオブジェクト上で実行可能なアクションをOLE動詞で指定します。

構文

```
PROCEDURE EXEC_VERB
  (item_id    Item,
   verb_index  VARCHAR2);
PROCEDURE EXEC_VERB
  (item_id    Item,
```

```
   verb_name  VARCHAR2);
PROCEDURE EXEC_VERB
  (item_name   VARCHAR2,
   verb_index  VARCHAR2);
PROCEDURE EXEC_VERB
  (item_name  VARCHAR2,
   verb_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
verb_index	動詞の数値索引を指定します。FORMS_OLE.Find_OLE_Verbビルトインを使用して、この値を取得します。索引のデータ型はVARCHAR2文字列です。
verb_name	動詞名を指定します。Forms_OLE.Get_Verb_Nameビルトインを使用して、この値を取得します。名前のデータ型はVARCHAR2 CHARです。

EXEC_VERBの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効。

EXEC_VERBの例

```
/*
** Built-in: EXEC_VERB
** Example:  Deactivates the OLE server associated with the object
**           in the OLE container.
** Trigger:  When-Button-Pressed
*/
DECLARE
 item_id  ITEM;
 item_name VARCHAR(25) := 'OLEITM';
 verb_cnt_str VARCHAR(20);
 verb_cnt NUMBER;
 verb_name VARCHAR(20);
 loop_cntr NUMBER;
```

```
BEGIN
 item_id := Find_Item(item_name);
 IF Id_Null(item_id) THEN
  message('No such item:'||item_name);
 ELSE
  verb_cnt_str := Forms_OLE.Get_Verb_Count(item_id);
  verb_cnt := TO_NUMBER(verb_cnt_str);
  FOR loop_cntr in 1..verb_cnt LOOP
    verb_name := Forms_OLE.Get_Verb_Name(item_id,loop_cntr);
    IF verb_name = 'Edit' THEN
   EXEC_VERB(item_id,verb_name);
    END IF;
  END LOOP;
 END IF;
END;
```

EXECUTE_QUERYビルトイン

説明

現ブロックのフラッシュおよび問合せのオープン、選択済みレコードのフェッチを行います。コミットすべき変更点がある場合は、EXECUTE_QUERY処理が一時中断され、Form Builderからオペレータに対し、変更点をコミットするよう指示が出されます。

構文

```
PROCEDURE EXECUTE_QUERY;
PROCEDURE EXECUTE_QUERY
  (keyword_one  VARCHAR2);
PROCEDURE EXECUTE_QUERY
  (keyword_two  VARCHAR2);
PROCEDURE EXECUTE_QUERY
  (keyword_one  VARCHAR2,
   keyword_two  VARCHAR2);
PROCEDURE EXECUTE_QUERY
  (keyword_one  VARCHAR2,
   keyword_two  VARCHAR2,
   locking      VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

パラメータなし	EXECUTE_QUERYと指定すると、現ブロックのフラッシュ、問合せのオープン、選択済みレコードのフェッチが行われます。
keyword_one	EXECUTE_QUERY(ALL_RECORDS)と指定すると、Form Builderにより、選択済みレコードすべてがフェッチされます。それ以外のアクションはEXECUTE_QUERYのみを指定した場合と同様です。
keyword_two	EXECUTE_QUERY(FOR_UPDATE)と指定すると、Form Builderにより、選択済みレコードすべてを即時にロックしようとします。それ以外のアクションはEXECUTE_QUERYのみを指定した場合と同様です。
keyword_one/ keyword_two	EXECUTE_QUERY(ALL_RECORDS, FOR_UPDATE)と指定すると、Form Builderでは、選択済みレコードすべてを即時にロックしようとし、さらにこれらのレコードをフェッチします。それ以外のアクションは、EXECUTE_QUERYのみの場合と同様です。
locking	FOR_UPDATEパラメータを使用する場合に限り、このパラメータをNO_WAITに設定できます。NO_WAITと設定すると、即時更新用のロックがかからないレコードがあった場合に、オペレータにその旨を示すダイアログがForm Builderから表示されます。 NO_WAITパラメータを指定しない場合、Form Builderにより、ロックがかかるまでトライが続けられます。このとき、オペレータはこのプロセスを取り消すことはできません。 NO_WAITパラメータは、この機能をサポートするデータ・ソースに対して実行する場合に限り使用してください。

EXECUTE_QUERYの制限事項

ALL_RECORDSパラメータおよびFOR_UPDATEパラメータは、注意して使用してください。多数の行をフェッチすると、処理に時間がかかることがあります。また、多数の行を一度にロックするには、大量のリソースが必要となります。

EXECUTE_QUERYの例

```
/*
** Built-in: EXECUTE_QUERY
** Example:  Visit several blocks and query their contents,
**           then go back to the block where original block.
*/
DECLARE
  block_before VARCHAR2(80) := :System.Cursor_Block;
BEGIN
  Go_Block('Exceptions_List');
  Execute_Query;
  Go_Block('User_Profile');
```

```
  Execute_Query;
  Go_Block('Tasks_Competed');
  Execute_Query;
  Go_Block( block_before );
END;
```

EXECUTE_TRIGGERビルトイン

説明

EXECUTE_TRIGGERでは、指定のトリガーを実行します。

構文

```
PROCEDURE EXECUTE_TRIGGER
  (trigger_name  VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ(ユーザー定義トリガーから制限付きビルトイン・サブプログラムをコールする場合)

問合せ入力モード

可

注意: EXECUTE_TRIGGERは、ユーザー命名トリガーを実行するのに好ましいメソッドではありません。ユーザー命名サブプログラムを作成することをお薦めします。

パラメータ

trigger_name	**有効なユーザー命名トリガーの名前を指定します。**

使用上の注意

このビルトイン処理では有効範囲を指定できないため、Form Builderでは、1番下のレベルから上のレベルに向かって、処理対象のトリガーが検索されます。

対応するキーのあるビルトイン処理を実行する場合は、EXECUTE_TRIGGERではなく、DO_KEYのビルトイン処理を使用してください。たとえば、次の項目へ移動する場合、

```
Execute_Trigger ('KEY-NEXT-ITEM');
```

ではなく、

```
Do_Key('NEXT_ITEM');
```

EXECUTE_TRIGGERの制限事項

EXECUTE_TRIGGERでは、ユーザー命名トリガーのほか、ビルトイン・トリガーも指定して実行できますが、ビルトイン・トリガーの指定はお薦めできません。これは、EXECUTE_TRIGGERでビルトイン・トリガーの起動に失敗した場合のデフォルト動作が、Form Builderのデフォルト処理の中でビルトイン・トリガーを自動起動したときとは異なる規則に基づいているためです。たとえば、デフォルト処理では、When-Validate-Itemトリガーの起動に失敗した場合は例外処理が行われ、フォームの処理は停止します。これに対し、EXECUTE_TRIGGERを使用してWhen-Validate-Itemトリガーを起動し、これに失敗した場合、フォームの処理は停止しません 。この場合、EXECUTE_TRIGGERビルトイン処理からの戻り値であるForm_FailureがFALSEにセットされて戻されます。

EXECUTE_TRIGGERの例

```
/*
** Built-in:EXECUTE_TRIGGER
** Example: Execute a trigger dynamically from the PL/SQL
**          code of a Menu item, depending on a menu
**          checkbox.
*/
DECLARE
  Cur_Setting   VARCHAR2(5);
  Advanced_Mode BOOLEAN;
BEGIN
  /*
  ** Check whether the 'Advanced' menu item under the
  ** 'Preferences' submenu is checked on or not.
  */
  Cur_Setting :=  Get_Menu_Item_Property
                 ('Preferences.Advanced',CHECKED);
  /*
  ** If it is checked on, then Advanced_Mode is boolean
  ** true.
  */
  Advanced_Mode := (Cur_Setting = 'TRUE');
  /*
  ** Run the appropriate trigger from the underlying form
  **
  */
  IF Advanced_Mode THEN
    Execute_Trigger('Launch_Advanced_Help');
  ELSE
    Execute_Trigger('Launch_Beginner_Help');
```

```
  END IF;
END;
```

EXIT_FORMビルトイン

説明

コミットを確認し、ロールバック・アクションを指定することにより、フォームを終了するための手段を提供します。

- 大抵のコンテキストの場合、EXIT_FORMはフォームを"外"へ導きます。現行フォームで変更した内容がポストまたはコミットされていない場合は、EXIT_FORM処理は一時中断し、Form Builderからオペレータに対してコミットするよう指示が出されます。
- オペレータが問合せ入力モードに入っている場合にEXIT_FORMを実行すると、問合せ入力モードの外にナビゲートされます。この場合、フォームからは出ません。
- CALL_INPUT処理中にEXIT_FORMを実行すると、CALL_INPUTファンクションを終了させます。

構文

```
PROCEDURE EXIT_FORM;
PROCEDURE EXIT_FORM
  (commit_mode  NUMBER);
PROCEDURE EXIT_FORM
  (commit_mode   NUMBER,
   rollback_mode NUMBER);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

commit_mode	ASK_COMMIT	EXIT_FORM処理の中で、Form Builderからオペレータに対し、コミットを行うよう指示が出されます。ただしRECORD_STATUSがINSERTでもレコードが無効の場合、Form Builderからオペレータに対してフォームを閉じるかどうかが確認されます。閉じることをオペレータが了承した場合、フォームが閉じられる前に変更がロールバックされます。
	DO_COMMIT	Form Builderにより、変更点の有効化およびコミット処理が行われ、現行フォームが終了します。この一連の処理の中で、オペレータへの指示はありませんNO_COMMIT Form Builderにより、変更点が有効化され、現行フォームが終了します。この一連の処理の中で、変更点のコミットは行われず、オペレータへの指示もありません。
	NO_VALIDATE	Form Builderでは、現行フォームを終了させます。このとき、変更点の有効化やコミット処理は行われず、オペレータへの指示もありません。
rollback_mode	TO_SAVEPOINT	Form Builderでは（ポスト済みのものを含む）コミットされていないすべての変更内容が現行フォームのセーブポイントにロールバックされます。
	FULL_ROLLBACK	Form Builderでは、現ランフォーム・セッション中に加えられたコミットされていないすべての変更内容（ポスト済みのものを含む）がロールバックされます。ポスト専用モードで実行されているフォームからFULL_ROLLBACKを指定することはできません。（フォームにポストしていないレコードがあるときに別のフォームをコールすると、ポスト専用モードになる可能性があります。コール側フォームが発行したロックの損失を防ぐために、Form Builderはコールされたフォーム内のコミット処理をすべて妨げます。
	NO_ROLLBACK	Form Builderでは、セーブポイントまでロールバックせずに現行フォームを終了させます。ロールバックせずに最上位レベルのフォームを終了させることもできますが、この場合は、NEW_FORM操作全体がロックされたままになります。外部3GLプログラムからForm Builderを実行している場合も同様のロックがかかります。このロックは、Form Builderから3GLプログラムに制御が戻っても保持されます。

使用上の注意

EXIT_FORMのデフォルト・パラメータは、commit_mode用のASK_COMMITおよびrollback_mode用のTO_SAVEPOINTであるため、一部のコンテキストでパラメータを指定せずにEXIT_FORMを起動すると、予期しない結果が生じる場合があります。たとえば、フォームがポスト専用モードにあり、EXIT_FORMがパラメータなしで起動されると、ユーザーに対して変更のコミットを求めるプロンプトが表示されます。ただし、そのプロンプトでの入力内容に関係なく、変更が行われたことを確認するメッセージが表示されますが、TO_SAVEPOINTのデフォルトのrollback_modeによって、フォームへの変更はロールバックされます。矛盾を回避するには、パラメータを明示的に指定します。

EXIT_FORMの例

```
/*
** Built-in: EXIT_FORM and POST
** Example:  Leave the called form, without rolling back the
**           posted changes so they may be posted and
**           committed by the calling form as part of the
**           same transaction.
*/
BEGIN
  Post;

  /*
  ** Form_Status should be 'QUERY' if all records were
  ** successfully posted.
  */
  IF :System.Form_Status <> 'QUERY' THEN
    Message('An error prevented the system from posting changes');
    RAISE Form_Trigger_Failure;
  END IF;
  /*
  ** By default, Exit_Form asks to commit and performs a
  ** rollback to savepoint.We've already posted, so we do
  ** not need to commit, and we don't want the posted changes
  ** to be rolled back.
  */
  Exit_Form(NO_COMMIT, NO_ROLLBACK);
END;
```

FETCH_RECORDSビルトイン

説明

On-Fetchトリガーからこのビルトイン処理をコールすると、Form Builderにより、デフォルトのフェッチ処理が起動されます。ここでフェッチされるのは、SELECT処理の結果、選択基準と一致したレコードです。

構文

```
PROCEDURE FETCH_RECORDS;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

このビルトインは主に、Oracle以外のデータ・ソースに対して実行されるアプリケーション用に組み込まれているものです。

パラメータ

なし

FETCH_RECORDSの例

```
/*
** Built-in: FETCH_RECORDS
** Example:  Perform Form Builder record fetch processing during
**           query time.Decide whether to use this built-in
**           or a user exit based on a global flag setup at
**           startup by the form, perhaps based on a
**           parameter.The block property RECORDS_TO_FETCH
**           allows you to know how many records Form Builder
**           is expecting.
** Trigger:  On-Fetch
*/
DECLARE
  numrecs NUMBER;
BEGIN
  /*
  ** Check the global flag we set during form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    /*
    ** How many records is the form expecting us to
    ** fetch?
    */
    numrecs := Get_Block_Property('EMP',RECORDS_TO_FETCH);
    /*
    ** Call user exit to determine if there are any
    ** more records to fetch from its cursor.User Exit
    ** will return failure if there are no more
    ** records to fetch.
    */
    User_Exit('my_fetch block=EMP remaining_records');
    /*
    ** If there ARE more records, then loop thru
```

```
    ** and create/populate the proper number of queried
    ** records.If there are no more records, we drop through
    ** and do nothing.Form Builder takes this as a signal that
    ** we are done.
    */
    IF Form_Success THEN
      /* Create and Populate 'numrecs' records */
      FOR j IN 1..numrecs LOOP
        Create_Queried_Record;
        /*
        ** User exit returns false if there are no more
        ** records left to fetch.We break out of the
        ** if we've hit the last record.
        */
        User_Exit('my_fetch block=EMP get_next_record');
        IF NOT Form_Success THEN
          EXIT;
        END IF;
      END LOOP;
    END IF;
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Fetch_Records;
  END IF;
END;
```

FIND_ALERTビルトイン

説明

Form Builderの有効な警告のリストを検索します。指定された警告が見つかると、サブプログラムは警告IDを戻します。 ユーザーは、適切に入力された変数にIDを戻す必要があります。変数はALERT型で定義してください。

構文

```
FUNCTION FIND_ALERT
  (alert_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

Alert

問合せ入力モード

可

パラメータ

alert_name	VARCHAR2型の警告名を指定します。

FIND_ALERTの例

```
/*
** Built-in: FIND_ALERT
** Example:   Show a user-warning alert.If the user presses
**            the OK button, then make REALLY sure they want
**            to continue with another alert.
*/
DECLARE
  al_id    Alert;
  al_button NUMBER;
BEGIN
  al_id := Find_Alert('User_Warning');
  IF Id_Null(al_id) THEN
    Message('User_Warning alert does not exist');
    RAISE Form_Trigger_Failure;
  ELSE
   /*
   ** Show the warning alert
   */
   al_button := Show_Alert(al_id);
   /*
   ** If user pressed OK (button 1) then bring up another
   ** alert to confirm -- button mappings are specified
   ** in the alert design
   */
   IF al_button = ALERT_BUTTON1 THEN
      al_id := Find_Alert('Are_You_Sure');

      IF Id_Null(al_id) THEN
```

```
        Message('The alert named:Are you sure? does not exist');
        RAISE Form_Trigger_Failure;
       ELSE
         al_button := Show_Alert(al_id);
         IF al_button = ALERT_BUTTON2 THEN
           Erase_All_Employee_Records;
         END IF;
       END IF;
     END IF;
   END IF;
END;
```

FIND_BLOCKビルトイン

説明

有効なブロック・リストを検索して、一意のブロックIDを戻します。適切な型を持つ変数を定義して戻り値を受け取ります。変数はBLOCK型で定義してください。

構文

```
FUNCTION FIND_BLOCK
  (block_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

Block

問合せ入力モード

可

パラメータ

block_name	VARCHAR2型のブロック名を指定します。

FIND_BLOCKの例

```
/*
** Built-in: FIND_BLOCK
** Example:  Return true if a certain blockname exists
*/
```

```
FUNCTION Does_Block_Exist( bk_name VARCHAR2 )
RETURN BOOLEAN IS
  bk_id Block;
BEGIN
  /*
  ** Try to locate the block by name
  */
  bk_id := Find_Block( bk_name );
  /*
  ** Return the boolean result of whether we found it.
  ** Finding the block means that its bk_id will NOT be NULL
  */
  RETURN (NOT Id_Null(bk_id));
END;
```

FIND_CANVASビルトイン

説明

キャンバス・リストを検索して、指定の名前を持つ有効キャンバスが見つかると、そのキャンバスのIDを戻します。適切な型を持つ変数を定義して戻り値を受け取ります。変数はCanvas型で定義してください。

構文

```
FUNCTION FIND_CANVAS
  (canvas_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

Canvas

問合せ入力モード

可

パラメータ

canvas_name	キャンバスを定義するときは、ユーザーが命名したVARCHAR2型のキャンバス名を指定します。

FIND_CANVASの例

```
DECLARE
  my_canvas Canvas;
BEGIN
  my_canvas := Find_Canvas('my_canvas');
END;
```

FIND_COLUMNビルトイン

説明

レコード・グループ列のリストを検索して、指定の名前を持つ有効な列が見つかると、そのグループ列のIDを戻します。適切な型を持つ変数を定義して戻り値を受け取ります。変数はGROUPCOLUMN型で定義してください。

構文

```
FUNCTION FIND_COLUMN
  (recordgroup.groupcolumn_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

GroupColumn

問合せ入力モード

可

パラメータ

recordgroup.groupcolumn_name	**VARCHAR2レコード・グループの完全修飾列名を指定します。**

FIND_COLUMNの例

```
/*
** Built-in: FIND_COLUMN
** Example   Get column IDs for three columns in a record
**           group before performing multiple Get's or Set's
**           of the record group's column values
*/
PROCEDURE Record_Machine_Stats( mach_number NUMBER,
```

```
                      pph        NUMBER,
                      temperature NUMBER) IS
  rg_id  RecordGroup;
  col1   GroupColumn;
  col2   GroupColumn;
  col3   GroupColumn;
  row_no NUMBER;
BEGIN
  rg_id := Find_Group('machine');
  col1  := Find_Column('machine.machine_no');
  col2  := Find_Column('machine.parts_per_hour');
  col3  := Find_Column('machine.current_temp');
  /*
  ** Add a new row at the bottom of the 'machine' record
  ** group, and make a note of what row number we just
  ** added.
  */
  Add_Group_Row( rg_id, END_OF_GROUP);
  row_no := Get_Group_Row_Count(rg_id);
  Set_Group_Number_Cell(col1, row_no, mach_number);
  Set_Group_Number_Cell(col2, row_no, pph);
  Set_Group_Number_Cell(col3, row_no, temperature);
END;
```

FIND_EDITORビルトイン

説明

エディタ・リストを検索し、指定の名前を持つ有効エディタを見つけると、エディタIDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をEDITOR型で定義します。

構文

```
FUNCTION FIND_EDITOR
  (editor_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

Editor

問合せ入力モード

可

パラメータ

editor_name	有効なVARCHAR2エディタ名を指定します。

FIND_EDITORの例

```
/*
** Built-in: FIND_EDITOR
** Example:  Find and show a user-defined editor
*/
DECLARE
  ed_id  Editor;
  status BOOLEAN;
BEGIN
  ed_id := Find_Editor('Happy_Edit_Window');

  IF NOT Id_Null(ed_id) THEN
    Show_Editor(ed_id, NULL, :emp.comments, status);
  ELSE
    Message('Editor "Happy_Edit_Window" not found');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

FIND_FORMビルトイン

説明

フォーム・リストを検索し、指定の名前を持つ有効フォームが見つかると、フォーム・モジュールIDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をMORMMODULE型で定義します。

構文

```
FUNCTION FIND_FORM
  (formmodule_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

FormModule

問合せ入力モード

可

パラメータ

formmodule_name	有効なVARCHAR2フォーム名を指定します。

FIND_FORMの例

```
/*
** Built-in: FIND_FORM
** Example:   Find a form's Id before inquiring about several
**             of its properties
*/
DECLARE
  fm_id  FormModule;
  tmpstr VARCHAR2(80);
BEGIN
  fm_id := Find_Form(:System.Current_Form);
  tmpstr := Get_Form_Property(fm_id,CURSOR_MODE);
  tmpstr := tmpstr||','||Get_Form_Property(fm_id,SAVEPOINT_MODE);
  Message('Form is configured as:'||tmpstr);
END;
```

FIND_GROUPビルトイン

説明

レコード・グループ・リストを検索し、指定の名前を持つ有効グループが見つかると、レコード・グループIDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をRECORDGROUP型で定義します。

構文

```
FUNCTION FIND_GROUP
  (recordgroup_name VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り

RECORDGROUP

問合せ入力モード

可

パラメータ

recordgroup_name	有効なVARCHAR2レコード・グループ名を指定します。

FIND_GROUPの制限事項

このファンクションのパフォーマンスは、レコード・グループの数によって異なります。

FIND_GROUPの例

```
/*
** Built-in: FIND_GROUP
** Example:  See CREATE_GROUP and DELETE_GROUP_ROW
*/
```

FIND_ITEMビルトイン

説明

所定のブロック内で項目リストを検索し、指定の名前を持つ有効な項目が見つかると、項目IDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をITEM型で定義します。

構文

```
FUNCTION FIND_ITEM
  (block.item_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

Item

問合せ入力モード

可

パラメータ

block_name.item_name	完全修飾項目名を指定します。その名前のデータ型はVARCHAR2です。

FIND_ITEMの例

```
/*
** Built-in: FIND_ITEM
** Example:  Find an item's Id before setting several
**           of its properties.
*/
PROCEDURE Hide_an_Item( item_name VARCHAR2, hide_it BOOLEAN) IS
  it_id   Item;
BEGIN
  it_id := Find_Item(item_name);
  IF Id_Null(it_id) THEN
    Message('No such item:'||item_name);
    RAISE Form_Trigger_Failure;
  ELSE
    IF hide_it THEN
      Set_Item_Property(it_id,VISIBLE,PROPERTY_FALSE);
    ELSE
      Set_Item_Property(it_id,VISIBLE,PROPERTY_TRUE);
      Set_Item_Property(it_id,ENABLED,PROPERTY_TRUE);
      Set_Item_Property(it_id,NAVIGABLE,PROPERTY_TRUE);
    END IF;
  END IF;
END;
```

FIND_LOVビルトイン

説明

値リストを検索し、指定の名前を持つ有効な値リストが見つかると、値リスト IDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数を値リスト型で定義します。

構文

```
FUNCTION FIND_LOV  (LOV_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

LOV

問合せ入力モード

可

パラメータ

LOV_name	VARCHAR2型の値リストの有効な名前を指定します。

FIND_LOVの例

```
/*
** Built-in: FIND_LOV
** Example:  Determine if an LOV exists before showing it.
*/
DECLARE
  lv_id  LOV;
  status BOOLEAN;
BEGIN
  lv_id := Find_LOV('My_Shared_LOV');
  /*
  ** If the 'My_Shared_LOV' is not part of the current form,
  ** then use the 'My_Private_LOV' instead.
  */
  IF Id_Null(lv_id) THEN
    lv_id := Find_LOV('My_Private_LOV');
  END IF;
  status := Show_LOV(lv_id,10,20);
END;
```

FIND_MENU_ITEMビルトイン

説明

メニュー項目リストを検索し、指定の名前を持つ有効なメニュー項目が見つかると、メニュー項目IDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をMENUITEM型で定義します。

構文

```
FUNCTION FIND_MENU_ITEM
  (menu_name.menu_item_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

MenuItem

問合せ入力モード

可

パラメータ

menu_name.menu_item_name	有効なVARCHAR2型の完全修飾メニュー項目名を指定します。

FIND_MENU_ITEMの例

```
/*
** Built-in: FIND_MENU_ITEM
** Example:   Find the id of a menu item before setting
**            multiple properties
*/
PROCEDURE Toggle_AutoCommit_Mode IS
  mi_id MenuItem;
  val   VARCHAR2(10);
BEGIN
  mi_id := Find_Menu_Item('Preferences.AutoCommit');
  /*
  ** Determine the current checked state of the AutoCommit
  ** menu checkbox item
  */
  val := Get_Menu_Item_Property(mi_id,CHECKED);
  /*
  ** Toggle the checked state
  */
  IF val = 'TRUE' THEN
    Set_Menu_Item_Property(mi_id,CHECKED,PROPERTY_FALSE);
  ELSE
    Set_Menu_Item_Property(mi_id,CHECKED,PROPERTY_TRUE);
  END IF;
END;
```

FIND_OLE_VERBビルトイン

説明

OLE動詞索引を返します。OLEオブジェクト上で実行可能なアクションをOLE動詞で指定すると、OLE動詞のそれぞれに対応するOLE動詞索引が付加されます。OLE動詞索引はVARCHAR2文字列として返されるので、FORMS_OLE.EXE_VERBに使用するときは、NUMBERに変換する必要があります。適切な型を持つ変数を定義して戻り値を受け取ります。

構文

```
FUNCTION FIND_OLE_VERB
  (item_id    Item,
   verb_name  VARCHAR2);
FUNCTION FIND_OLE_VERB
  (item_name  VARCHAR2,
   verb_name  VARCHAR2);
```

戻し値

VARCHAR2

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
verb_name	OLE動詞名を指定します。OLEオブジェクト上で実行可能なアクションをOLE動詞で指定します。Forms_OLE.Get_Verb_Nameビルトインを使用して、この値を取得します。その名前のデータ型は、VARCHAR2文字列です。

FIND_OLE_VERBの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効。

FIND_OLE_VERBの例

```
/*
```

```
** Built-in: EXEC_VERB
** Example:  Finds an OLE verb index for use with the
**           Forms_OLE.Exec_Verb built-in.
** Trigger:  When-Button-Pressed
*/
DECLARE
 item_id  ITEM;
 item_name VARCHAR(25) := 'OLEITM';
 verb_index_str VARCHAR(20);
 verb_index NUMBER;
BEGIN
 item_id := Find_Item(item_name);
 IF Id_Null(item_id) THEN
  message('No such item:'||item_name);
 ELSE
  verb_index_str := Forms_OLE.Find_OLE_Verb(item_id,'Edit');
  verb_index := TO_NUMBER(verb_index_str);
  Forms_OLE.Exec_Verb(item_id,verb_index);
 END IF;
END;
```

FIND_RELATIONビルトイン

説明

リレーション・リストを検索し、指定の名前を持つ有効なリレーションが見つかると、リレーションIDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をRELATION型で定義します。

構文

```
FUNCTION FIND_RELATION
  (relation_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

Relation

問合せ入力モード

可

パラメータ

relation_name	有効なVARCHAR2型のリレーション名を指定します。

FIND_RELATIONの例

```
/*
** Built-in: FIND_RELATION
** Example:  Find the id of a relation before inquiring about
**           multiple properties
*/
FUNCTION Detail_of( Relation_Name VARCHAR2 )
RETURN VARCHAR2 IS
  rl_id Relation;
BEGIN
  rl_id := Find_Relation( Relation_Name );

  /*
  ** Signal error if relation does not exist
  */
  IF Id_Null(rl_id) THEN
    Message('Relation '||Relation_Name||' does not exist.');
    RAISE Form_Trigger_Failure;
  ELSE
    RETURN Get_Relation_Property(rl_id,DETAIL_NAME);
  END IF;
END;
```

FIND_REPORT_OBJECTビルトイン

説明

指定されたレポートのreport_idを返します。このIDを、RUN_REPORT_OBJECTなど、他のビルトインのパラメータとして使用できます。

構文

```
FUNCTION FIND_REPORT_OBJECT
  (report_name VARCHAR2
);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

report_name	検索するレポートの一意名を指定します。

FIND_REPORT_OBJECTの例

```
DECLARE
    repid REPORT_OBJECT;
    v_rep  VARCHAR2(100);
BEGIN
    repid := find_report_object('report4');
    v_rep := RUN_REPORT_OBJECT(repid);
    ....
END;
```

FIND_TAB_PAGEビルトイン

説明

指定のタブ・キャンバス内でタブ・ページのリストを検索し、指定の名前を持つ有効なタブ・ページが見つかると、タブ・ページIDを返します。戻り値を受け取るには、データ型TAB_PAGEの変数を定義する必要があります。

構文

```
FUNCTION FIND_TAB_PAGE
  (tab_page_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

tab_page

問合せ入力モード

可

パラメータ

tab_page_name	一意のタブ・ページ名。データ型はVARCHAR2です。（注意:複数のタブ・キャンバスが同じ識別名のタブ・ページを持っている場合、タブ・ページの完全修飾名、たとえばMY_TAB_CVS.TAB_PAGE_1を指定する必要があります。

FIND_TAB_PAGEの例

```
/* Use FIND_TAB_PAGE to find the ID of the top-most tab
** page on tab canvas TAB_PAGE_1, then use the ID to set
** properties of the tab page:
*/
DECLARE
  tp_nm   VARCHAR2(30);
  tp_id   TAB_PAGE;

BEGIN
  tp_nm := GET_CANVAS_PROPERTY('tab_page_1', topmost_tab_page);
  tp_id := FIND_TAB_PAGE(tp_nm);
  SET_TAB_PAGE_PROPERTY(tp_id, visible, property_true);
  SET_TAB_PAGE_PROPERTY(tp_id, label, 'Order Info');
END;
```

FIND_TIMERビルトイン

説明

タイマー・リストを検索し、指定の名前を持つ有効タイマーを見つけると、タイマーIDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をタイマー型で定義します。

構文

```
FUNCTION FIND_TIMER
  (timer_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り

Timer

問合せ入力モード

可

パラメータ

timer_name	有効なVARCHAR2型のタイマー名を指定します。

FIND_TIMERの例

```
/*
** Built-in: FIND_TIMER
** Example:  If the timer exists, reset it.Otherwise create
**           it.
*/
PROCEDURE Reset_Timer_Interval( Timer_Name VARCHAR2,
                      Timer_Intv NUMBER ) IS
  tm_id      Timer;
  tm_interval NUMBER;
BEGIN
  /*
  ** User gives the interval in seconds, the timer subprograms
  ** expect milliseconds
  */
  tm_interval := 1000 * Timer_Intv;
  /* Lookup the timer by name */
  tm_id := Find_Timer(Timer_Name);
  /* If timer does not exist, create it */
  IF Id_Null(tm_id) THEN
    tm_id := Create_Timer(Timer_Name,tm_interval,NO_REPEAT);
  /*
  ** Otherwise, just restart the timer with the new interval
  */
  ELSE
    Set_Timer(tm_id,tm_interval,NO_REPEAT);
  END IF;
END;
```

FIND_TREE_NODEビルトイン

説明

ラベルまたは値が検索文字列と一致する、ツリー内の次のノードを検索します。

構文

```
FUNCTION FIND_TREE_NODE
  (item_name VARCHAR2,
   search_string VARCHAR2,
```

```
   search_type NUMBER,
   search_by NUMBER,
   search_root NODE,
   start_point NODE);
FUNCTION FIND_TREE_NODE
  (item_id ITEM,
   search_string VARCHAR2,
   search_type NUMBER,
   search_by NUMBER,
   search_root NODE,
   start_point NODE);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NODE

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
search_string	VARCHAR2検索文字列を指定します。
search_type	NUMBER検索型を指定します。指定できる値は次のとおりです。 FIND_NEXT FIND_NEXT_CHILD search_rootノードの子のみを検索します。
search_by	検索条件となるNUMBERを指定します。指定できる値は次のとおりです。 NODE_LABEL NODE_VALUE
search_root	検索ツリーのルートを指定します。
start_point	NODE検索の開始点を指定します。

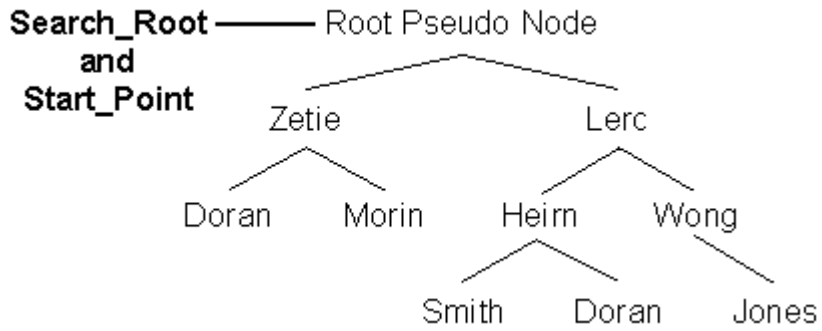
FIND_TREE_NODEの例

```
/*
```

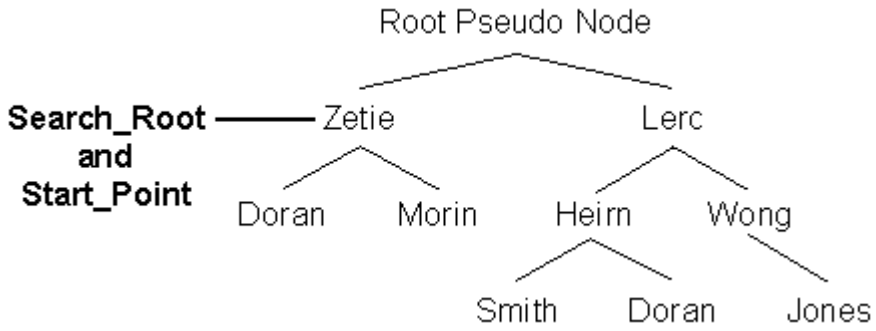
```
** Built-in:FIND_TREE_NODE
*/

-- This code finds a node with a label of "Doran"
-- within the subtree beginning with the a node
-- with a label of "Zetie".

DECLARE
   htree        ITEM;
   find_node     Ftree.NODE;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');
```




```
   -- Find the node with a label "Zetie".
   find_node := Ftree.Find_Tree_Node(htree, 'Zetie', Ftree.FIND_NEXT,
                Ftree.NODE_LABEL, Ftree.ROOT_NODE, Ftree.ROOT_NODE);
```




```
   -- Find the node with a label "Doran"
   -- starting at the first occurance of "Zetie".
   find_node := Ftree.Find_Tree_Node(htree, 'Doran', Ftree.FIND_NEXT,
```

```
                Ftree.NODE_LABEL, find_node, find_node);

  IF NOT Ftree.ID_NULL(find_node) then
     ...
  END IF;
END;
```

FIND_VAビルトイン

説明

任意のブロック内にある項目の可視属性を検索し、その属性の値をテキスト文字列として返します。

構文

```
FUNCTION FIND_VA
  (va_name PROPERTY);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VisualAttribute

問合せ入力モード

可

パラメータ

va_name	**可視属性の作成時に付けた名前。データ型はVARCHAR2です。**

FIND_VIEWビルトイン

説明

キャンバス・リストを検索し、指定の名前を持つ有効なキャンバスを見つけると、ビューIDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をVIEWPORT型で定義します。

構文

```
FUNCTION FIND_VIEW
  (viewcanvas_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

ViewPort

問合せ入力モード

可

パラメータ

viewcanvas_name	VARCHAR2文字列でキャンバス名を指定します。

FIND_VIEWの例

```
/*
** Built-in: FIND_VIEW
** Example:  Change the visual attribute and display position
**           of a stacked view before making it visible to
**           the user.
*/
DECLARE
  vw_id ViewPort;
BEGIN
  vw_id := Find_View('Sales_Summary');
  Set_Canvas_Property('Sales_Summary', VISUAL_ATTRIBUTE,
            'Salmon_On_Yellow');
  Set_View_Property(vw_id, VIEWPORT_X_POS, 30);
  Set_View_Property(vw_id, VIEWPORT_Y_POS, 5);
  Set_View_Property(vw_id, VISIBLE, PROPERTY_TRUE);
END;
```

FIND_WINDOWビルトイン

説明

ウィンドウ・リストを検索し、指定の名前を持つ有効ウィンドウを見つけると、ウィンドウIDを返します。適切な型を持つ変数を定義して戻り値を受け取ります。変数をWINDOW型で定義します。

構文

```
FUNCTION FIND_WINDOW
  (window_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

Window

問合せ入力モード

可

パラメータ

window_name	VARCHAR2文字列の有効なウィンドウ名を指定します。

FIND_WINDOWの例

```
/*
** Built-in:FIND_WINDOW
** Example:Anchor the upper left corner of window2 at the
**          bottom right corner of window1.
*/
PROCEDURE Anchor_Bottom_Right( Window2 VARCHAR2, Window1
VARCHAR2) IS
  wn_id1 Window;
  wn_id2 Window;
  x     NUMBER;
  y     NUMBER;
  w     NUMBER;
  h     NUMBER;
BEGIN
  /*   ** Find Window1 and get its (x,y) position, width,
       ** and height.
```

```
  */
  wn_id1 := Find_Window(Window1);
  x      := Get_Window_Property(wn_id1,X_POS);
  y      := Get_Window_Property(wn_id1,Y_POS);
  w      := Get_Window_Property(wn_id1,WIDTH);
  h      := Get_Window_Property(wn_id1,HEIGHT);
  /*
  ** Anchor Window2 at (x+w,y+h)
  */
  wn_id2 := Find_Window(Window2);
  Set_Window_Property(wn_id2,X_POS, x+w );
  Set_Window_Property(wn_id2,Y_POS, y+h );
END;
```

FIRST_RECORDビルトイン

説明

ブロック内のレコード・リストの最初のレコードへナビゲートします。

構文

```
PROCEDURE FIRST_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

FIRST_RECORDの例

```
/*
** Built-in: FIRST_RECORD
** Example:   Have a button toggle between the first and last
**            records in a block.
** Trigger:   When-Button-Pressed
*/
BEGIN
```

```
  /*
  ** If we're not at the bottom, then go to the last record
  */
  IF :System.Last_Record <> 'TRUE' THEN
    Last_Record;
  ELSE
    First_Record;
  END IF;
END;
```

FORM_FAILUREビルトイン

説明

現行のランフォーム・セッションにおいて、最後に実行したアクションの結果を示す値を返します。

結果	戻り値
成功	FALSE
失敗	TRUE
致命的エラー	FALSE

現行のランフォーム・セッション中に、アクションを実行しなかった場合には、FORM_FAILUREを使用して、FALSEを返します。

FORM_FAILUREを使用して、ビルトイン結果をテストし、トリガーで処理を続行するかどうかを判断します。正確なテスト結果を得るには、正確なテスト結果を得るには、アクションの実行直後にテストを行う必要があります。つまり、テスト前にもう1つのアクションが発生するといけないからです。

注意: "もう一つのアクション"は、ビルトインおよびPL/SQL割当て文を含みます。もう1つのアクションが発生すると、FORM_FAILUREはテスト中のビルトイン・ステータスではなく、それより後で実行した別のアクションのステータスを示すことがあります。もっと正確なテクニックは、たとえば、COMMIT_FORMを実行するときに、動作終了後、SYSTEM.FORM_STATUS変数が'QUERY'に設定されるようにすることです。

構文

```
FUNCTION FORM_FAILURE;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

BOOLEAN

問合せ入力モード

可

パラメータ

なし

FORM_FAILUREの例

```
/*
** Built-in: FORM_FAILURE
** Example:  Determine if the most recently executed built-in
**           failed.
*/
BEGIN
  GO_BLOCK('Success_Factor');
  /*
  ** If some validation failed and prevented us from leaving
  ** the current block, then stop executing this trigger.
  **
  ** Generally it is recommended to test
  **    IF NOT Form_Success THEN ...
  ** Rather than explicitly testing for FORM_FAILURE
  */
  IF Form_Failure THEN
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

FORM_FATALビルトイン

説明

現行のランフォーム・セッションにおいて、最後に実行したアクションの結果を返します。

結果	戻り値

成功	FALSE
失敗	FALSE
致命的エラー	TRUE

FORM_FATALを使用して、ビルトイン結果をテストし、トリガーで処理を続行するかどうかを判断します。正確なテスト結果を得るには、正確なテスト結果を得るには、アクションの実行直後にテストを行う必要があります。つまり、テスト前にもう1つのアクションが発生するといけないからです。

注意: "もう一つのアクション"は、ビルトインおよびPL/SQL割当て文を含みます。もう1つのアクションが発生すると、FORM_FATALはテスト中のビルトイン・ステータスではなく、それより後で実行した別のアクションのステータスを示すことがあります。もっと正確なテクニックは、たとえば、COMMIT_FORMを実行するときに、動作終了後、SYSTEM.FORM_STATUS変数が'QUERY'に設定されるようにすることです。

構文

```
FUNCTION FORM_FATAL;
```

ビルトイン・タイプ

制限なしファンクション

戻り値の型:

BOOLEAN

問合せ入力モード

可

パラメータ

なし

FORM_FATALの例

```
/*
** Built-in: FORM_FATAL
** Example:  Check whether the most-recently executed built-in
**           had a fatal error.
*/
BEGIN
  User_Exit('Calculate_Line_Integral control.start control.stop');
  /*
  ** If the user exit code returned a fatal error, print a
  ** message and stop executing this trigger.
```

```
  **
  ** Generally it is recommended to test   **
  **    IF NOT FORM_SUCCESS THEN ...**
  ** Rather than explicitly testing for FORM_FATAL
  */

  IF Form_Fatal THEN
    Message('Cannot calculate the Line Integral due to internal
            error.');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

FORM_SUCCESSビルトイン

説明

現行のランフォーム・セッションにおいて、最後に実行したアクションの結果を返します。

結果	戻り値
成功	TRUE
失敗	FALSE
致命的エラー	FALSE

構文

```
FUNCTION FORM_SUCCESS;
```

ビルトイン・タイプ

制限なしファンクション

戻り値の型:

BOOLEAN

問合せ入力モード

可

パラメータ

なし

使用上の注意

- FORM_SUCCESSを使用して、ビルトイン結果をテストし、トリガーで処理を続行するかどうかを判断します。正確なテスト結果を得るには、正確なテスト結果を得るには、アクションの実行直後にテストを行う必要があります。つまり、テスト前にもう1つのアクションが発生するといけないからです。"もう一つのアクション"は、ビルトインおよびPL/SQL割当て文を含みます。もう1つのアクションが発生すると、FORM_SUCCESSはテスト中のビルトイン・ステータスではなく、それより後で実行した別のアクションのステータスを示すことがあります。

- COMMIT_FORMまたはPOSTビルトインが成功したかどうかをテストするときは、FORM_SUCCESSを使用しないでください。COMMIT_FORMによってそれ以外の多数のトリガーが起動される可能性があるため、FORM_SUCCESSを評価するときには、COMMIT_FORMの状態が反映されず、それ以外のより最近に実行されたビルトインの状態が反映されることがあります。より正確な技法としては、操作の完了後にSYSTEM.FORM_STATUS変数が「QUERY」に設定されているかをチェックする方法があります。

- Microsoft Windows NTでは、HOSTビルトインを使用して16ビット・アプリケーションを実行するときに、FORM_SUCCESSビルトインは、そのアプリケーションが正常に実行されても失敗してもTRUEを戻します。これは、Microsoft Win32の問題です。32ビット・アプリケーションおよびOSコマンドは、正常に実行されるとTRUEを正しく戻し、失敗するとFALSEを戻します。無効なコマンドは、FALSEを戻します。

- Windows 95プラットフォーム上では、FORM_SUCCESSビルトインは失敗したHOSTコマンドに対して常にTRUEを戻します。この中には、command.comまたはOSファックションに対するコール、任意の16ビットDOSアプリケーション、GUIアプリケーションまたは無効なコマンドが含まれます。FORM_SUCCESSは、32ビット・アプリケーションが正常に実行されるとTRUEを戻し、失敗するとFALSEを戻します。

FORM_SUCCESSの例

```
/*
** Built-in: FORM_SUCCESS
** Example:   Check whether the most-recently executed built-in
**             succeeded.
*/
BEGIN
  /*
  ** Force validation to occur
  */
  Enter;
  /*
  ** If the validation succeeded, then Commit the data.
  **
```

```
  */
  IF Form_Success THEN
    Commit;
    IF :System.Form_Status <> 'QUERY' THEN
      Message('Error prevented Commit');
      RAISE Form_Trigger_Failure;
    END IF;
  END IF;
END;
```

FORMS_DDLビルトイン

説明

実行時にサーバ側のPL/SQLおよびデータ定義言語（DDL）など、動的SQL文を発行します。

注意: データ定義言語を操作すると、必ず暗黙のCOMMITが発行され、Form Builderが保留中の変更を処理しないうちに現行のトランザクションを終了してしまいます。

構文

```
FUNCTION FORMS_DDL
  (statement  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

パラメータ

statement	**32K以内で、次の任意の文字列式を指定します。** **■ リテラル** **■ 動的に作成したPL/SQLコード・ブロックのテキストを表す式や変数** **■ データ操作言語（DML）文** **■ データ定義言語(DDL)文**

使用上の注意

保留中の変更をすべてコミット（またはロールバック）してから、FORMS_DDLコマンドを発行してください。データ定義言語を操作すると、必ず暗黙のCOMMITが発行され、Form Builderが

保留中の変更を処理したり、取得していたロックを解除したりしないうちに現行のトランザクションを終了してしまいます。

提供されているストアド・プロシージャの中には、それらの論理の一部として、COMMITやROLLBACKコマンドを発行するものがあります。フォームにおける保留中の変更はすべて、必ずコミットまたはロールバックしてから、それらのビルトインをコールしてください。SYSTEM.FORM_STATUS変数を使って、現行のフォーム内に保留中の変更があるかどうかをチェックしてから、FORMS_DDLコマンドを発行します。（例4を参照。）

FORMS_DDLで、有効なPL/SQLブロックを実行する場合:

- 必要に応じて、セミコロンを使用する。
- PL/SQLブロックを有効なBEGIN/ENDブロック構造体で囲む。
- PL/SQLブロックの終点にスラッシュを使用しない。
- 許可されていても、改行は必要ない。

FORMS_DDLを使って、単一のデータ操作言語文か、データ定義言語文を実行する場合:

- 無効文字エラーが発生しないように、後続のセミコロンを削除する。

FORMS_DDLを使って発行した文が、正しく実行されたかどうかをチェックするときは、FORM_SUCCESSまたはFORM_FAILUREブール関数を使用します。正しく実行されなかった場合は、DBMS_ERROR_CODEとDBMS_ERROR_TEXTを使って、エラーコードとエラーテキストをチェックします。DBMS_ERROR_CODEとDBMS_ERROR_TEXTの値は、実行が成功した後は自動リセットできません。したがって、FORM_SUCCESSか、FORM_FAILUREをコールして、エラーを検出した後しか検査する必要はありません。

FORMS_DDLの制限

FORMS_DDLに渡す文には文字列におけるバインド変数の「参照元」は含められません。しかし、FORMS_DDLに結果を渡す前に、バインド変数の値を連結すれば、文字列に挿入できます。たとえば、この文は無効です:

```
Forms_DDL ('Begin Update_Employee (:emp.empno); End;');
```

しかし、この文は有効なので、望みの結果が得られます:

```
Forms_DDL ('Begin Update_Employee ('||TO_CHAR(:emp.empno)
     ||');End;');
```

しかし、ストアード・プロシージャを直接コールすることもできます。それには、emp.empnoの値が異なる複数の実行に対して、Oracle8の共用SQL領域を使用します。

```
Update_Employee (:emp.empno);
```

FORMS_DDLで実行したSQL文とPL/SQLブロックでは、結果をForm Builderに直接返せません。（例4を参照。）

さらに、データ定義言語の中には、 Form Builderが操作対象のオブジェクトに対してカーソルをオープンにしておくと、表やデータベース・リンクが削除されるなど、FORMS_DDLで操作できないものがあります。

FORMS_DDLの例

例1

```
/*
** Built-in:  FORMS_DDL
** Example:   The expression can be a string literal.
*/
BEGIN
  Forms_DDL('create table temp(n NUMBER)');
  IF NOT Form_Success THEN
    Message ('Table Creation Failed');
  ELSE
    Message ('Table Created');
  END IF;
END;
```

例2

```
/*
** Built-in:  FORMS_DDL
** Example:   The string can be an expression or variable.
**            Create a table with n Number columns.
**            TEMP(COL1, COL2, ..., COLn).
*/
PROCEDURE Create_N_Column_Number_Table (n NUMBER) IS
  my_stmt VARCHAR2(2000);
BEGIN
  my_stmt := 'create table tmp(COL1 NUMBER';
  FOR I in 2..N LOOP
    my_stmt := my_stmt||',COL'||TO_CHAR(i)||' NUMBER';
  END LOOP;
  my_stmt := my_stmt||')';
  /*
  ** Now, create the table...
  */
  Forms_DDL(my_stmt);
  IF NOT Form_Success THEN
    Message ('Table Creation Failed');
  ELSE
    Message ('Table Created');
  END IF;
END;
```

例3

```
/*
** Built-in:  FORMS_DDL
** Example:   The statement parameter can be a block
**            of dynamically created PL/SQL code.
*/
DECLARE
  procname VARCHAR2(30);
BEGIN
  IF :global.flag = 'TRUE' THEN
    procname := 'Assign_New_Employer';
  ELSE
    procname := 'Update_New_Employer';
  END IF;
  Forms_DDL('Begin '|| procname ||'; End;');
  IF NOT Form_Success THEN
    Message ('Employee Maintenance Failed');
  ELSE
    Message ('Employee Maintenance Successful');
  END IF;
END;
```

例4

```
/*
** Built-in:  FORMS_DDL
** Example:  Issue the SQL statement passed in as an argument,
**           and return a number representing the outcome of
**           executing the SQL statement.
**           A result of zero represents success.
*/
FUNCTION Do_Sql (stmt VARCHAR2, check_for_locks BOOLEAN := TRUE)
RETURN NUMBER
IS
  SQL_SUCCESS CONSTANT NUMBER := 0;
BEGIN
  IF stmt IS NULL THEN
    Message ('DO_SQL: Passed a null statement.');
    RETURN SQL_SUCCESS;
  END IF;
  IF Check_For_Locks AND :System.Form_Status = 'CHANGED' THEN
    Message ('DO_SQL: Form has outstanding locks pending.');
    RETURN SQL_SUCCESS;
  END IF;
    Forms_DDL(stmt);
  IF Form_Success THEN
    RETURN SQL_SUCCESS;
  ELSE
```

```
    RETURN Dbms_Error_Code;
  END IF;
END;
```

GENERATE_SEQUENCE_NUMBERビルトイン

説明

レコード作成時に、重複しない順序番号を生成するForm Builderのデフォルトの処理を起動します。データベース内に順序オブジェクトが定義されている場合は、「初期値」プロパティをSEQUENCE.my_seq.NEXTVALにすれば、順序オブジェクトを項目の初期値として参照することができます。デフォルトでは、レコードが作成されると、Form Builderによって順序の次の値が取得されます。Oracle以外のデータ・ソースに接続している場合は、On-Sequence-Numberトリガーにこのビルトインへのコールを含めることができます。

構文

```
PROCEDURE GENERATE_SEQUENCE_NUMBER;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

GENERATE_SEQUENCE_NUMBERの制限事項

On-Sequence-Number内でのみ有効です。

GENERATE_SEQUENCE_NUMBERの例

```
/*
** Built-in: GENERATE_SEQUENCE_NUMBER
** Example:  Perform Form Builder standard sequence number
**           processing based on a global flag setup at
**           startup by the form, perhaps based on a
**           parameter.
** Trigger:  On-Sequence-Number
*/
BEGIN
  /*
  ** Check the global flag we setup at form startup
```

```
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    User_Exit('my_seqnum seq=EMPNO_SEQ');
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Generate_Sequence_Number;
  END IF;
END;
```

GET_APPLICATION_PROPERTYビルトイン

説明

現行のForm Builderアプリケーションに関する情報を返します。このビルトインは、取り出す値ごとコールする必要があります。

構文

```
FUNCTION GET_APPLICATION_PROPERTY
  (property  NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

property（続き）	アプリケーションに関する情報を返す次の定数をどれか1つ指定します。 APPLICATION_INSTANCE インスタンス・ハンドルのポインタ値が返されます。Microsoft Windowsプラットフォームにのみ適用されます。他のプラットフォームでは、NULLが返されます。 BUILTIN_DATE_FORMAT Builtin日付書式マスク（Builtin_Date_Formatプロパティに保持されている）の現行値が返されます。 CALLING_FORM 現行フォームがCALL_FORMビルトインによって起動されている場合、コール側フォームの名前が返されます。現行フォームがコール先フォームでない場合、Form BuilderはNULLを返します。 CONNECT_STRING 現行のオペレータのデータベース接続文字列が返されます。現行のオペレータが接続文字列を指定していない場合は、NULLが返されます。 CURRENT_FORM 現在実行しているフォームの.FMXファイル名が返されます。 CURRENT_FORM_NAME フォーム・モジュールの「名前」プロパティに示される現行フォームの名前が返されます。 CURSOR_STYLE 現行のカーソル・スタイル・プロパティの名前が返されます。データ型VARCHAR2の有効な戻り値は、BUSY、CROSSHAIR、DEFAULT、HELPおよびINSERTIONです。 DATASOURCE 現在使用しているデータベースの名前が返されます。有効な戻り値は、NULL、ORACLE、DB2、NONSTOP、TERADATA、NCR/3600/NCR/3700およびSQLSERVERです。このコールでは、On-Logonトリガーによって確立された接続を除き、Form Builderによって確立された接続のデータベース名のみが返されます。 DATE_FORMAT_COMPATIBILITY_MODE このプロパティに含まれる互換性設定が返されます。 DISPLAY_HEIGHT 画面の高さが返されます。高さの単位は、そのフォーム・モジュールに対して「座標システム」プロパティをどのように定義したかによって異なります。 DISPLAY_WIDTH 画面の幅が返されます。高さの単位は、そのフォーム・モジュールに対して「座標システム」プロパティをどのように定義したかによって異なります。 ERROR_DATE/DATETIME_FORMAT エラー日付または日時書式マスク（FORMSnn_Error_Date/Datetime_Format環境変数で設定される）の現行値が返されます。 FLAG_USER_VALUE_TOO_LONG このプロパティの現行値(「TRUE」または「FALSE」)が返されます。この値によって、「最大長」プロパティを超えるユーザー入力値の切捨てが制御されます。

property（続き）	OPERATING_SYSTEM	現在使用しているオペレーティング・システムの名前が返されます。有効な戻り値は、MSWINDOWS、MSWINDOWS32、WIN32COMMON、UNIX、SunOS、OUTPUT_DATE/DATETIME_FORMAT出力日付または日時書式マスク（FORMSnn_Output_Date/Datetime_Format環境変数で設定される）の現行値が返されます。
	PASSWORD	現行オペレータのパスワードが返されます。
	PLSQL_DATE_FORMAT	PLSQL日付書式マスク（PLSQL_Date_Formatプロパティに保持されている）の現行値が返されます。
	RUNTIME_COMPATIBILITY_MODE	このプロパティに含まれる互換性設定が返されます。MACINTOSH、VMSおよびHP-UXです。
	SAVEPOINT_NAME	Form Builderから発行された最後のセーブポイントの名前が返されます。この定数は、On-Savepointトリガーまたは On-Rollbackトリガーからコールした場合にのみ有効です。これは本来、トランザクション・トリガーを使用してOracle以外のデータ・ソースにアクセスする開発者向けに用意された定数です。
	TIMER_NAME	最新の期限切れタイマーの名前が返されます。タイマーがない場合は、NULLが返されます
	USER_DATE/DATETIME_FORMAT	ユーザー日付または日時書式マスク（FORMSnn_User_Date/Datetime_Format環境変数で設定される）の現行値が返されます。
	USER_INTERFACE	現在使用しているユーザー・インタフェースの名前が返されます。有効な戻り値は、MOTIF、MACINTOSH、MSWINDOWS、MSWINDOWS32、WIN32COMMON、WEB、PM、CHARMODE、BLOCKMODE、XおよびUNKNOWNです。
	USER_NLS_CHARACTER_SET	フォーム・オペレータ用に定義されたUSER_NLS_LANG環境変数のうち、文字セットに関する部分の現行の値のみが返されます。USER_NLS_LANGが明示的に設定されない場合は、デフォルトでNLS_LANGが設定されます。
	USER_NLS_LANG	各国語サポートの目的でフォーム・オペレータ用に定義されたUSER_NLS_LANG環境変数の現行の値が、現在の設定値で返されます。USER_NLS_LANGが明示的に設定されない場合、NLS_LANGにデフォルト設定されます。USER_NLS_LANGは、USER_NLS_LANGUAGE、USER_NLS_TERRITORYおよびUSER_NLS_CHARACTER_SETを連結したものと同じです。
	USER_NLS_LANGUAGE	フォーム・オペレータ用に定義されたUSER_NLS_LANG環境変数のうち、言語に関する部分の現行の値のみが返されます。USER_NLS_LANGが明示的に設定されない場合は、デフォルトでNLS_LANGが設定されます。

property（続き）	USER_NLS_TERRITORY	フォーム・オペレータ用に定義されたUSER_NLS_LANG環境変数のうち、地域に関する部分の現行の値のみが返されます。USER_NLS_LANGが明示的に設定されない場合は、デフォルトでNLS_LANGが設定されます。
	USERNAME	現行オペレータの名前が返されます。

注意: ユーザーがOPS$アカウントを使用して接続する場合GET_APPLICATION_PROPERTY(USERNAME)は実際のユーザー名を返しません。この場合、GET_APPLICATION_PROPERTY(CONNECT_STRING)を使用して、ユーザー名情報を取り出す必要があります。

使用上の注意

接続文字列の追加も含めた完全なログインを要求するには、「Username」プロパティ、「Password」プロパティおよび「Connect_String」プロパティを指定します。たとえば、ユーザーが次のような接続文字列を指定して、Microsoft Windowsのランフォーム・セッションを起動した場合、次のようになります。

```
ifrun60 my_form scott/tiger@corpDB1
```

GET_APPLICATION_PROPERTY(USERNAME)をコールすると、Form Builderによって次の文字列が返されます。

```
scott
```

GET_APPLICATION_PROPERTY(PASSWORD)をコールすると、Form Builderによって次の文字列が返されます。

```
tiger
```

GET_APPLICATION_PROPERTY(CONNECT_STRING)をコールすると、Form Builderによって次の文字列が返されます。

```
corpDB1
```

GET_APPLICATION_PROPERTYの制限事項

最近実行されたタイマーのタイマー名を取り出すには、When-Timer-Expiredトリガー内からGET_APPLICATION_PROPERTYへのコールを起動する必要があります。それ以外の場合、ビルトインの結果は未定義です。

GET_APPLICATION_PROPERTYの例

例1

```
/*
** Built-in: GET_APPLICATION_PROPERTY
** Example:  Determine the name of the timer that just
```

```
**           expired, and based on the username perform a
**           task.
** Trigger:  When-Timer-Expired
*/
DECLARE
  tm_name  VARCHAR2(40);
BEGIN
  tm_name  := Get_Application_Property(TIMER_NAME);

  IF tm_name = 'MY_ONCE_EVERY_FIVE_MINUTES_TIMER' THEN

  :control.onscreen_clock := SYSDATE;

  ELSIF tm_name = 'MY_ONCE_PER_HOUR_TIMER' THEN

    Go_Block('connected_users');
    Execute_Query;

  END IF;
END;
```

例2

```
/*
** Built-in: GET_APPLICATION_PROPERTY
** Example:  Capture the username and password of the
**           currently logged-on user, for use in calling
**           another Tool.
*/
PROCEDURE Get_Connect_Info( the_username IN OUT VARCHAR2,
              the_password IN OUT VARCHAR2,
              the_connect  IN OUT VARCHAR2) IS
BEGIN
  the_username := Get_Application_Property(USERNAME);
  the_password := Get_Application_Property(PASSWORD);
  the_connect   := Get_Application_Property(CONNECT_STRING);
END;
```

GET_BLOCK_PROPERTYビルトイン

説明

特定のブロックに関する情報を返します。取り出すプロパティの値ごとに、このビルトインをコールする必要があります。

構文

```
FUNCTION GET_BLOCK_PROPERTY
  (block_id  Block,
   property  NUMBER);
FUNCTION GET_BLOCK_PROPERTY
  (block_name  VARCHAR2,
   property    NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

block_id	ブロック作成時にForm Builderによってそのブロックに割り当てられた一意のID。データ型はBLOCKです。
block_name	ブロック作成時にユーザーがそのブロックに付けた名前。データ型はVARCHAR2です。
property	指定したブロックに関する情報を返す次の定数をどれか1つ指定します。 ALL_RECORDS　問合せ基準に一致するすべてのレコードを、問合せの実行時にデータ・ブロック内にフェッチするかどうかを指定します。 BLOCKSCROLLBAR_X_POS　ブロックのスクロール・バーのx位置を、「座標システム」フォーム・プロパティによって示されるフォーム座標単位で指定された数として返します。

property（続き）	BLOCKSCROLLBAR_Y_POS ブロックのスクロール・バーのy位置を、「座標システム」フォーム・プロパティによって示されるフォーム座標単位で指定された数として返します。 COLUMN_SECURITY 「列セキュリティの強化」が「はい」に設定されていればVARCHAR2値TRUEが返され、「列セキュリティの強化」 が「いいえ」に設定されていればVARCHAR2文字列FALSEが返されます。 COORDINATION_STATUS マスター/ディテールのブロック・リレーションでディテール・ブロックにあたるブロックの場合、このプロパティでは、このブロックのマスター・ブロックに対する調整ステータスが指定されます。そのブロックがすべてのマスター・ブロックと調整されていれば、VARCHAR2値COORDINATEDが返されます。すべてのマスター・ブロックと調整されていなければ、VARCHAR2値NON_COORDINATEDが返されます。ディテール・ブロックのステータスは、レコードがディテール・ブロックにフェッチされた直後に COORDINATEDになります。また、別のレコードがマスター・ブロックの現レコードになると、ディテール・ブロックのステータスが「NON_COORDINATED」に戻ります。 CURRENT_RECORD 現レコードの番号が返されます。 CURRENT_RECORD_ATTRIBUTE 指定したブロックのユーザー命名可視属性の名前がVARCHAR2型で返されます。 CURRENT_ROW_BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 CURRENT_ROW_FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 CURRENT_ROW_FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 CURRENT_ROW_FONT_SIZE ポイント数で指定されるフォントのサイズ。 CURRENT_ROW_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 CURRENT_ROW_FONT_STYLE フォントのスタイル。

property（続き）	CURRENT_ROW_FONT_WEIGHT フォントの太さ。 CURRENT_ROW_FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 CURRENT_ROW_WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 DEFAULT_WHERE ブロックの「WHERE句」プロパティの現在の設定を示す、ブロックに対するWHERE句が返されます。 DELETE_ALLOWED ブロックの「削除可」プロパティが「はい」であればVARCHAR2値TRUEが返され、「いいえ」であればFALSEが返されます。このプロパティによって、オペレータまたはアプリケーションがブロックのレコードを削除できるかどうかが決定されます。 DML_DATA_TARGET_NAME ブロックのDMLデータ・ソースの名前がVARCHAR2型で返されます。 DML_DATA_TARGET_TYPE 「DMLデータ宛先タイプ」プロパティの現在の設定を示すVARCHAR2値が返されます。このプロパティの戻り値は、NONE、TABLE、STORED PROCEDUREまたはTRANSACTIONAL TRIGGERです。 ENFORCE_PRIMARY_KEY 「主キーの強化」プロパティが「はい」に設定されていれば、VARCHAR2値TRUEが返されます。「主キーの強化」プロパティが「いいえ」に設定されていれば、VARCHAR2値FALSEを返します。 ENTERABLE ブロックが入力可能である場合、つまり、ブロック内のいずれかの項目の「変更可能」プロパティと「キーボードで移動可能」プロパティが 「はい」に設定されていれば、VARCHAR2値TRUEが返されます。ブロックが入力可能でなければ、VARCHAR2値FALSEを返します。 FIRST_DETAIL_RELATION 指定したブロックがディテールである最初のリレーションの名前がVARCHAR2型で返されます。そのようなリレーションが存在しなければ、NULLが返されます。 FIRST_ITEM 指定したブロックの最初の項目の名前がVARCHAR2型で返されます。

property（続き）	FIRST_MASTER_RELATION 指定したブロックがマスターである最初のリレー　ションの名前がVARCHAR2型で返されます。そのようなリレーションが存在しなければ、NULLが返されます。 INSERT_ALLOWED　ブロックの「挿入可」プロパティが「はい」であれば、VARCHAR2値TRUEが返され、「いいえ」であればFALSEが返されます。このプロパティによって、オペレータまたはアプリケーションがブロックにレコードを挿入できるかどうかが決定されます。 KEY MODE　ブロックの「キー・モード」プロパティの現在の設定を示すVARCHAR2値が返されます。このプロパティの戻り値は、UNIQUE_KEY、UPDATEABLE_PRIMARY_KEYまたはNON_UPDATEABLE_PRIMARY_KEYです。 LAST_ITEM　指定したブロックの最後の項目の名前が返されます。 LAST_QUERY　指定したブロックの最後の問合せのSQL文が返されます。 LOCKING_MODE　実表項目の変更時、行が直ちにロックされる場合は、VARCHAR2値IMMEDIATEが返され、行がコミットの直前にロックされる場合は、VARCHAR2値DELAYEDが返されます。 MAX_QUERY_TIME　「最長問合せ時間」プロパティの現在の設定を示すVARCHAR2値が返されます。このプロパティによって、問合せの経過時間がこのプロパティの値を超えた場合に、オペレータが問合せを中止できるかどうかが決定されます。 MAX_RECORDS_FETCHED フェッチできるレコードの最大数を示す数値が返されます。このプロパティは、「全レコード問合せ」プロパティが「はい」に設定されている場合にのみ有効です。 NAVIGATION_STYLE 「ナビゲーション形式」ブロック・プロパティの現在の設定を示す、SAME_RECORD、CHANGE_RECORD、またはCHANGE_BLOCKのいずれかのVARCHAR2値が返されます。 NEXTBLOCK　次のブロックの名前が返されます。指定されたブロックがフォームの最後のブロックの場合は、NULLが返されます。「次のナビゲーション・データ・ブロック」ブロック・プロパティに設定しても、NEXBLOCKの値に影響しないことに注意してください。 NEXT_NAVIGATION_BLOCK ブロックの次のナビゲーションブロックの名前がVARCHAR2型で返されます。デフォルトでは、次のナビゲーション・ブロックはオブジェクト・ナビゲータで定義したブロック順序の次のブロックです。ただし、「次のナビゲーション・データ・ブロック」を設定すれば、このデフォルトのナビゲーション順序を上書きできます。 OPTIMIZER_HINT　問合せの組立て時にForm BuilderからRDBMSオプティマイザへ渡されるヒントがVARCHAR2文字列の形式で返されます。

property（続き）	ORDER_BY	ブロックに影響するデフォルトのORDER BY句が、「ORDER BY句」ブロック・プロパティの現在の設定が示すとおりに返されます。
	PRECOMPUTE_SUMMARIES(現在、定義されていません。)	
	PREVIOUSBLOCK	フォーム内で1つ前の順序番号を持つブロックの名前が、Object Navigatorでブロックの順序として定義されたとおりに返されます。指定したブロックがフォームの最初のブロックの場合は、NULLが返されます。ブロックの「前のナビゲーション・データ・ブロック」プロパティの設定は、PREVIOUSBLOCKの値に影響しないことに注意してください。
	PREVIOUS_NAVIGATION_BLOCK	ブロックの前のナビゲーション・ブロックの名前がVARCHAR2型で返されます。デフォルトでは、前のナビゲーション・ブロックは、オブジェクト・ナビゲータで定義したブロックの順序番号が次に小さいブロックです。ただし、「前のナビゲーション・データ・ブロック」ブロック・プロパティを設定すれば、デフォルトのブロック・ナビゲータ順序を上書きできます。
	QUERY_ALLOWED	「問合せ可」ブロック・プロパティが「はい」であれば、VARCHAR2値TRUEが返され、「いいえ」であればFALSEが返されます。このプロパティによって、オペレータまたはアプリケーションがブロック内のレコードを問合せできるかどうかが決定されます。
	QUERY_DATA_SOURCE_NAME	ブロックの問合せデータ・ソースの名前がVARCHAR2型で返されます。
	QUERY_DATA_SOURCE_TYPE	「問合せデータ・ソース・タイプ」プロパティの現在の設定を示すVARCHAR2値が返されます。このプロパティの戻り値は、NONE、TABLE、STORED PROCEDURE、TRANSACTIONAL TRIGGERまたはSUB-QUERYです。
	QUERY_HITS	COUNT_QUERY操作によって識別されたレコードの数を示すVARCHAR2値が返されます。問合せからレコードが取り出されている最中に「QUERY_HITS」の値を調べる場合は、この値は取り出されたレコード数を表します。
	QUERY_OPTIONS	VARCHAR2値VIEW、FOR_UPDATE、COUNT_QUERYまたはオプションがない場合はNULLが返されます。デフォルトの処理を迂回しなかった場合は、Form Builderによってデフォルトで実行される問合せ操作のタイプをユーザー・イグジットで識別するときに、トランザクション・トリガー内でこのパラメータを指定してGET_BLOCK_PROPERTYコールすることができます。

property（続き）	
	RECORDS_DISPLAYED 指定したブロックが表示できるレコードの数が返されます。このプロパティは、「表示レコード数」ブロック・プロパティに相当します。
	RECORDS_TO_FETCH On-Fetchトリガーがフェッチし、問い合わせられたレコードとして作成されることをForm Builderが予期するレコードの数が返されます。
	STATUS ブロックに新規レコードのみが含まれている場合は、VARCHAR2値NEWが、ブロックに1つ以上の変更されたレコードが含まれている場合は、CHANGEDが、ブロックにデータベースから検索された有効なレコードのみが含まれている場合は、QUERYが返されます。
	TOP_RECORD 指定したブロック内にある先頭の可視レコードのレコード番号が返されます。
	UPDATE_ALLOWED 「更新可」ブロック・プロパティが「はい」であれば、VARCHAR2値TRUEが返され、「いいえ」であればFALSEが返されます。このプロパティによって、オペレータとアプリケーションがブロック内のレコードを更新できるかどうかが決定されます。
	UPDATE_CHANGED_COLUMNS オペレータが更新した列のみがデータベースに送信されることを指定します。「変更列のみ更新」プロパティが「いいえ」に設定されていれば、更新されたかどうかに関係なくすべての列が送信されます。この場合、特にブロックにLONGデータ型が含まれていれている場合は、ネットワークの通信量が大幅に増加します。

GET_BLOCK_PROPERTYの例

```
/*
** Built-in: GET_BLOCK_PROPERTY
** Example:   Return the screen line of the current record in
**            a multi-record block.Could be used to
**            dynamically position LOV to a place on the
**            screen above or below the current line so as to
**            not obscure the current record in question.
*/
FUNCTION Current_Screen_Line
RETURN NUMBER IS
  cur_blk VARCHAR2(40) := :System.Cursor_Block;
  cur_rec NUMBER;
  top_rec NUMBER;
  itm_lin NUMBER;
  cur_lin NUMBER;
  bk_id   Block;
```

```
BEGIN
  /*
  ** Get the block id since we'll be doing multiple
  ** Get_Block_Property operations for the same block
  */
  bk_id := Find_Block( cur_blk );
  /*
  ** Determine the (1) Current Record the cursor is in,
  **             (2) Current Record which is visible at the
  **                 first (top) line of the multirecord
  **                 block.
  */
  cur_rec := Get_Block_Property( bk_id, CURRENT_RECORD);
  top_rec := Get_Block_Property( bk_id, TOP_RECORD);
  /*
  ** Determine the position on the screen the first field in
  ** the multirecord block
  */
  itm_lin := Get_Item_Property( Get_Block_Property
                 (bk_id,FIRST_ITEM),Y_POS);
  /*
  ** Add the difference between the current record and the
  ** top record visible in the block to the screen position
  ** of the first item in the block to get the screen
  ** position of the current record:
  */
  cur_lin := itm_lin + (cur_rec - top_rec);
  RETURN cur_lin;
END;
```

GET_CANVAS_PROPERTYビルトイン

説明

指定したキャンバスの指定したキャンバス・プロパティが返されます。 .

構文

```
FUNCTION GET_CANVAS_PROPERTY
  (canvas_id  Canvas,
   property   NUMBER);
```

```
FUNCTION GET_CANVAS_PROPERTY
  (canvas_name  VARCHAR2,
   property     NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

canvas_id	キャンバス・オブジェクト作成時にForm Builderによってキャンバス・オブジェクトに割り当てられる一意のID。FIND_CANVASビルトインを使用すると、このIDをデータ型CANVASの変数に返すことができます。
canvas_name	キャンバス・オブジェクトの定義時に付けた名前。
property	指定したキャンバスの値を取得するためのプロパティ。値を返す次の定数をどれか1つ指定します。 BACKGROUND_COLOR　オブジェクトのバックグラウンド領域のカラー。 FILL_PATTERN　オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME　オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE　ポイント数で指定されるフォントのサイズ。 FONT_SPACING　フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE　フォントのスタイル。 FONT_WEIGHT　フォントの太さ。 FOREGROUND_COLOR　オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 HEIGHT　「座標システム」フォーム・プロパティによって示される座標単位で、指定したキャンバスの高さが返されます。

property（続き）	TAB_PAGE_X_OFFSET	タブ・キャンバスの左端からタブ・ページの左端までの距離が返されます。返される値は、フォーム座標システムの単位がピクセル、センチメートル、インチ、またはポイントのどれであるかによって異なります。
	TAB_PAGE_Y_OFFSET	タブ・キャンバスの上端からタブ・ページの上端までの距離が返されます。返される値は、フォーム座標システムの単位がピクセル、センチメートル、インチ、またはポイントのどれであるかによって異なります。
	TOPMOST_TAB_PAGE	現在、指定したタブ・キャンバスの一番上に表示されているタブ・ページの名前が返されます。
	WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	WIDTH	「座標システム」フォーム・プロパティによって示されるフォーム座標の単位で、指定されたキャンバスの幅が返されます。
	VISUAL_ATTRIBUTE	現在有効な可視属性の名前が返されます。キャンバスに名前付き可視属性が割り当てられていない場合は、カスタム可視属性であればCUSTOMが、デフォルトの可視属性であればDEFAULTが返されます。

GET_CANVAS_PROPERTYの例

```
/*
** Built-in: GET_CANVAS_PROPERTY
** Example:   Can get the width/height of the canvas.
*/
DECLARE
  the_width  NUMBER;
  the_height NUMBER;
  cn_id      CANVAS;
BEGIN
  cn_id      := FIND_CANVAS('my_canvas_1');
  the_width  := GET_CANVAS_PROPERTY(cn_id, WIDTH);
  the_height := GET_CANVAS_PROPERTY(cn_id,HEIGHT);
END;
```

GET_CUSTOM_PROPERTYビルトイン

説明

プラグイン可能Javaコンポーネント内のユーザー定義プロパティの値を取得します。

構文

このビルトインは、文字列、数値またはブール値のデータからなるVARCHAR2を戻します。

```
GET_CUSTOM_PROPERTY
  (item,
   row-number,
   prop-name);
```

ビルトイン・タイプ

制限なしプロシージャ

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

item	ターゲットのプラグイン可能Javaコンポーネントに関連付けられた項目の名前またはID。名前の形式は、VARCHAR2リテラルまたは名前の値が設定された変数です。
row-number	取得対象項目のインスタンスの行番号。（インスタンスの行番号は1から始まります。）
prop-name	取得対象Javaコンポーネントのプロパティ名

使用上の注意

- プラグイン可能Javaコンポーネントでは、各カスタム・プロパティのタイプは、ID.registerPropertyを使用して作成されるIDクラスの1つのインスタンスによって表す必要があります。
- フォームで実行される各Get_Custom_Propertyビルトインについて、JavaコンポーネントのgetPropertyメソッドがコールされます。
- 項目の名前は、Find_Item('Item_Name')または単に'Item_Name'を介して取得できます。

GET_FILE_NAMEビルトイン

説明

ユーザーが既存のファイルを選択したり新規ファイルを指定したりできる、標準のファイル・オープン・ダイアログ・ボックスが表示されます。

構文

```
FUNCTION GET_FILE_NAME
   (directory_name  VARCHAR2,
   file_name        VARCHAR2,
   file_filter      VARCHAR2,
   message          VARCHAR2,
   dialog_type      NUMBER,
   select_file      BOOLEAN;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

directory_name	オープンするファイルが含まれるディレクトリの名前を指定します。デフォルト値はNULLです。directory_nameがNULLの場合、次に表示されるダイアログでは、最後に表示したディレクトリがオープンされます。
file_name	オープンするファイルの名前を指定します。デフォルト値はNULLです。
file_filter	特定のファイルのみが表示されるように指定します。デフォルト値はNULLです。ファイル・フィルタはさまざまな形式をとり、現在のmotifおよび文字モードのプラットフォームでは無視されます。Windowsでは、"Writeファイル (*.WRI)|*.WRI|"の形式をとります。NULLが指定されればデフォルトで“ すべてのファイル(*.*)|*.*| ”になります。Macintoshでは、現在この属性は"Text"の形式をとります。
message	選択されているファイルのタイプを指定します。デフォルト値はNULLです。
dialog_type	ダイアログがOPEN_FILEまたはSAVE_FILEのどちらを示すかを指定します。デフォルト値はOPEN_FILEです。

select_file	ユーザーがファイルまたはディレクトリのどちらを選択しているかを指定します。デフォルト値はTRUEです。dialog_typeがSAVE_FILEに設定されている場合、select_fileは内部的にTRUEに設定されます。

GET_FILE_NAMEの例

```
/*
** Built-in: GET_FILE_NAME
** Example:  Can get an image of type TIFF.
*/
DECLARE
  filename VARCHAR2(256)
BEGIN
  filename := GET_FILE_NAME(File_Filter=> 'TIFF Files (*.tif)|*.tif|');
  READ_IMAGE_FILE(filename, 'TIFF', 'block5.imagefld);
END;
```

GET_FORM_PROPERTYビルトイン

説明

指定されたフォームに関する情報を返します。アプリケーションがマルチ・フォーム・アプリケーションの場合は、このビルトインをコールして、コール側フォーム、現行のフォーム、コール先フォームに関する情報を返すことができます。

構文

```
FUNCTION GET_FORM_PROPERTY
  (formmodule_id  FormModule,
   property       NUMBER);
FUNCTION GET_FORM_PROPERTY
  (formmodule_name  VARCHAR2,
   property         NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

formmodule_id	フォーム・モジュール作成時にForm Builderによってそのモジュールに割り当てられた一意のIDを指定します。FIND_FORMビルトインを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型はFormModule型です。
formmodule_name	ユーザーがフォーム・モジュールを定義したときにそのモジュールに付けた名前をデータ型VARCHAR2で指定します。
property	このビルトインに次のどの定数が適用されたかにより、フォームの特定の要素に関する情報を返します。 CHARACTER_CELL_HEIGHT 文字セルのディメンションが「座標システム」プロパティによって指定されたフォーム単位で返されます。「座標システム」が「文字セル」であれば、値はピクセルで返されます。 CHARACTER_CELL_WIDTH 文字セルのディメンションが「座標システム」プロパティによって指定されたフォーム単位で返されます。「座標システム」が「文字セル」であれば、値はピクセルで返されます。 COORDINATE_SYSTEM フォーム・モジュールで使用される座標システムを示すVARCHAR2文字列が返されます。フォームの現行の座標システムが文字セル・ベースの場合はHARACTER_CELLが返されます。フォームの現行の座標システムがポ ントの場合は、POINTSが返されます。フォームの現行の座標システムがセンチメートルの場合は、CENTIMETERSが返されます。フォームの現行の座標システムがインチの場合は、INCHESが返されます。フォームの現行の座標システムがピクセルの場合は、PIXELSが返されます。 CURRENT_RECORD_ATTRIBUTE 現在行で使用する必要があるユーザー命名可視属性の名前がVARCHAR2で返されます。 CURRENT_ROW_BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 CURRENT_ROW_FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 CURRENT_ROW_FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。

property（続き）	CURRENT_ROW_FONT_SIZE ポイント数で指定されるフォントのサイズ。 CURRENT_ROW_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 CURRENT_ROW_FONT_STYLE フォントのスタイル。 CURRENT_ROW_FONT_WEIGHT フォントの太さ。 CURRENT_ROW_FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 CURRENT_ROW_WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 CURSOR_MODE 既存のカーソルに対するコミット・アクションの予想される影響を示す設定が返されます。 DEFER_REQUIRED_ENFORCEMENT 必須フィールドの強制が、項目の妥当性チェック時からレコードの妥当性チェック時まで延期されるかどうかを示す設定が返されます。有効な戻り値は、「TRUE」、「4.5」および「FALSE」です。 DIRECTION 両方向オブジェクトのレイアウト方向が返されます。有効な戻り値はRIGHT_TO_LEFT、LEFT_TO_RIGHTです。 FILE_NAME 指定したフォームが格納されているファイルの名前が返されます。 FIRST_BLOCK 指定したフォーム内の最も小さい順序番号を持つブロックの名前が返されます。 FIRST_NAVIGATION_BLOCK Form Builderがフォームの起動時にナビゲートするブロックの名前を返します。デフォルトでは、オブジェクト・ナビゲータで定義した最初のブロックが最初のナビゲーション・ブロックになりますが、「最初のナビゲーション・データ・ブロック」ブロック・プロパティを設定すれば、フォーム起動時に別のブロックを最初のナビゲーション・ブロックに指定することができます。 FORM_NAME フォームの名前が返されます。 INTERACTION_MODE フォームの相互作用モードが返されます。有効な戻り値は、BLOCKINGまたはNONBLOCKINGです。 ISOLATION_MODE READ_COMMITTEDまたはSERIALIZABLEのいずれかの、フォームの独立モードの設定が返されます。

property（続き）	LAST_BLOCK	指定したフォーム内の最も大きな順序番号を持つブロックの名前が返されます。
	MAX_QUERY_TIME	「最長問合せ時間」プロパティの現在の設定を示すVARCHAR2値が返されます。このプロパティによって、問合せの経過時間がこのプロパティの値を超えた場合に、オペレータが問合せを中止できるかどうかが決定されます。
	MAX_RECORDS_FETCHED	フェッチできるレコードの最大数を示す数値が返されます。このプロパティは、「全レコード問合せ」プロパティが「はい」に設定されている場合にのみ有効です。
	MODULE_NLS_CHARACTER_SET	フォーム用に定義されたDEVELOPER_NLS_LANG環境変数のうち、文字セットに関する部分の現行の値のみが返されます。DEVELOPER_NLS_LANGが明示的に設定されない場合は、デフォルトでNLS_LANGに設定されます。
	MODULE_NLS_LANG	フォーム用に定義されたDEVELOPER_NLS_LANG環境変数に含まれる各国語サポートの現行の値が、完全な形式で返されます。DEVELOPER_NLS_LANGが明示的に設定されない場合は、デフォルトでNLS_LANGに設定されます。MODULE_NLS_LANGは、MODULE_NLS_LANGUAGE、MODULE_NLS_TERRITORYおよびMODULE_NLS_CHACTER_SETを連結したものに相当します。
	MODULE_NLS_LANGUAGE	フォーム用に定義されたDEVELOPER_NLS_LANG環境変数の言語に関する部分の現行の値のみが返されます。DEVELOPER_NLS_LANGが明示的に設定されない場合は、デフォルトでNLS_LANGに設定されます。
	MODULE_NLS_TERRITORY	フォーム用に定義されたDEVELOPER_NLS_LANG環境変数のうち、地域に関する部分の現行の値のみが返されます。DEVELOPER_NLS_LANGが明示的に設定されない場合は、デフォルトでNLS_LANGに設定されます。
	SAVEPOINT_MODE	データ・ソースでセーブポイントがサポートされているかどうかを示すPROPERTY_ONまたはPROPERTY_OFFが返されます。
	VALIDATION	Form Builderのデフォルトの妥当性チェックが有効であるかどうかを示すTRUEまたはFALSEが返されます。
	VALIDATION_UNIT	フォームに対する現行の妥当性チェック単位を示すVARCHAR2文字列が返されます。 ■ 現行の妥当性検査単位がフォームであれば、FORM_SCOPEが返されます。 ■ 現行の妥当性検査単位がブロックであれば、BLOCK_SCOPEが返されます。 ■ 現行の妥当性検査単位がレコードであれば、RECORD_SCOPEが返されます。 ■ 現行の妥当性検査単位が項目またはDEFAULTであれば、ITEM_SCOPEが返されます。

GET_FORM_PROPERTYの例

例1

```
/*
** Built-in:GET_FORM_PROPERTY
** Example:Determine the name of the first block in the form.
*/
DECLARE
  curform VARCHAR2(40);
  blkname VARCHAR2(40);
BEGIN
  curform := :System.Current_Form;
  blkname := Get_Form_Property(curform,FIRST_BLOCK);
END;
```

例2

```
/*
** Built-in:GET_FORM_PROPERTY
** Example:Evaluate the current setting of the
**          Validate property.
*/
BEGIN
  IF Get_Form_Property('curform', VALIDATION) = 'FALSE'
  THEN
    Message ('Form currently has Validation turned OFF');
  END IF;
END;
```

GET_GROUP_CHAR_CELLビルトイン

説明

指定された行と列から特定されたレコード・グループ・セルのVARCHAR2またはLONG値を返します。セルは行と列が交差する部分です。

構文

```
FUNCTION GET_GROUP_CHAR_CELL
  (groupcolumn_id  GroupColumn,
   row_number      NUMBER);
FUNCTION GET_GROUP_CHAR_CELL
  (groupcolumn_name  VARCHAR2,
   row_number        NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

groupcolumn_id	レコード・グループの列作成時にForm Builderによって割り当てられた一意のIDを指定します。適切な型の変数にIDを戻すには、FIND_COLUMNビルトインを使用します。このIDのデータ型はGROUPCOLUMN型です。
groupcolumn_name	レコード・グループの列を定義するときに付けたデータ型VARCHAR2の完全修飾名を指定します。recordgroup_name.groupcolumn_nameのように、レコード・グループ名およびドットが前に付けられます。列が問合せの結果定義された場合は、その列の名前は対応するデータベース列の名前と同じです。
row_number	セルの値を取り出す行を指定します。

GET_GROUP_CHAR_CELLの制限事項

パラメータrow_numberには、レコード・グループ内に存在する行番号を指定する必要があります。存在しないrow_numberを指定すると、索引範囲外エラーが発生します。

GET_GROUP_CHAR_CELLの例

```
/*
** Built-in:GET_GROUP_CHAR_CELL
** Example:Search thru names in a static record group to
**          determine if the value passed into this subprogram
**          exists in the list.Returns the row number
**          where the record was first located, or zero (0)
**          if no match was found.
*/
FUNCTION Is_Value_In_List( the_value    VARCHAR2,
            the_rg_name   VARCHAR2,
            the_rg_column VARCHAR2)
RETURN NUMBER IS
  the_Rowcount   NUMBER;
  rg_id         RecordGroup;
  gc_id         GroupColumn;
  col_val       VARCHAR2(80);
  Exit_Function  Exception;
```

```
BEGIN
  /*
  ** Determine if record group exists, and if so get its ID.
  */
  rg_id := Find_Group( the_rg_name );

  IF Id_Null(rg_id) THEN
    Message('Record Group '||the_rg_name||' does not exist.');
    RAISE Exit_Function;
  END IF;

  /*
  ** Make sure the column name specified exists in the
  ** record Group.
  */
  gc_id := Find_Column( the_rg_name||'.'||the_rg_column );

  IF Id_Null(gc_id) THEN
    Message('Column '||the_rg_column||' does not exist.');
    RAISE Exit_Function;
  END IF;
  /*
  ** Get a count of the number of records in the record
  ** group
  */
  the_Rowcount := Get_Group_Row_Count( rg_id );

  /*
  ** Loop through the records, getting the specified column's
  ** value at each iteration and comparing it to 'the_value'
  ** passed in.  Compare the values in a case insensitive
  ** manner.
  */
  FOR j IN 1..the_Rowcount LOOP
    col_val := GET_GROUP_CHAR_CELL( gc_id, j );
    /*
    ** If we find a match, stop and return the
    ** current row number.
    */
    IF UPPER(col_val) = UPPER(the_value) THEN
      RETURN j;
    END IF;
  END LOOP;
```

```
  /*
  ** If we get here, we didn't find any matches.
  */
  RAISE Exit_Function;
EXCEPTION
  WHEN Exit_Function THEN
    RETURN 0;
END;
```

GET_GROUP_DATE_CELLビルトイン

説明

指定された行と列から特定されたレコード・グループ・セルのDATE値を返します。セルは行と列が交差する部分です。

構文

```
FUNCTION GET_GROUP_DATE_CELL
  (groupcolumn_id  GroupColumn,
   row_number      NUMBER);
FUNCTION GET_GROUP_DATE_CELL
  (groupcolumn_name  VARCHAR2,
   row_number        NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

DATE

問合せ入力モード

可

パラメータ

groupcolumn_id	レコード・グループの列作成時にForm Builderによって割り当てられた一意のIDを指定します。適切な型の変数にIDを戻すには、FIND_COLUMNビルトインを使用します。このIDのデータ型はGROUPCOLUMN型です。

groupcolumn_name	レコード・グループの列を定義するときに付けたデータ型VARCHAR2の完全修飾名を指定します。recordgroup_name.groupcolumn_nameのように、レコード・グループ名およびドットが前に付けられます。列が問合せの結果定義された場合は、その列の名前は対応するデータベース列の名前と同じです。
row_number	セルの値を取り出す行を指定します。

GET_GROUP_DATE_CELLの制限事項

パラメータrow_numberには、レコード・グループ内に存在する行番号を指定する必要があります。存在しないrow_numberを指定すると、索引範囲外エラーが発生します。

GET_GROUP_DATE_CELLの例

```
/*
** Built-in:GET_GROUP_DATE_CELL
** Example:Lookup a row in a record group, and return the
**          minimum order date associated with that row in
**          the record group.Uses the 'is_value_in_list'
**          function from the GET_GROUP_CHAR_CELL example.
*/
FUNCTION Max_Order_Date_Of( part_no VARCHAR2 )
RETURN DATE IS
  fnd_row NUMBER;
BEGIN
  /*
  ** Try to lookup the part number among the temporary part
  ** list record group named 'TMPPART' in its 'PARTNO'
  ** column.
  */
  fnd_row := Is_Value_In_List( part_no, 'TMPPART', 'PARTNO');
  IF fnd_row = 0 THEN
    Message('Part Number '||part_no||' not found.');
    RETURN NULL;
  ELSE
    /*
    ** Get the corresponding Date cell value from the
    ** matching row.
    */
    RETURN Get_Group_Date_Cell( 'TMPPART.MAXORDDATE', fnd_row );
 END IF;
END;
```

GET_GROUP_NUMBER_CELLビルトイン

説明

指定された行と列から特定されたレコード・グループ・セルのNUMBER値を返します。セルは行と列が交差する部分です。

構文

```
FUNCTION GET_GROUP_NUMBER_CELL
  (groupcolumn_id  GroupColumn,
   row_number      NUMBER);
FUNCTION GET_GROUP_NUMBER_CELL
  (groupcolumn_name  VARCHAR2,
   row_number        NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

問合せ入力モード

可

パラメータ

groupcolumn_id	レコード・グループの列作成時にForm Builderによって割り当てられた一意のIDを指定します。適切な型の変数にIDを戻すには、FIND_COLUMNビルトインを使用します。このIDのデータ型はGROUPCOLUMN型です。
groupcolumn_name	レコード・グループの列を定義するときに付けたデータ型VARCHAR2の完全修飾名を指定します。recordgroup_name.groupcolumn_nameのように、レコード・グループ名およびドットが前に付けられます。列が問合せの結果定義された場合は、その列の名前は対応するデータベース列の名前と同じです。
row_number	セルの値を取り出す行を指定します。

GET_GROUP_NUMBER_CELLの制限事項

パラメータrow_numberには、レコード・グループ内に存在する行番号を指定する必要があります。存在しないrow_numberを指定すると、索引範囲外エラーが発生します。

GET_GROUP_NUMBER_CELLの例

```
/*
```

```
** Built-in:GET_GROUP_NUMBER_CELL
** Example:Lookup a row in a record group, and return the
**          minimum order quantity associated with that row
**          in the record group.Uses the
**          'is_value_in_list' function from the
**          GET_GROUP_CHAR_CELL example.
*/
FUNCTION Min_Order_Qty_Of( part_no VARCHAR2 )
RETURN NUMBER IS
  fnd_row NUMBER;
BEGIN
  /*
  ** Try to lookup the part number among the temporary part
  ** list record group named 'TMPPART' in its 'PARTNO'
  ** column.
  */
  fnd_row := Is_Value_In_List( part_no, 'TMPPART', 'PARTNO');

  IF fnd_row = 0 THEN
    Message('Part Number '||part_no||' not found.');
    RETURN NULL;
  ELSE
    /*
    ** Get the corresponding Number cell value from the
    ** matching row.
    */
    RETURN Get_Group_Number_Cell( 'TMPPART.MINQTY', fnd_row );
  END IF;
END;
```

GET_GROUP_RECORD_NUMBERビルトイン

説明

列の値がcell_valueパラメータに等しいレコード・グループ内の最初のレコードのレコード番号を返します。一致するレコードがなければ、0（ゼロ）が返されます。

構文

```
FUNCTION GET_GROUP_RECORD_NUMBER
  (groupcolumn_id  GroupColumn,
   cell_value      NUMBER);
```

```
FUNCTION GET_GROUP_RECORD_NUMBER
  (groupcolumn_name  VARCHAR2,
   cell_value      NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

問合せ入力モード

可

パラメータ

groupcolumn_id	レコード・グループの列作成時にForm Builderによって割り当てられた一意のIDを指定します。FIND_COLUMNビルトインを使用すると、このIDを変数に返すことができます。このIDのデータ型はGROUPCOLUMN型です。
groupcolumn_name	ユーザーがグループを作成するときに付けたレコード・グループ列の名前を指定します。その名前のデータ型はVARCHAR2です。
cell_value	指定したレコード・グループ列で検索する値を指定します。データ型は、VARCHAR2、NUMBERまたはDATEです。

GET_GROUP_RECORD_NUMBERの制限事項

cell_valueパラメータのデータ型は、レコード・グループ列と同じにしてください。データ型VARCHAR2の比較では、大文字と小文字が区別されます。

GET_GROUP_RECORD_NUMBERの例

```
/*
** Built-in:GET_GROUP_RECORD_NUMBER
** Example:Find the first record in the record group with a
**     cell in a column that is identical to the value
**    specified in the cell_value parameter.
*/
DECLARE
  rg_id        RecordGroup;
  match        NUMBER := 2212;
  status       NUMBER;
  the_recordnum NUMBER;
BEGIN
  rg_id := Create_Group_From_Query('QGROUP',
         'SELECT ENAME,EMPNO,SAL FROM EMP ORDER BY SAL DESC');
```

```
  status := Populate_Group( rg_id );
  */  *** Zero status is success***  /
  IF status = 0 THEN
    the_recordnum :=Get_Group_Record_Number('QGROUP.ENAME',match);
    Message('The first match is record number
'||to_CHAR(the_recordnum));
  ELSE
    Message('Error creating query record group.');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

GET_GROUP_ROW_COUNTビルトイン

説明

レコード・グループ内の行数を返します。

構文

```
FUNCTION GET_GROUP_ROW_COUNT
  (recordgroup_id  RecordGroup);
FUNCTION GET_GROUP_ROW_COUNT
  (recordgroup_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

問合せ入力モード

可

パラメータ

recordgroup_id	レコード・グループ作成時にForm Builderによってそのグループに割り当てられた一意のIDを指定します。FIND_GROUPビルトインを使用して、そのIDを変数に戻します。そのIDのデータ型はRECORDGROUPです。
recordgroup_name	ユーザーがレコード・グループ作成時に与えたレコード・グループ名を指定します。その名前のデータ型はVARCHAR2です。

GET_GROUP_ROW_COUNTの例

```
/*
** Built-in:GET_GROUP_ROW_COUNT
** Example:Determine how many rows were retrieved by a
**          Populate_Group for a query record group.
*/
DECLARE
  rg_id       RecordGroup;
  status      NUMBER;
  the_rowcount NUMBER;
BEGIN
  rg_id := Create_Group_From_Query('MY_QRY_GROUP',
         'SELECT ENAME,EMPNO,SAL FROM EMP ORDER BY SAL DESC');
  status := Populate_Group( rg_id );
  */  *** Zero status is success***  /
  IF status = 0 THEN
    the_rowcount := Get_Group_Row_Count( rg_id );
    Message('The query retrieved '||to_CHAR(the_rowcount)||
          ' record(s)');
  ELSE
    Message('Error creating query record group.');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

GET_GROUP_SELECTIONビルトイン

説明

指定されたグループで選択された行の順序番号を返します。

構文

```
FUNCTION GET_GROUP_SELECTION
  (recordgroup_id   RecordGroup,
   selection_number  NUMBER);
FUNCTION GET_GROUP_SELECTION
  (recordgroup_name  VARCHAR2,
   selection_number  NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

問合せ入力モード

可

パラメータ

recordgroup_id	レコード・グループ作成時にForm Builderによってそのグループに割り当てられた一意のIDを指定します。FIND_GROUPビルトインを使用して、そのIDを変数に戻します。そのIDのデータ型はRECORDGROUPです。
recordgroup_name	ユーザーがレコード・グループ作成時に与えたレコード・グループ名を指定します。
selection_number	選択された行が何番目に選択されたかを識別します。たとえば、行3、7、および21が選択されている場合、選択の値は、それぞれ1、2および3です。selection_numberの引数は、NUMBERデータ型の値をとります。

GET_GROUP_SELECTIONの例

```
/*
** Built-in:GET_GROUP_SELECTION
** Example:Return a comma-separated list (string) of the
**          selected part numbers from the presumed
**          existent PARTNUMS record group.
*/
FUNCTION Comma_Separated_Partnumbers
RETURN VARCHAR2 IS
  tmp_str     VARCHAR2(2000);
  rg_id       RecordGroup;
  gc_id       GroupColumn;
  the_Rowcount NUMBER;
  sel_row     NUMBER;
  the_val     VARCHAR2(20);
BEGIN
  rg_id := Find_Group('PARTNUMS');
  gc_id := Find_Column('PARTNUMS.PARTNO');
  /*
  ** Get a count of how many rows in the record group have
  ** been marked as "selected"
  */
  the_Rowcount := Get_Group_Selection_Count( rg_id );
  FOR j IN 1..the_Rowcount LOOP
    /*
    ** Get the Row number of the J-th selected row.
```

```
    */
    sel_row := Get_Group_Selection( rg_id, j );
    /*
    ** Get the (VARCHAR2) value of the J-th row.
    */
    the_val := Get_Group_CHAR_Cell( gc_id, sel_row );
    IF j = 1 THEN
      tmp_str := the_val;
    ELSE
      tmp_str := tmp_str||','||the_val;
    END IF;
  END LOOP;
  RETURN tmp_str;
END;
```

GET_GROUP_SELECTION_COUNTビルトイン

説明

指定されたレコード・グループ内で、SET_GROUP_SELECTIONコールによって「選択済み」とマークされた行数を返します。

構文

```
FUNCTION GET_GROUP_SELECTION_COUNT
  (recordgroup_id  RecordGroup);
FUNCTION GET_GROUP_SELECTION_COUNT
  (recordgroup_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

問合せ入力モード

可

パラメータ

recordgroup_id	レコード・グループ作成時にForm Builderによってそのグループに割り当てられた一意のIDを指定します。FIND_GROUPビルトインを使用して、そのIDを変数に戻します。そのIDのデータ型はRECORDGROUPです。

recordgroup_name	ユーザーがレコード・グループ作成時に与えたレコード・グループ名を指定します。

GET_GROUP_SELECTION_COUNTの例

```
/*
** Built-in:GET_GROUP_SELECTION_COUNT
** Example:See GET_GROUP_SELECTION
*/
```

GET_INTERFACE_POINTERビルトイン

説明

OLE2オートメーション・オブジェクトにハンドルを返します。

構文

```
FUNCTION GET_INTERFACE_POINTER
  (item_id  Item);
FUNCTION GET_INTERFACE_POINTER
  (item_name  VARCHAR2);
```

戻り値

PLS_INTEGER

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。

GET_INTERFACE_POINTERの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効。

GET_INTERFACE_POINTERの例

```
/*
** Built-in:GET_INTERFACE_POINTER
** Example:Finds a handle to an OLE object
*/
FUNCTION HandleMap(MapName VARCHAR2) RETURN OLE2.obj_type is
BEGIN
 RETURN(Get_interface_Pointer(MapName));
END;
```

GET_ITEM_INSTANCE_PROPERTYビルトイン

説明

指定された項目インスタンスのプロパティの値を返します。
GET_ITEM_INSTANCE_PROPERTYは、初期値または指定された項目インスタンスについてSET_ITEM_INSTANCE_PROPERTYによって最後に指定された値を返します。プロパティの有効値は返しません(項目インスタンス、項目およびブロックの各レベルで指定されたプロパティを組み合わせて導出される値)。有効なプロパティ値の詳細は、SET_ITEM_INSTANCE_PROPERTYを参照してください。

構文

```
FUNCTION GET_ITEM_INSTANCE_PROPERTY
  (item_id  ITEM,
   record_number  NUMBER,
   property  NUMBER);
FUNCTION GET_ITEM_INSTANCE_PROPERTY
  (item_name  VARCHAR2,
   record_number  NUMBER,
   property   NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

item_id	オブジェクト作成時にForm Builderによってそのオブジェクトに割り当てられた一意のID。FIND_ITEMビルトインを使用すると、このIDをデータ型ITEMの変数に返すことができます。
item_name	ユーザーがオブジェクトを作成するときにそのオブジェクトに付けた名前。
record_number	レコード番号またはCURRENT_RECORD。
property	指定した項目で値を取出すプロパティ。有効なプロパティの値は次のとおりです。 BORDER_BEVEL 項目インスタンスのレベルでBORDER_BEVELプロパティがRAISED、LOWEREDまたはPLAINに設定されている場合、それぞれRAISED、LOWEREDまたはPLAINが返されます。項目インスタンスのレベルでBORDER_BEVELが指定されていない場合は、" "が返されます。 INSERT_ALLOWED 項目インスタンスのINSERT_ALLOWEDプロパティがTRUEに設定されていれば、VARCHAR2文字列TRUEが返されます。FALSEに設定されている場合はFALSEが返されます。 NAVIGABLE 項目インスタンスのNAVIGABLEプロパティがTRUEに設定されていれば、VARCHAR2文字列TRUEが返されます。FALSEに設定されている場合はFALSEが返されます。 REQUIRED 項目インスタンスのREQUIREDプロパティがTRUEに設定されていれば、VARCHAR2文字列TRUEが返されます。FALSEに設定されている場合はFALSEが返されます。 SELECTED_RADIO_BUTTON 指定のレコードのラジオ・グループ内の選択されたラジオ・ボタンのラベルが返されます。指定したレコードのラジオ・グループに選択したラジオ・ボタンがないか、指定したレコードが表示領域外にスクロールされている場合に、NULLが返されます。 UPDATE_ALLOWED 項目インスタンスの「UPDATE_ALLOWED」プロパティがTRUEに設定されていれば、VARCHAR2文字列TRUEが返されます。FALSEに設定されている場合はFALSEが返されます。 VISUAL_ATTRIBUTE 現在有効な可視属性の名前が返されます。項目インスタンスに名前付き可視属性が割り当てられていなければ、デフォルトの可視属性を表すDEFAULTが返されます。項目インスタンスのレベルでVISUAL_ATTRIBUTEが指定されていなければ、" "が返されます。

GET_ITEM_PROPERTYビルトイン

説明

指定された項目のプロパティの値を返します。特定のオブジェクトのプロパティを取得できる場合があることに注意してください。ただし、設定することはできません。

構文

```
FUNCTION GET_ITEM_PROPERTY
  (item_id,  ITEM
   property  NUMBER);
FUNCTION GET_ITEM_PROPERTY
  (item_name  VARCHAR2,
   property   NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

item_id	オブジェクト作成時にForm Builderによってそのオブジェクトに割り当てられた一意のID。FIND_ITEMビルトインを使用すると、このIDをデータ型ITEMの変数に返すことができます。
item_name	ユーザーがオブジェクトを作成するときにそのオブジェクトに付けた名前。
property	指定した項目から取り出すプロパティの値。有効なpropertyの値は次のとおりです。 AUTO_HINT 「ヒントの自動表示」プロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」に設定されていればVARCHAR2文字列FALSEが返されます。 AUTO_SKIP 「自動スキップ」が「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」に設定されていれば文字列FALSEが返されます。 BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 BLOCK_NAME この項目のVARCHAR2ブロック名が返されます。 BORDER_BEVEL 項目のレベルでBORDER_BEVELプロパティがRAISED、LOWEREDまたはPLAINに設定されている場合、それぞれRAISED、LOWEREDまたはPLAINが返されます。

property（続き）	CASE_INSENSITIVE_QUERY このプロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」 に設定されていれば文字列FALSEが返されます。 CASE_RESTRICTION 項目のテキストが大文字で表示される場合は、UPPERCASEが返され、テキストが小文字で表示される場合は、LOWERCASEが返されます。大文字小文字の制限が有効でない場合は、NONEが返されます。 COLUMN_NAME データブロック項目が関連付けられているデータベース内の列の名前が返されます。 COMPRESS Oracleの内部形式へ変換される場合に、ファイルからフォームに読み込まれるサウンド・オブジェクトを圧縮する必要があるかどうかを示す値（TRUEまたはFALSE）が返されます。 CONCEAL_DATA オペレータがテキスト項目へ入力するテキストが非表示の場合はVARCHAR2文字列TRUEが返され、オペレータがテキスト項目へ入力するテキストが表示される場合はVARCHAR2文字列FALSEが返されます。 CURRENT_RECORD_ATTRIBUTE 指定した項目のユーザー命名可視属性の名前がVARCHAR2型で返されます。 CURRENT_ROW_BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 CURRENT_ROW_FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 CURRENT_ROW_FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 CURRENT_ROW_FONT_SIZE ポイント数の100倍で指定されるフォントのサイズ(例えば、フォント･サイズが8ポイントの項目では、800が返されます)。 CURRENT_ROW_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 CURRENT_ROW_FONT_STYLE フォントのスタイル。 CURRENT_ROW_FONT_WEIGHT フォントの太さ。

property（続き）	CURRENT_ROW_FOREGROUND_COLOR	オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。
	CURRENT_ROW_WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	DATABASE_VALUE	実表項目の場合は、本来データベースからフェッチされた値が返されます。
	DATATYPE	ALPHA、CHAR、DATE、JDATE, EDATE、DATETIME、INT、RINT、MONEY、RMONEY、NUMBER、RNUMBER、TIME、LONG、GRAPHICS、またはIMAGEの項目のデータ型が返されます。ボタンやチャートなど、データ型を持たない種類の項目があることに注意してください。データ型がないためにエラー・メッセージが出力されることのないよう、データ型を取得する前に項目タイプを選別してください。
	DIRECTION	両方向オブジェクトのレイアウト方向が返されます。有効な戻り値はRIGHT_TO_LEFT、LEFT_TO_RIGHTです。
	DISPLAYED	VARCHAR2文字列TRUEまたはFALSEが返されます。
	ECHO	この項目の「データ隠ぺい」プロパティが「いいえ」に設定されていればVARCHAR2文字列TRUEが返され、「はい」に設定されていればVARCHAR2文字列FALSEが返されます。
	EDITOR_NAME	テキスト項目に連結されたエディタの名前が返されます。
	EDITOR_X_POS	テキスト項目に連結されたエディタのx座標が返されます。（このプロパティは、「エディタY位置」プロパティに相当します。）
	EDITOR_Y_POS	テキスト項目に連結されたエディタのy座標が返されます。（このプロパティは、「エディタY位置」プロパティに相当します。）
	ENFORCE_KEY	新規レコードがマスター/ディテールのリレーションの一部として作成されたときに、値が外部キーとしてこの項目にコピーされた項目の名前が返されます。（このプロパティは、「Copy」プロパティに相当します。）
	ENABLED	「変更可能」プロパティが「はい」に設定されていればTRUEが返され、「いいえ」に設定されていればFALSEが返されます。
	FILL_PATTERN	オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。

property（続き）	FIXED_LENGTH	この項目のこのプロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」に設定されていればVARCHAR2文字列FALSEが返されます。
	FONT_NAME	オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。
	FONT_SIZE	ポイント数で指定されるフォントのサイズ。
	FONT_SPACING	フォントの幅つまり文字間のスペース（カーニング）。
	FONT_STYLE	フォントのスタイル。
	FONT_WEIGHT	フォントの太さ。
	FOREGROUND_COLOR	オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。
	FORMAT_MASK	テキスト項目で使用されている書式マスクが返されます。
	HEIGHT	項目の高さが返されます。高さの単位は、そのフォームに指定した「座標システム」プロパティの値とデフォルトのフォント・スケーリングによって異なります。
	HINT_TEXT	実行時にメッセージ行に表示される項目特有のヘルプ・テキストが返されます。
	ICON_NAME	「アイコン化」プロパティがTRUEに設定されたボタン項目に関連するアイコン・リソースのファイル名が返されます。
	ICONIC_BUTTON	ボタンがアイコン・ボタンとして定義されていればVARCHAR2値TRUEが返され、ボタンがアイコン・ボタンとして定義されていなければVARCHAR2値FALSEが返されます。
	IMAGE_DEPTH	指定したイメージ項目の色の濃さが返されます。
	IMAGE_FORMAT	指定したイメージ項目の形式が返されます。
	INSERT_ALLOWED	項目のレベルでINSERT_ALLOWEDプロパティが「はい」に設定されている場合、VARCHAR2値TRUEが返されます。「いいえ」に設定されている場合はFALSEが返されます。
	ITEM_CANVAS	この項目が割り当てられているキャンバスの名前が返されます。
	ITEM_IS_VALID	現項目が有効であればVARCHAR2文字列TRUEが返され、現項目が有効でなければVARCHAR2文字列FALSEが返されます。
	ITEM_NAME	項目の名前が返されます。
	ITEM_TAB_PAGE	項目が割り当てられているタブ・ページの名前が返されます。Form Builderが項目のタブ・ページの名前を返すことができるように、項目をタブ・キャンバスに割り当てる必要があることに注意してください。

property（続き）	ITEM_TYPE	項目のタイプが返されます。項目がボタンであればBUTTONが、チャート項目であればCHART ITEMが、チェック・ボックスであればCHECKBOXが、表示項目であればDISPLAY ITEMが、イメージ項目であればIMAGEが、リスト項目であればLISTが、OCXコントロールまたはOLEコンテナであればOLE OBJECTが、ラジオ・グループであればRADIO GROUPが、テキスト項目であればTEXT ITEMが、ユーザー領域であればUSER AREAが、VBXコントロールであるカスタム項目であればVBX CONTROLが返されます。
	JUSTIFICATION	テキストの文字揃えが返されます。テキストの文字揃えはテキスト項目および表示項目だけに適用されます。有効な戻り値は、START、END、LEFT、CENTER、RIGHTです。
	KEEP_POSITION	カーソルが項目から移動されたとき、移動前にあった特定の位置へ再入力される場合はVARCHAR2文字列TRUEが返され、カーソルがデフォルトの位置へ再入力される場合はVARCHAR2文字列FALSEが返されます。
	LABEL	項目の「ラベル」プロパティに定義されたVARCHAR2値が返されます。このプロパティは、ボタンやチェックボックスのようにラベルを持つ項目にのみ有効です。
	LIST	項目が、値リスト(LOV)が割り当てられているテキスト項目であればVARCHAR2文字列TRUEが返され、それ以外であればVARCHAR2文字列FALSEが返されます。
	LOCK_RECORD_ON_CHANGE	この項目が変更される可能性があるときに、Form Builderが行をロックする必要があれば、VARCHAR2文字列TRUEが返され、ロックする必要がなければ、VARCHAR2文字列FALSEが返されます。
	LOV_NAME	任意の項目に関連付けられているLOVのVARCHAR2名が返されます。指定した値リスト名が存在しない場合は、エラー・メッセージが表示されます。
	LOV_X_POS	このテキスト項目に関連するLOVのx座標が返されます。（このプロパティは、「X位置のリスト」プロパティに相当します。
	LOV_Y_POS	このテキスト項目に関連するLOVのy座標が返されます。（このプロパティは、「Y位置のリスト」プロパティに相当します。
	MAX_LENGTH	項目の最大長の設定が返されます。この値はデータ型NUMBERで返されます。
	MERGE_CURRENT_ROW_VA	指定された可視属性の内容を現行行の可視属性とマージします（置換ではない）。
	MERGE_TOOLTIP_ATTRIBUTE	指定された可視属性の内容をツールチップの現行の可視属性とマージします（置換ではない）。

property（続き）	MERGE_VISUAL_ATTRIBUTE 指定された可視属性の内容をオブジェクトの現行の可視属性とマージします（置換ではない）。 MOUSE_NAVIGATE　項目の「マウス・ナビゲート」プロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」に設定されていれば VARCHAR2文字列FALSEが返されます。 MULTI_LINE　指定した項目が複数行のテキスト項目であればVARCHAR2値TRUEが返され、1行のテキスト項目であればVARCHAR2文字列FALSEが返されます。 NAVIGABLE　項目の「キーボードで移動可能」プロパティが「はい」に設定されていれば、VARCHAR2文字列TRUEが返され、「いいえ」に設定されていれば、VARCHAR2文字列FALSEが返されます。 NEXTITEM　デフォルトのナビゲーション順序での次の項目の名前が、オブジェクト・ナビゲータで定義した項目の順序のとおりに返されます。 NEXT_NAVIGATION_ITEM この現項目の"次ナビゲーション項目"として定義されている項目の名前が返されます。 POPUPMENU_CONTENT_ITEM 次のいずれかのOLEポップアップ・メニュー項目プロパティの設定が返されます。 POPUPMENU_COPY_ITEM POPUPMENU_CUT_ITEM POPUPMENU_DELOBJ_ITEM POPUPMENU_INSOBJ_ITEM POPUPMENU_LINKS_ITEM POPUPMENU_OBJECT_ITEM POPUPMENU_PASTE_ITEM POPUPEMNU_PASTESPEC_ITEM OLEポップアップ・メニュー項目が非表示の場合は、VARCHAR2文字列HIDDENを返します。OLEポップアップ・メニュー項目が表示され使用可能に設定されている場合は、VARCHAR2文字列ENABLEDを返します。OLEポップアップ・メニュー項目が表示され変更可能に設定されていない場合は、VARCHAR2文字列DISABLEDを返します。また、プラットフォームがMicrosoft Windowsでない場合は、VARCHAR2文字列UNSUPPORTEDを返します。 PREVIOUSITEM　前の項目の名前が返されます。

property（続き）	PREVIOUS_NAVIGATION_ITEM この現項目の"前ナビゲーション項目"として定義されている項目の名前が返されます。 PRIMARY_KEY 項目が主キーであればVARCHAR2文字列TRUEが返され、主キーでなければVARCHAR2文字列FALSEが返されます。 PROMPT_ALIGNMENT_OFFSET 項目と項目のプロンプトの間の距離がVARCHAR2値として返されます。 PROMPT_BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 PROMPT_DISPLAY_STYLE プロンプトの表示スタイルとしてFIRST_RECORD、HIDDEN、ALL_RECORDSのいずれかが返されます。 PROMPT_EDGE START、END、TOP、BOTTOMのいずれかのうち、どの境界線に項目プロンプトが連結されているかを示す値が返されます。 PROMPT_EDGE_ALIGNMENT START、END、CENTERのうち、どの境界線に項目のプロンプトが位置揃えされているかを示す値が返されます。 PROMPT_EDGE_OFFSET 項目とその項目のプロンプトの間の距離がVARCHAR2値として返されます。 PROMPT_FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 PROMPT_FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 PROMPT_FONT_SIZE ポイント数で指定されるフォントのサイズ。 PROMPT_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 PROMPT_FONT_STYLE フォントのスタイル。

property（続き）	PROMPT_FONT_WEIGHT フォントの太さ。 PROMPT_FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 PROMPT_TEXT　項目について表示されるテキスト・ラベルが返されます。 PROMPT_TEXT_ALIGNMENT START、LEFT、RIGHT、CENTER、ENDのうち、プロンプトの位置揃えの方法を示す値が返されます。 PROMPT_VISUAL_ATTRIBUTE プロンプトのユーザー命名可視属性を示す値が返されます。 PROMPT_WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 QUERYABLE　項目を問合せに含めることができる場合はVARCHAR2文字列TRUEが返され、項目を問合せに含めることができない場合はVARCHAR2文字列FALSEが返されます。 QUERY_LENGTH　フォームが問合せ入力モードの場合にオペレータがテキスト項目に入力できる文字数が返されます。 QUERY_ONLY　項目のこのプロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」に設定されていればVARCHAR2文字列FALSEが返されます。 RANGE_HIGH　範囲制限の最大値が返されます。（このプロパティは「最低許容値」プロパティに相当します。 RANGE_LOW　範囲制限の最小値が返されます。（このプロパティは「最低許容値」プロパティに相当します。 REQUIRED　複数行のテキスト項目において、項目のレベルでREQUIREDプロパティが真に設定されている場合、VARCHAR2文字列TRUEが返されます。偽に設定されている場合はFALSEが返されます。 SCROLLBAR　「垂直スクロール・バー表示」プロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」に設定されていればVARCHAR2文字列FALSEが返されます。 SHOW_FAST_FORWARD_BUTTON  が指定したサウンド項目上に表示される場合はVARCHAR2値TRUEが返され、そうでない場合はFALSEが返されます。

property（続き）	SHOW_PALETTE　イメージ操作パレットが指定したイメージ項目の近くに表示される場合はVARCHAR2値TRUEが返され、そうでない場合はFALSEが返されます。 SHOW_PLAY_BUTTON　指定したサウンド項目に  が表示される場合はVARCHAR2値TRUEが返され、表示されない場合はFALSEが返されます。 SHOW_RECORD_BUTTON 指定したサウンド項目に  が表示される場合はVARCHAR2値TRUEが返され、表示されない場合はFALSEが返されます。 SHOW_REWIND_BUTTON 指定したサウンド項目に  が表示される場合はVARCHAR2値TRUEが返され、表示されない場合はFALSEが返されます。 SHOW_SLIDER　指定したサウンド項目にスライダー位置制御が表示される場合はVARCHAR2値TRUEが返され、表示されない場合はFALSEが返されます。 SHOW_TIME_INDICATOR 指定したサウンド項目に  が表示される場合はVARCHAR2値TRUEが返され、表示されない場合はFALSEが返されます。 SHOW_VOLUME_CONTROL 指定したサウンド項目に  が表示される場合はVARCHAR2値TRUEが返され、表示されない場合はFALSEが返されます。TOOLTIP_BACKGROUND_COLORオブジェクトのバックグラウンド領域のカラー。 TOOLTIP_BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 TOOLTIP_FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 TOOLTIP_FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 TOOLTIP_FONT_SIZE ポイント数で指定されるフォントのサイズ。 TOOLTIP_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 TOOLTIP_FONT_STYLE フォントのスタイル。

property（続き）	TOOLTIP_FONT_WEIGHT フォントの太さ。 TOOLTIP_FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 TOOLTIP_WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 TOOLTIP_TEXT　項目のヒント・テキストが返されます。 UPDATE_ALLOWED　項目のレベルで「更新可」プロパティが「はい」に設定されている場合、VARCHAR2文字列TRUEが返されます。「いいえ」に設定されている場合はFALSEが返されます。 UPDATE_COLUMN　Form Builderで項目を更新済みとして扱う必要があればVARCHAR2文字列TRUEが返され、更新済みとして扱う必要がなければFALSEが返されます。 UPDATE_NULL　項目がNULLの場合のみに更新の必要がある場合はVARCHAR2文字列TRUEが返され、項目を常に更新できる場合はVARCHAR2文字列FALSEが返されます。（このプロパティは、「NULLのみ更新」プロパティに相当します。 UPDATE_PERMISSION 「UPDATE_PERMISSION」プロパティが「はい」に設定され、項目の「更新可」プロパティと「NULLのみ更新」プロパティが「ON」に設定されていればVARCHAR2文字列TRUEが返されます。VARCHAR2文字列FALSEは、「更新可」と「NULLのみ更新」が「いいえ」であることを表します VALIDATE_FROM_LIST Form Builder 側で、連結されている値リスト内の値に対するテキスト項目の値の妥当性チェックが必要な場合はVARCHAR2文字列TRUEが返され、妥当性チェックが不要な場合はVARCHAR2文字列FALSEが返されます。 VISIBLE　項目のこのプロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」に設定されていればVARCHAR2文字列FALSEが返されます。 VISUAL_ATTRIBUTE 現在有効な可視属性の名前が返されます。項目に名前付き可視属性が割り当てられていなければ、デフォルトの可視属性を表すDEFAULTが返されます。

property（続き）	WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	WIDTH	項目の幅が返されます。
	WINDOW_HANDLE	オブジェクトの参照に使用される一意の内部VARCHAR2定数が返されます。プラットフォームがMicrosoft Windows以外であればVARCHAR2値'0'が返されます。
	WRAP_STYLE	項目の「ラップ形式」プロパティが「文字単位」に設定されていればVARCHAR2が、「単語」が設定されていればWORDが、ラップ形式が項目に指定されていなければNONEが返されます。
	X_POS	キャンバスの左上隅に対する項目の左上隅の現行の位置を示すx座標が返されます。
	Y_POS	キャンバスの左上隅に対する項目の左上隅の現行の位置を示すy座標が返されます。

使用上の注意

GET_ITEM_PROPERTYビルトインを使用して、その項目では無効なプロパティを取得しようとすると、エラーが発生します。たとえば、ラジオ・グループからLISTを取得しようとすると、LISTはテキスト項目にのみ有効なプロパティであるため、エラーが発生します。

GET_ITEM_PROPERTYの例

```
/*
**  Built-in:GET_ITEM_PROPERTY
**  Example:Navigate to the next required item in the
**           current block.*/
PROCEDURE Go_Next_Required_Item IS
  cur_blk       VARCHAR2(40);
  cur_itm       VARCHAR2(80);
  orig_itm      VARCHAR2(80);
  first_itm     VARCHAR2(80);
  wrapped       BOOLEAN := FALSE;
  found         BOOLEAN := FALSE;
  Exit_Procedure EXCEPTION;
  /*
  ** Local function returning the name of the item after the
  ** one passed in. Using NVL we make the item after the
  ** last one in the block equal the first item again.
  */
  */
  FUNCTION The_Item_After(itm VARCHAR2)
  RETURN VARCHAR2 IS
```

```
 BEGIN
    RETURN cur_blk||'.'||
      NVL(Get_Item_Property(itm,NEXTITEM),
      first_itm);
 END;
BEGIN
  cur_blk   := :System.Cursor_Block;
  first_itm := Get_Block_Property( cur_blk, FIRST_ITEM );
  orig_itm  := :System.Cursor_Item;
  cur_itm   := The_Item_After(orig_itm);
  /*
  ** Loop until we come back to the item name where we started
  */
  WHILE (orig_itm <> cur_itm) LOOP

    /*
    ** If required item, set the found flag and exit procedure
    */
    IF Get_Item_Property(cur_itm,REQUIRED) = 'TRUE' THEN
      found := TRUE;
      RAISE Exit_Procedure;
    END IF;
    /*
    ** Setup for next iteration
    */
    cur_itm := The_Item_After(cur_itm);
  END LOOP;
  /*
  ** If we get here we wrapped all the way around the
  ** block's item names
  */
  wrapped := TRUE;
  RAISE Exit_Procedure;
EXCEPTION
  WHEN Exit_Procedure THEN
    /*
    ** If we found a required item and we didn't come back
    ** to the item we started in, then navigate there
    */
    IF found AND NOT wrapped THEN
      Go_Item(cur_itm);
    END IF;
END;
```

GET_LIST_ELEMENT_COUNTビルトイン

説明

NULL値の要素も含め、リスト内のリスト項目要素数の合計を返します。

構文

```
FUNCTION GET_LIST_ELEMENT_COUNT
  (list_id  Item);
FUNCTION GET_LIST_ELEMENT_COUNT
  (list_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

list_id	Form Builderがリスト項目を作成するときに割り当てる一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
list_name	ユーザーがリスト項目を作成するときに付けた名前。その名前のデータ型はVARCHAR2です。

GET_LIST_ELEMENT_COUNTの例

```
/*
**  Built-in:GET_LIST_ELEMENT_COUNT
**  Example:Add an element to the list item.Before adding
**           the element, verify that the element is not in
**           the current list.
*/
DECLARE
   total_list_count    NUMBER(2);
   loop_index_var     NUMBER(2) := 1;
   list_element       VARCHAR2(50);
```

```
   list_element_value  VARCHAR2(50);
   list_element_to_add   VARCHAR2(50);
   list_value_to_add   VARCHAR2(50);
   element_match      VARCHAR2(5) := 'TRUE';
   value_match        VARCHAR2(5) := 'TRUE';
BEGIN
/*
** Determine the total number of list elements.
*/
   total_list_count := Get_List_Element_Count(list_id);
/*
** Compare the current list item elements with the element that
** will be added.
*/
   LOOP
     list_element := Get_List_Element_Value(list_id,
     loop_index_var);
     loop_index_var := loop_index_var + 1;
     IF list_element_to_add = list_element THEN
       element_match := 'FALSE';
     END IF;
     EXIT WHEN list_element = list_element_to_add OR
     loop_index_var = total_list_count;
   END LOOP;
/*
** Compare the current list item values with the value that
** will be added.
*/
   loop_index_var := 1;
   LOOP
     list_element_value:= Get_List_Element_Value(list_id,
        loop_index_var);
     loop_index_var := loop_index_var + 1;
     IF list_value_to_add = list_element_value THEN
       value_match := 'FALSE';
     END IF;
     EXIT WHEN list_element_value = list_value_to_add OR
     loop_index_var = total_list_count;
   END LOOP;
/*
**  Add the element and value if it is not in the current list
*/
   IF element_match AND value_match = 'TRUE' THEN
```

```
    Add_List_Element(list_id, list_name, list_element_to_add,
    list_value_to_add);
  END IF
END;
```

GET_LIST_ELEMENT_LABELビルトイン

説明

要求されたリスト要素のラベルに関する情報を返します。

構文

```
FUNCTION GET_LIST_ELEMENT_LABEL
  (list_id     ITEM,
   list_name   VARCHAR2,
   list_index  NUMBER);
FUNCTION GET_LIST_ELEMENT_LABEL
  (list_name   VARCHAR2,
   list_index  NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

list_id	**Form Builderがリスト項目を作成するときに割り当てる一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。**
list_name	**ユーザーがリスト項目を作成するときに付けた名前。その名前のデータ型はVARCHAR2です。**
list_index	**リスト索引値を指定します。リスト索引は1を基礎としています。リスト内の要素数より大きい値を索引に指定すると、GET_LIST_ELEMENT_LABELビルトインは異常終了します。**

使用上の注意

リスト項目の要素に関連付けられた値は、リスト項目の現行の値と同じである必要はありません。つまり、:block.list_itemの値です。

GET_LIST_ELEMENT_LABELの例

```
/*
**  Built-in:GET_LIST_ELEMENT_LABEL
**  Example:See GET_LIST_ELEMENT_COUNT
*/
```

GET_LIST_ELEMENT_VALUEビルトイン

説明

指定されたリスト項目の要素に関連付けられている値を返します。

構文

```
FUNCTION GET_LIST_ELEMENT_VALUE
  (list_id    ITEM,
   list_index  NUMBER);
FUNCTION GET_LIST_ELEMENT_VALUE
  (list_name   VARCHAR2,
   list_index  NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

list_id	Form Builderがリスト項目を作成するときに割り当てる一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
list_name	ユーザーがリスト項目を作成するときに付けた名前。その名前のデータ型はVARCHAR2です。

list_index	リスト索引値を指定します。リスト索引は1を基礎としています。このパラメータは、要求された要素の値を含む文字列を返します。リスト内の要素数より大きい値を索引に指定すると、GET_LIST_ELEMENT_LABELビルトインは異常終了します。

GET_LIST_ELEMENT_VALUEの例

```
/*
**  Built-in:GET_LIST_ELEMENT_VALUE
**  Example:See GET_LIST_ELEMENT_COUNT
*/
```

GET_LOV_PROPERTYビルトイン

説明

指定された値リスト（LOV）に関する情報を返します。

取り出すプロパティの値ごとに、このビルトインをコールする必要があります。

構文

```
FUNCTION GET_LOV_PROPERTY
  (lov_id, property  LOV);
FUNCTION GET_LOV_PROPERTY
  (lov_name  VARCHAR2,
   property  NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

lov_id	オブジェクト作成時にForm Builderによってそのオブジェクトに割り当てられた一意のIDを指定します。FIND_LOVビルトインを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型は値リスト型です。

lov_name	ユーザーがオブジェクトを作成したときにそのオブジェクトに付けた名前を指定します。
property	指定した値リストに設定するプロパティを指定します。設定できるプロパティは次のとおりです。 AUTO_REFRESH　このプロパティが「はい」に設定されている場合、つまり値リストがコールされるたびにForm Builderによって問合せが再実行される場合は、VARCHAR2文字列TRUEが返されます。このプロパティが「いいえ」に設定されていれば、VARCHAR2文字列FALSEを返します。 GROUP_NAME　現在この値リストに関連付けられているレコード・グループの名前が返されます。その名前のデータ型はVARCHAR2です。 HEIGHT　値リストの高さが返されます。高さの単位は、そのフォームに指定した「座標システム」プロパティの値とデフォルトのフォント・スケーリングによって異なります。 WIDTH　値リストの幅が返されます。高さの単位は、そのフォームに指定した「座標システム」プロパティの値とデフォルトのフォント・スケーリングによって異なります。 X_POS　画面の左上隅に対する値リストの左上隅の現行の位置を示すx座標が返されます。 Y_POS　画面の左上隅に対する値リストの左上隅の現行の位置を示すy座標が返されます。

GET_LOV_PROPERTYの例

```
/*
** Built-in:GET_LOV_PROPERTY
** Example:Can get the width/height of the LOV.
*/
DECLARE
  the_width  NUMBER;
  the_height NUMBER;
  lov_id     LOV;
BEGIN
  lov_id    := Find_LOV('My_LOV_1');
  the_width := Get_LOV_Property(lov_id, WIDTH);
  the_height := Get_LOV_Property(lov_id,HEIGHT);
END;
```

GET_MENU_ITEM_PROPERTYビルトイン

説明

特定のプロパティで指定されたメニュー項目の状態を返します。このビルトイン関数を使用して状態を取得した後、SET_MENU_ITEM_PROPERTYビルトインを使用するとプロパティの状態を変更することができます。

構文

```
FUNCTION GET_MENU_ITEM_PROPERTY
  (menuitem_id  MenuItem,
   property     NUMBER);
FUNCTION GET_MENU_ITEM_PROPERTY
  (menu_name.menuitem_name  VARCHAR2,
   property                 NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

menuitem_id	メニュー項目作成時にForm Builderによってその項目に割り当てられた一意のID。FIND_MENU_ITEMを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型はMENUITEMです。
menu_name.menuitem_name	ユーザーがメニュー項目を作成したときにその項目に付けた名前。名前でメニュー項目を指定する場合は、menu_name.menuitem_nameのように、メニュー項目を修飾するメニュー名も合せて指定してください。データ型はVARCHAR2です。

property	メニュー項目に関する情報を取り出す次の定数をどれか1つ指定します。 CHECKED　チェックボックス・メニュー項目がオンであればVARCHAR2文字列TRUEが返され、オフであればFALSEが返されます。ラジオ・メニュー項目がラジオ・グループ内で選択された項目であればVARCHAR2文字列TRUEを、グループ内の別のラジオ項目が選択されていればFALSEが返されます。その他のタイプのメニュー項目では、TRUEが返されます。 ENABLED　メニュー項目が変更可能であればVARCHAR2文字列TRUEが返され、使用不可であれば（単色で表示され、使用できない場合）、FALSEが返されます。 ICON_NAME　「メニューのアイコン」プロパティがTRUEに設定されたメニュー項目に関連付けられたアイコン・リソースのファイル名が返されます。 LABEL　メニュー項目のラベルのVARCHAR2文字列が返されます。 VISIBLE　メニュー項目を参照できる場合はVARCHAR2文字列TRUEが返され、非表示の場合にはFALSEが返されます。

GET_MENU_ITEM_PROPERTYの例

```
/*
** Built-in:GET_MENU_ITEM_PROPERTY
** Example:Toggle the enabled/disable status of the menu
**         item whose name is passed in.  Pass in a string
**         of the form 'MENUNAME.MENUITEM'.
*/
PROCEDURE Toggle_Enabled( menuitem_name VARCHAR2) IS
  mi_id MenuItem;
BEGIN
  mi_id := Find_Menu_Item( menuitem_name );
  IF Get_Menu_Item_Property(mi_id,ENABLED) = 'TRUE' THEN
    Set_Menu_Item_Property(mi_id,ENABLED,PROPERTY_FALSE);
  ELSE
    Set_Menu_Item_Property(mi_id,ENABLED,PROPERTY_TRUE);
  END IF;
END;
```

GET_MESSAGEビルトイン

説明

タイプを問わず、現行のメッセージを返します。

構文

```
FUNCTION GET_MESSAGE;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

なし

GET_MESSAGEの制限事項

Form BuilderまたはMESSAGEビルトイン・コールによって、メッセージが直接表示デバイスに渡された場合のみ、GET_MESSAGEビルトインがインスタンシエートされます。On-Messageトリガーを使用してメッセージの宛先を変えると、GET_MESSAGEコールでは値が返されません。詳細は、On-Messageトリガーを参照してください。

GET_MESSAGEの例

```
/*
** Built-in:GET_MESSAGE
** Example:Capture the contents of the Message Line in a
**          local variable
*/
DECLARE
  string_var VARCHAR2(200);
BEGIN
  string_var := Get_Message;
END;
```

GET_OLE_<proptype>ビルトイン

説明

OLEプロパティを取得します。

ファンクションには4つのバージョンがあり（proptypeの値によって示される）、引数の型CHAR、NUM、OBJおよびVARについて1つずつあります。

構文

```
FUNCTION GET_OLE_CHAR
  (obj OLEOBJ, memberid PLS_INTEGER)
RETURN oleprop VARCHAR2;

FUNCTION GET_OLE_NUM
  (obj OLEOBJ, memberid PLS_INTEGER)
RETURN oleprop NUMBER;

FUNCTION GET_OLE_OBJ
  (obj OLEOBJ, memberid PLS_INTEGER)
RETURN oleprop OLEOBJ;

FUNCTION GET_OLE_VAR
  (obj OLEOBJ, memberid PLS_INTEGER,
   persistence BOOLEAN)
RETURN oleprop OLEVAR;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

OLEプロパティ。選択されたファンクションのフォームによって、型が変わることに注意してください。

パラメータ

obj	OLEオブジェクトへのポインタ。
memberid	OLEプロパティのメンバーID。
persistence	OLEVAR引数を取り出した後の永続性を制御します。このパラメータはオプションであり、指定しない場合のデフォルト値はFALSEです（つまり非永続）。

使用上の注意

■INIT_OLEARGSコールとADD_OLEARGコールがこのGET_OLE_typeコールよりも先に発生し、その間にGET_OLE、SET_OLEまたはCALL_OLEの各コールが発生しなかった場合、このコールでは、INIT_OLEARGSコールとADD_OLEARGコールで指定された引数を使用してプロパティが取り出されます。

■ 返されるOLEVAR引数はユーザーが永続性を制御できますが、返されるOLEOBJ引数は常に非永続に設定されます。

GET_OLEARG_<type>ビルトイン

説明

OLE引数スタックからn番目の引数を取得します。

ファンクションには4つのバージョンがあり（typeの値によって示される）、引数の型CHAR、NUM、OBJおよびVARについて1つずつあります。

構文

```
FUNCTION GET_OLEARG_CHAR
  (which NUMBER)
RETURN olearg VARCHAR2;

FUNCTION GET_OLEARG_NUM
  (which NUMBER)
RETURN olearg NUMBER;

FUNCTION GET_OLEARG_OBJ
  (which NUMBER)
RETURN olearg OLEOBJ;

FUNCTION GET_OLEARG_VAR
  (which NUMBER, persistence BOOLEAN)
RETURN olearg OLEVAR;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

指定された引数。使用されるファンクションのフォームによって、型が変わることに注意してください。

パラメータ

which	OLE引数スタックのどの引数を取り出すかを示す相対数。
persistence	OLEVAR引数を取り出した後の永続性を制御します。このパラメータはオプションであり、指定しない場合のデフォルト値はFALSEです（つまり非永続）。

使用上の注意

- メソッド・コールの結果によって値が変化する可能性のある引数を取り出すには、この関数を使用します。
- 返されるOLEVAR引数はユーザーが永続性を制御できますが、返されるOLEOBJ引数は常に非永続に設定されます。

GET_OLE_MEMBERIDビルトイン

説明

OLEオブジェクトのユーザー命名メソッドまたはユーザー命名プロパティのメンバーIDを取得します。

構文

```
FUNCTION GET_OLE_MEMBERID
  (obj OLEOBJ, name VARCHAR2)
RETURN memberid PLS_INTEGER;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

メソッドまたはプロパティのメンバーID

パラメータ

obj	OLEオブジェクトへのポインタ。
name	オブジェクトのメソッドまたはプロパティの名前。

使用上の注意

メンバーIDはハードコードされた値です。返される結果は、OLEサーバーの実行に使用される言語の種類によって異なる場合があります。

GET_PARAMETER_ATTRビルトイン

説明

指定されたパラメータ・リスト内の指定されたパラメータの現行の値とタイプを返します。

構文

```
FUNCTION GET_PARAMETER_ATTR
  (list      VARCHAR2,
   key       VARCHAR2,
   paramtype  NUMBER,
   value     VARCHAR2);
FUNCTION GET_PARAMETER_ATTR
  (name      VARCHAR2,
   key       VARCHAR2,
   paramtype  NUMBER,
   value     VARCHAR2);
```

ビルトイン・タイプ

2つのOUTパラメータを返す制限なしプロシージャ

問合せ入力モード

可

パラメータ

list*または*name	**パラメータを割り当てるパラメータ・リストを指定します。実際のパラメータには、PARAMLIST型のパラメータ・リストIDまたはパラメータ・リストのVARCHAR2名のどちらかを指定できます。**
key	**パラメータの名前をVARCHAR2で指定します。**
paramtype	**データ型NUMBERのOUTパラメータ。実パラメータは、データ型NUMBERの変数で指定します。式は指定できません。このパラメータを実行すると、変数の値が次の数値定数のいずれかに設定されます。** **DATA_PARAMETER　パラメータの値がレコード・グループの名前であることを示します。** **TEXT_PARAMETER　パラメータの値が実際のデータ値であることを示します。**
value	**VARCHAR2型のOUTパラメータ。このパラメータがデータ型パラメータの場合、値はレコード・グループの名前です。パラメータがテキスト・パラメータの場合は、値は実際のテキスト・パラメータです。**

PL/SQLプロシージャでOUTパラメータを使用する場合の概要は、『PL/SQLユーザーズ・ガイドおよびリファレンス リリース8.0』を参照してください。

GET_PARAMETER_LISTビルトイン

説明

複数のパラメータ・リストから指定されたリストを検索して、指定された名前の有効なパラメータ・リストが見つかれば、パラメータ・リストIDを返します。戻り値を受け取るには、データ型PARAMLISTの変数を定義する必要があります。この関数は、他のオブジェクトを検索する場合に使用するFIND_関数に似ています。

構文

```
FUNCTION GET_PARAMETER_LIST
  (name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

ParamList

問合せ入力モード

可

パラメータ

name	有効なパラメータ・リスト名をデータ型 VARCHAR2で指定します。

GET_PARAMETER_LISTの例

CREATE_PARAMETER_LISTビルトインを参照

GET_RADIO_BUTTON_PROPERTYビルトイン

説明

指定されたラジオ・ボタンに関する情報を返します。

構文

```
FUNCTION GET_RADIO_BUTTON_PROPERTY
  (item_id     ITEM,
```

```
   button_name  VARCHAR2,
   property     NUMBER);
FUNCTION GET_RADIO_BUTTON_PROPERTY(
  item_name    VARCHAR2,
  button_name  VARCHAR2,
  property     NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

item_id	ラジオ・グループの項目IDを指定します。オブジェクト作成時に、Form Builderによってそのオブジェクトに一意のIDが割り当てられます。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
item_name	ラジオ・グループの名前を指定します。ラジオ・グループは、従属するラジオ・ボタンの所有者または親です。その名前のデータ型はVARCHAR2です。
button_name	プロパティを取得するラジオ・ボタンの名前を指定します。その名前のデータ型はVARCHAR2です。

property	現行の状態を取得するプロパティを指定します。指定できるプロパティの定数は次のとおりです。 BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 ENABLED このプロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、「いいえ」に設定されていればVARCHAR2文字列FALSEが返されます。 FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE ポイント数で指定されるフォントのサイズ。 FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE フォントのスタイル。 FONT_WEIGHT フォントの太さ。 FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 HEIGHT ラジオ・ボタンの高さが返される。この値は、データ型VARCHAR2で、また、そのフォームの「座標システム」フォーム・モジュール・プロパティに設定されている単位で返されます。 LABEL ラジオ・ボタンの実際の文字列ラベルが返されます。 PROMPT_BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 PROMPT_FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 PROMPT_FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 PROMPT_FONT_SIZE ポイント数で指定されるフォントのサイズ。 PROMPT_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 PROMPT_FONT_STYLE フォントのスタイル。 PROMPT_FONT_WEIGHT フォントの太さ。

property（続き）		
	PROMPT_FOREGROUND_COLOR	オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。
	PROMPT_WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	VISIBLE	このプロパティが「はい」に設定されていればVARCHAR2文字列TRUEが返され、このプロパティが「いいえ」に設定されていればVARCHAR2文字列FALSEが返されます。
	VISUAL_ATTRIBUTE	現在有効な可視属性の名前が返されます。ラジオ・ボタンに名前付き可視属性が割り当てられていない場合は、カスタム可視属性であればCUSTOMが、デフォルトの可視属性であればDEFAULTが返されます。
	WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	WIDTH	ラジオ・ボタンのラベル部分を含んだ幅が返されます。この値は、データ型VARCHAR2で、また、そのフォームの「座標システム」フォーム・モジュール・プロパティに設定されている単位で返されます。
	WINDOW_HANDLE	オブジェクトの参照に使用される一意の内部VARCHAR2定数が返されます。プラットフォームがMicrosoft Windowsでなければ数値0が返されます。
	X_POS	キャンバスの左上隅に対するボタンの左上隅の現行の位置を示すx座標が返されます。この値は、データ型VARCHAR2で、また、そのフォーム・モジュールの「座標システム」プロパティに設定されている単位で返されます。
	Y_POS	キャンバスの左上隅に対するボタンの左上隅の現行の位置を示すy座標が返されます。この値は、データ型VARCHAR2で、また、そのフォーム・モジュールの「座標システム」プロパティに設定されている単位で返されます。

GET_RADIO_BUTTON_PROPERTYの例

```
/*
** Built-in:GET_RADIO_BUTTON_PROPERTY
** Example:Determine whether a given radio button is
**          displayed and has a particular visual
**          attribute.
*/
DECLARE
  it_id   Item;
  disp    VARCHAR2(5);
```

```
  va_name VARCHAR2(40);
BEGIN
  it_id := Find_Item('My_Favorite_Radio_Grp');
  disp    := Get_Radio_Button_Property( it_id, 'REJECTED', VISIBLE);
  va_name := Get_Radio_Button_Property( it_id, 'REJECTED',
         VISUAL_ATTRIBUTE);

  IF disp = 'TRUE' AND va_name = 'BLACK_ON_PEACH' THEN
    Message('You win a prize!');
  ELSE
    Message('Sorry, no luck today.');
  END IF;
END;
```

GET_RECORD_PROPERTYビルトイン

説明

指定されたブロック内の指定されたレコード行の指定されたプロパティの値を返します。パラメータは3つとも必須です。適切な定数が渡さなければ、Form Builderによってエラーが出力されます。たとえば、record_numbeパラメータには、引数として有効なレコード番号を渡す必要があります。

構文

```
FUNCTION GET_RECORD_PROPERTY
  (record_number  NUMBER,
   block_name     VARCHAR2,
   property       NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

record_number	プロパティの値を取得するブロック内のレコードを指定します。レコード番号を数値で指定してください。
block_name	ターゲット・レコードを含むブロックを指定します。
property	現行の状態を取得するプロパティを指定します。指定できるプロパティ定数は、Statusのみです。 STATUSでは、レコードが新規としてマークされ、ブロック内に変更されたレコードが存在しなければNEWが返されます。レコードが変更済みとしてマークされていれば、CHANGEDを返します。レコードが問合せとしてマークされていれば、QUERYを返します。レコードが挿入としてマークされていれば、INSERTを返します。

使用上の注意

NEWステータスが返される状況は、次表のとおりです。

	レコード・ステータス	ブロック・ステータス	フォーム・ステータス
フィールドを修正せずに作成されたレコード	NEW	<N|Q|C>	<N|Q|C>
および現ブロックのすべてのレコードが新規作成	NEW	NEW	<N|Q|C>
および現行のフォームのすべてのブロックが新規作成	NEW	NEW	NEW

次表は、実表項目、および実表や制御ブロック内の制御項目が変更されたときに、レコード、ブロック、フォームのステータスにどのように変化するかを表しています。

ブロックのタイプまたは項目のタイプを変更	変更前のレコード・ステータス	変更後のレコード・ステータス	ブロック・ステータス	フォーム・ステータス
実表ブロック内:実表項目の変更	NEW	INSERT	CHANGED	CHANGED
実表ブロック内の実表項目を変更	QUERY	CHANGED	CHANGED	CHANGED
実表ブロック内の制御項目を変更	QUERY	QUERY	<Q|C>	<Q|C>
および現ブロックに変更されたレコードがない		QUERY	QUERY	<Q|C>

および現行のフォームに変更されたブロックがない		QUERY	QUERY	QUERY
実表ブロック内:制御項目を変更	NEW	INSERT	<Q|C>	<Q|C>
制御ブロック内:制御ブロックを変更	NEW	INSERT	<Q>	<Q|C>
および現ブロックに変更されたレコードがない		INSERT	QUERY	<Q|C>
および現行のフォームに変更されたブロックがない		INSERT	QUERY	QUERY

注意: 通常、データベース項目に対して値を割り当てると、割り当てられる値が前の値と同じである場合でも、レコードのステータスがQUERYからCHANGED（またはNEWからINSERT）に変わります。項目をOUTまたはIN OUTパラメータとしてプロシージャに渡すと、そのプロシージャに対する割当てとしてカウントされます。

GET_RECORD_PROPERTYおよびシステム変数SYSTEM.RECORD_STATUS処理は、どちらも指定されたブロックでレコードの状態に戻り、ほとんどの場合は同一の状態に戻ります。ただし、特定のケースでは、結果が異なることがあります。

GET_RECORD_PROPERTYは処理シーケンスやそのレコードが現レコードかどうかに関係なくレコードのステータスを返すため、GET_RECORD_PROPERTYが返す値は常にNEW、CHANGED、QUERYまたはINSERTです。

一方、システム上に現レコードが存在しないときはSYSTEM.RECORD_STATUSが定義されないため、場合によってSYSTEM.RECORD_STATUSはNULL値を返すことがあります。たとえば、When-Clear-Blockトリガーでは、Form Builderはその処理順序のブロック・ レベルにあります。このためレポートする現レコードがなく、SYSTEM.RECORD_STATUSの値はNULLになっています。

GET_RECORD_PROPERTYの例

```
/*
** built-in:GET_RECORD_PROPERTY
** Example:Obtain the status of a record in given block
*/
BEGIN
  IF Get_Record_Property(1,'orders',STATUS) = 'NEW' AND
     Get_Record_Property(1,'customers',STATUS) = 'NEW' THEN
    Message('You must enter a customer and order first!');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

GET_RELATION_PROPERTYビルトイン

説明

指定されたリレーション・プロパティのステータスを返します。

構文

```
FUNCTION GET_RELATION_PROPERTY
  (relation_id  Relation,
   property     NUMBER);
FUNCTION GET_RELATION_PROPERTY
  (relation_name  VARCHAR2,
   property       NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

relation_id	リレーション作成時にForm Builderによってそのリレーションに割り当てられた一意のIDを指定します。FIND_RELATIONビルトインを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型はRELATION型です。
relation_name	ユーザーがリレーションを作成するときにそのリレーションに付けた名前、あるいは作成時にForm Builder によってそのリレーションに割り当てられた名前を、データ型VARCHAR2で指定します。

property	現行の状態を取得するプロパティを指定します。指定できるプロパティ定数は次のとおりです。 AUTOQUERY 「自動問合せ」リレーション・プロパティが「はい」に設定されていればVARCHAR2値TRUEが返され、「いいえ」に設定されていればFALSEが返されます。「延期」リレーション・プロパティが「はい」に設定されている場合は、別のレコードがマスター・ブロックの現レコードになるときに、Form Builderがディテール・ブロックを自動的に挿入するかどうかがこのプロパティによって決定されます。 DEFERRED_COORDINATION 「延期」リレーション・プロパティが「はい」に設定されていればVARCHAR2値TRUEが返され、「いいえ」に設定されていればFALSEが返されます。ディテール・ブロックがただちに現行のマスター・レコードを使用して調整されるか、このプロパティによってオペレータがディテール・ブロックへナビゲートするまでクリアな状態で残されるかが決定されます DETAIL_NAME 指定したマスター/ディテールのリレーションでのディテール・ブロックの名前がVARCHAR2型で返されます。 MASTER_DELETES ブロックの「レコード削除の動作」リレーション・プロパティの現在の設定を示す、NON_ISOLATED、ISOLATEDまたはCASCADINGのVARCHAR2値が返されます。 MASTER_NAME 指定したマスター/ディテールのリレーションでのマスター・ブロックの名前がVARCHAR2型で返されます。 NEXT_DETAIL_RELATION 次のディテール・リレーションが存在する場合、そのディテール・リレーションの名前がVARCHAR2型で返されます。指定されたブロックの最初のディテール・リレーションの名前を取得するには、GET_BLOCK_PROPERTYビルトインをコールします。そのようなリレーションが存在しなければ、NULLが返されます。 NEXT_MASTER_RELATION 次のリレーションが存在する場合、そのリレーションの名前がVARCHAR2型で返されます。指定されたブロックの最初のマスター・リレーションの名前を取得するには、GET_BLOCK_PROPERTYビルトインをコールします。そのようなリレーションが存在しなければ、NULLが返されます。 PREVENT_MASTERLESS_OPERATION 「マスターなし操作防止」リレーション・プロパティが「はい」に設定されていればVARCHAR2値TRUEが返され、「いいえ」に設定されていればFALSEが返されます。「はい」に設定されている場合、Form Builderでは、マスター・ブロックにマスター・レコードがなければディテール・ブロックへレコードを挿入できません。さらに、データベースからのマスター・レコードがなければディテール・ブロックでの問合せは許可されません。

GET_RELATION_PROPERTYの例

```
/*
** Built-in:GET_RELATION_PROPERTY
** Example:If the relation is not deferred, then go
**          coordinate the detail block.Otherwise, mark
**          the detail block as being in need of
**          coordination for an eventual deferred query.
*/
PROCEDURE Query_The_Details(rel_id Relation,
   detail VARCHAR2) IS
BEGIN
  IF Get_Relation_Property(rel_id, DEFERRED_COORDINATION)
   = 'FALSE' THEN
    Go_Block(detail);
    IF NOT Form_Success THEN
      RAISE Form_Trigger_Failure;
    END IF;
    Execute_Query;
  ELSE
    Set_Block_Property(detail, coordination_status,
           NON_COORDINATED);
  END IF;
End;
```

GET_REPORT_OBJECT_PROPERTYビルトイン

説明

レポート・オブジェクトのプロパティをプログラムによって取得します。

構文

```
FUNCTION GET_REPORT_OBJECT_PROPERTY
  (report_id REPORT_OBJECT,
   property NUMBER
);
FUNCTION GET_REPORT_OBJECT_PROPERTY
  (report_name VARCHAR2,
   property NUMBER
);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

report_id	レポートの一意のIDを指定します。FIND_REPORT_OBJECTを使用すると、特定のレポートのレポートIDを取得できます。
report_name	レポートの一意の名前を指定します。
property	次のいずれかの定数です。 REPORT_EXECUTION_MODE　レポート実行モード、BATCHまたはRUNTIMEの文字列値が返されます。 REPORT_COMM_MODE　レポート通信モード、SYNCHRONOUSまたはASYNCHRONOUSの文字列値が返されます。 REPORT_DESTYPE　レポート宛先タイプ、PREVIEW、FILE、PRINTER、MAIL、CACHEまたはSCREENの文字列値が返されます。 REPORT_FILENAME　レポート・ファイル名の文字列値が返されます。REPORT_SOURCE_BLOCKレポート・ソース・ブロック名の文字列値が返されます。 REPORT_QUERY_NAME　レポート問合せ名の文字列値が返されます。 REPORT_DESNAME　レポート宛先名の文字列値が返されます。 REPORT_DESFORMAT　レポート宛先フォーマットの文字列値が返されます。 REPORT_SERVER　レポート・サーバー名の文字列値が返されます。 REPORT_OTHER　ユーザー指定による他のレポート・プロパティの文字列が返されます。

使用上の注意

- GET_REPORT_OBJECT_PROPERTYは、すべてのプロパティの文字列値を返します。それとは反対に、SET_REPORT_OBJECT_PROPERTYでは、定数値または文字列値を使用してプロパティを設定します。値の型は、設定されている特定のプロパティによって異なります。

GET_REPORT_OBJECT_PROPERTYの例

```
DECLARE
   repid REPORT_OBJECT;
   report_prop VARCHAR2(20);
BEGIN
   repid := find_report_object('report4');
```

```
    report_prop := get_report_object_property(repid,
         REPORT_EXECUTION_MODE);
    message('REPORT EXECUTION MODE PROPERTY IS ' || report_prop);
    report_prop  := get_report_object_property(repid, REPORT_COMM_MODE);
    message('REPORT COMM_MODE PROPERTY IS ' || report_prop);
    report_prop  := get_report_object_property(repid, REPORT_DESTYPE);
    message('REPORT DESTYPE PROPERTY IS ' || report_prop);
    report_prop := get_report_object_property(repid, REPORT_FILENAME);
    message('REPORT_FILENAME PROPERTY IS ' || report_prop);
END;
```

GET_TAB_PAGE_PROPERTYビルトイン

説明

指定されたタブ・ページのプロパティの値を返します。

構文

```
FUNCTION GET_TAB_PAGE_PROPERTY
  (tab_page_id  TAB_PAGE,
   property     NUMBER);
FUNCTION GET_TAB_PAGE_PROPERTY
  (tab_page_name  VARCHAR2,
   property       NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

tab_page_id	タブ・ページ作成時にForm Builderによってそのページに割り当てられた重複しないID。FIND_TAB_PAGEビルトインを使用すると、このIDをデータ型TAB_PAGEの変数に返すことができます。
tab page_name	ユーザーがタブ・ページを作成したときにそのページに付けた名前。注意: 同じフォーム・モジュールに同じ名前のタブ・ページが2つある場合は、キャンバス名とタブ・ページ名（例: CVS_1.TAB_PG_1）を指定してください。

property	指定したタブ・ページで値を取り出すプロパティ。設定できるプロパティは次のとおりです。
	BACKGROUND_COLOR　オブジェクトのバックグラウンド領域のカラー。
	CANVAS_NAME　タブ・ページが属するキャンバスの名前がVARCHAR2型で返されます。
	ENABLED　タブ・ページが変更可能であればVARCHAR2文字列TRUEが返され、使用不可であれば（たとえばグレーで表示され、使用できない場合）FALSEが返されます。
	FILL_PATTERN　オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。
	FONT_NAME　オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。
	FONT_SIZE　ポイント数で指定されるフォントのサイズ。
	FONT_SPACING　フォントの幅つまり文字間のスペース（カーニング）。
	FONT_STYLE　フォントのスタイル。
	FONT_WEIGHT　フォントの太さ。
	FOREGROUND_COLOR　オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。
	LABEL　タブ・ページのラベル用のVARCHAR2文字列が返されます。
	VISIBLE　タブ・ページが表示される場合はVARCHAR2文字列TRUEが返され、表示されない場合はFALSEが返される。タブ・ページが現在画面にマップされていれば、そのページが他のタブ・ページの後ろに隠れていても、そのページは参照可能とみなされます。
	VISUAL_ATTRIBUTE　現在有効な可視属性の名前が返されます。タブ・ページに名前付き可視属性が割り当てられていない場合は、カスタム可視属性であればCUSTOM がデフォルトの可視属性であればDEFAULTが返されます。
	WHITE_ON_BLACK　白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。

GET_TAB_PAGE_PROPERTYの例

```
/* Use FIND_TAB_PAGE and GET_TAB_PAGE_PROPERTY to check
** if a tab page is enabled:
*/
DECLARE
  tp_id  TAB_PAGE;
  live   VARCHAR2(32);
```

```
BEGIN
  tp_id := FIND_TAB_PAGE('tab_page_1');
  live  := GET_TAB_PAGE_PROPERTY(tp_id, enabled);
END;
```

GET_TREE_NODE_PARENTビルトイン

説明

指定されたノードの親を返します。

構文

```
FUNCTION GET_TREE_NODE_PARENT
   (item_name VARCHAR2,
node NODE);
FUNCTION GET_TREE_NODE_PARENT
   (item_id ITEM,
node NODE);
```

戻り値

NODE

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
node	有効なノードを指定します。

GET_TREE_NODE_PARENTの例

```
/*
** Built-in:GET_TREE_NODE_PARENT
*/

-- This code could be used in a WHEN-TREE-NODE-SELECTED
-- trigger to locate the parent of the node that was
-- clicked on.

DECLARE
   htree        ITEM;
   parent_node   FTREE.NODE;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Get the parent of the node clicked on.
   parent_node :=
   Ftree.Get_Tree_Node_Parent(htree, :SYSTEM.TRIGGER_NODE);

   ...
END;
```

GET_TREE_NODE_PROPERTYビルトイン

説明

階層ツリー・ノードの指定されたプロパティの値を返します。

構文

```
FUNCTION GET_TREE_NODE_PROPERTY
  (item_name VARCHAR2,
   node NODE,
   property NUMBER);
FUNCTION GET_TREE_NODE_PROPERTY
  (item_id ITEM,
   node NODE,
property NUMBER);
```

戻り値

VARCHAR2

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
node	有効なノードを指定します。
property	次のいずれかのプロパティを指定します。 NODE_STATE　階層ツリー・ノードの状態が返されます。EXPANDED_NODE、COLLAPSED_NODEまたはLEAF_NODEのいずれかです。 NODE_DEPTH　階層ツリー・ノードのネスト・レベルが返されます。 NODE_LABEL　ラベルが返されます。 NODE_ICON　アイコン名が返されます。 NODE_VALUE　階層ツリー・ノードの値が返されます。

GET_TREE_NODE_PROPERTYの例

```
/*
** Built-in:GET_TREE_NODE_PROPERTY
*/

-- This code could be used in a WHEN-TREE-NODE-SELECTED
-- trigger to return the value of the node that was
-- clicked on.

DECLARE
   htree         ITEM;
   node_value    VARCHAR2(100);
BEGIN
  -- Find the tree itself.
  htree := Find_Item('tree_block.htree3');

  -- Find the value of the node clicked on.
  node_value := Ftree.Get_Tree_Node_Property(htree, :SYSTEM.TRIGGER_NODE,
  Ftree.NODE_VALUE);
```

```
   ...
END;
```

GET_TREE_PROPERTYビルトイン

説明

指定された階層ツリーのプロパティ値を返します。

構文

```
FUNCTION GET_TREE_PROPERTY
  (item_name VARCHAR2,
   property NUMBER);
FUNCTION GET_TREE_PROPERTY
  (item_id ITEM,
   property NUMBER);
```

戻り値

VARCHAR2

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_name	オブジェクトの作成時に付けた名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。

property	次のいずれかのプロパティを指定します。 DATASOURCE　Form Builder内またはSET_TREE_PROPERTYビルトインを使用して、階層ツリーを最初に生成するときに使用したソースが返されます。そのどちらでも設定されなかった場合、EXTERNALが返されます。 RECORD_GROUP　Form Builder内でまたはSET_TREE_PROPERTYビルトインを使用して、階層ツリーを最初に生成するときに使用したRECORDGROUPが返されます。NULL文字列の場合があります。 QUERY_TEXT　Form Builder内でまたはSET_TREE_PROPERTYビルトインを使用して、階層ツリーを最初に生成するときに使用した問合せのテキストが返されます。NULL文字列の場合があります。 NODE_COUNT　階層ツリーのデータ・セット内の行数が返されます。 SELECTION_COUNT 階層ツリー内で選択されている行の数が返されます。 ALLOW_EMPTY_BRANCHES 文字列、TRUEまたはFALSEが返されます。 ALLOW_MULTI-SELECT 文字列、TRUEまたはFALSEが返されます。

使用上の注意

データソース、RECORD_GROUPおよびQUERY_TEXTによって返される値は、ツリーの現行の値または状態を反映しているとは限りません。返される値は、Form Builderに設定された値で、SET_TREE_PROPERTYビルトインを使用して設定された値ではありません。

GET_TREE_PROPERTYの例

```
/*
** Built-in:GET_TREE_PROPERTY
*/

-- This code could be used to find out how many nodes are
-- in a given tree.

DECLARE
   htree        ITEM;
   node_count    NUMBER;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Get the node count of the tree.
   node_count := Ftree.Get_Tree_Property(htree, Ftree.NODE_COUNT);

   ...
```

```
END;
```

GET_TREE_SELECTIONビルトイン

説明

*selection*によって示されるデータ・ノードを返します。selectionは選択されたノード・リストの索引です。

構文

```
FUNCTION GET_TREE_SELECTION
  (item_name VARCHAR2,
   selection NUMBER);
FUNCTION GET_TREE_SELECTION
  (item_id ITEM,
   selection NUMBER);
```

戻り値

FTREE.NODE

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
selection	選択した単一ノードを指定します。

GET_TREE_SELECTIONの例

```
/*
** Built-in:GET_TREE_SELECTION
*/

-- This code will process all tree nodes marked as selected.See the
-- Ftree.Set_Tree_Selection built-in for a definition of "selected".
```

```
DECLARE
   htree         ITEM;
   num_selected NUMBER;
   current_node FTREE.NODE;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Find the number of nodes marked as selected.
   num_selected := Ftree.Get_Tree_Property(htree,
   Ftree.SELECTION_COUNT);

   -- Loop through selected nodes and process them.If you are deleting
   -- nodes, be sure to loop in reverse!
   FOR j IN 1..num_selected LOOP
   current_node := Ftree.Get_Tree_Selection(htree, j);
   ...
   END LOOP;
END;
```

GET_VA_PROPERTYビルトイン

説明

指定されたプロパティの可視属性プロパティ値を返します。

構文

```
FUNCTION GET_VA_PROPERTY
  (va_id VISUALATTRIBUTE
   property NUMBER);
FUNCTION GET_VA_PROPERTY
  (va_name VARCHAR2
   property NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

va_id	可視属性の作成時にForm Builderによって割り当てられる一意のID。データ型はVISUALATTRIBUTEです。
va_name	可視属性の作成時に付けた名前。データ型はVARCHAR2です。
property	次のいずれかのプロパティを指定します。 BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE ポイント数で指定されるフォントのサイズ。たとえば、8ptは800です。 FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE フォントのスタイル。 FONT_WEIGHT フォントの太さ。 FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。

GET_VAR_BOUNDSビルトイン

説明

OLEバリアントの配列の範囲を取得します。

構文

```
PROCEDURE GET_VAR_BOUNDS
  (var OLEVAR, bounds OLE_SAFEARRAYBOUNDS);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

var	バリアント。
bounds	配列の範囲によって生成されるPL/SQL表。 このパラメータの内容とレイアウトの詳細は、以下のOLE用配列型のサポートを参照してください。

配列の型またはサブタイプ	定義
VT_TYPE	PLS_INTEGER型のフォームにおけるサブタイプです。詳細はOLEバリアント型を参照してください。
OLE_SAFEARRAYBOUND	次の2つのフィールドを含むレコードです。 ■ Count PLS_INTEGER Countは配列内の項目の数。 ■ lbound PLS_INTEGER lboundのデフォルト値は1。
OLE_SAFEARRAYBOUNDS	OLE_SAFEARRAYBOUND型のレコードを含む表です。 表走査はINDEX BY BINARY_INTEGER。 この配列型は、CREATE_VARファンクションおよびGET_VAR_BOUNDSファンクションで使われます。
ITEMS_IN_BLOCK	VARCHAR2(30)型のフォームを持つ表です。 表走査はINDEX BY BINARY_INTEGER。 この型はTABLE_FROM_BLOCKファンクションで使われます。

GET_VAR_DIMSビルトイン

説明

OLEバリアントが配列かどうかを判別し、もし配列である場合は配列の次元数を取得します。

構文

```
FUNCTION GET_VAR_DIMS
  (var OLEVAR)
RETURN vardims PLS_INTEGER;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

バリアントが配列でない場合、ゼロ(0)の値。バリアントが配列の場合、戻り値は配列の次元数です。

パラメータ

var	バリアント。

GET_VAR_TYPEビルトイン

説明

OLEバリアントの型を取得します。

構文

```
FUNCTION GET_VAR_TYPE
  (var OLEVAR)
RETURN vartype VT_TYPE;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

変数の型

パラメータ

var	バリアント。
vartype	バリアントの型。

使用上の注意

サポートされるVT_TYPEのリストは、以下のOLEバリアントの型を参照してください。

OLE定義のバリアント型	Oracle Developerにおける解釈
VT_BLOB	VT_UNKNOWN
VT_BLOB_OBJECT	VT_UNKNOWN
VT_BOOL	ブール
VT_BSTR	文字列
VT_BYREF	16384。これは「へのポインタ」を示すために任意のタイプに追加されることがあります
VT_CARRAY	- 未サポート

OLE定義のバリアント型	Oracle Developerにおける解釈
VT_CF	- 未サポート
VT_CLSID	- 未サポート
VT_CY	- 未サポート
VT_DATE	日付
VT_DISPATCH	OLEオートメーション・コールが可能なOLEオブジェクト・ポインタ
VT_EMPTY	初期化されていないメンバー
VT_ERROR	エラー・フィールド
VT_FILETIME	VT_DATE
VT_HRESULT	VT_ERROR
VT_INT	整数
VT_I1	符号つき1バイト整数
VT_I2	符号つき2バイト整数
VT_I4	符号つき4バイト整数
VT_I8	- 未サポート
VT_LPSTR	VT_BSTR
VT_LPWSTR	VT_BSTR
VT_NULL	NULL
VT_PTR	タイプ分けされていないポインタ
VT_R4	4バイト浮動小数点数
VT_R8	8バイト浮動小数点数
VT_STORAGE	VT_UNKNOWN
VT_STORED_OBJECT	VT_UNKNOWN
VT_STREAM	VT_UNKNOWN
VT_STREAMED_OBJECT	VT_UNKNOWN
VT_UINT	符号なし4バイト整数
VT_UI1	符号なし1バイト整数
VT_UI2	符号なし2バイト整数
VT_UI4	符号なし4バイト整数
OLE定義のバリアント型	Oracle Developerにおける解釈
VT_UNKNOWN	OLEオブジェクト・ポインタ
VT_USERDEFINED	型付けされたデータ
VT_VARIANT	任意の型

GET_VERB_COUNTビルトイン

説明

OLEサーバーが認識する動詞の数を返します。OLEオブジェクト上で実行可能なアクションをOLE動詞で指定しますが、使用できる動詞の数はOLEサーバーによって異なります。動詞の数は、VARCHAR2文字列として返されるので、各動詞の索引と名前を決定する際に使用するNUMBER型に変換する必要があります。適切な型を持つ変数を定義して戻り値を受け取ります。

構文

```
FUNCTION GET_VERB_COUNT
  (item_id  Item);
FUNCTION GET_VERB_COUNT
  (item_name VARCHAR2);
```

戻り値

VARCHAR2

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。

GET_VERB_COUNTの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効。

GET_VERB_COUNTの例

```
/*
** Built-in:GET_VERB_COUNT
** Example:Obtains the number of verbs that the OLE server
** issues and recognizes when executed from the OLE container.
** Trigger:When-Button-Pressed
*/
```

```
DECLARE
 item_id  ITEM;
 item_name VARCHAR(25) := 'OLEITM';
 verb_cnt_str VARCHAR(20);
 verb_cnt NUMBER;
 verb_name VARCHAR(20);
 loop_cntr NUMBER;
BEGIN
 item_id := Find_Item(item_name);
 IF Id_Null(item_id) THEN
  message('No such item:'||item_name);
 ELSE
  verb_cnt_str := Get_Verb_Count(item_id);
  verb_cnt := TO_NUMBER(verb_cnt_str);
  FOR loop_cntr in 1..verb_cnt LOOP
    verb_name := Get_Verb_Name(item_id,loop_cntr);
    IF verb_name = 'Edit' THEN
   Exec_Verb(item_id,verb_name);
    END IF;
  END LOOP;
 END IF;
END;
```

GET_VERB_NAMEビルトイン

説明

指定の動詞索引と対応する動詞名を返します。OLEオブジェクト上で実行可能なアクションをOLE動詞で指定すると、OLE動詞のそれぞれに対応するOLE動詞索引が付加されます。適切な型を持つ変数を定義して戻り値を受け取ります。

構文

```
FUNCTION GET_VERB_NAME
  (item_id    Item,
   verb_index  VARCHAR2);
FUNCTION GET_VERB_NAME
  (item_name   VARCHAR2,
   verb_index  VARCHAR2);
```

戻り値

VARCHAR 2

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型はVARCHAR2です。
verb_index	動詞の数値索引を指定します。この値を取得するにはFIND_OLE_VERBビルトインを使用します。索引のデータ型はVARCHAR2です。

GET_VERB_NAMEの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効。

GET_VERB_NAMEの例

EXEC_VERBおよびGET_VERB_COUNTを参照してください。

GET_VIEW_PROPERTYビルトイン

説明

指定されたキャンバスの指定されたプロパティの設定値を返します。

構文

```
FUNCTION GET_VIEW_PROPERTY
  (view_id   ViewPort,
   property  NUMBER);
FUNCTION GET_VIEW_PROPERTY
  (view_name  VARCHAR2,
   property   NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

view_id	オブジェクト作成時にForm Builder によってキャンバスに割り当てられた一意のIDを指定します。FIND_VIEWのビルトインを使用して、そのIDを適切なタイプの変数に戻します。このIDのデータ型はViewPort型です。
view_name	ユーザーがオブジェクト定義時にそのオブジェクトに与えた名前を指定します。
property	指定したキャンバスで状態を取得するプロパティを指定します。使用例のように、状態を取得するプロパティごとにGET_VIEW_PROPERTYビルトインのコールをする必要があります。プロパティに指定できる定数は次のとおりです。 DIRECTION　両方向オブジェクトのレイアウト方向が返されます。有効な戻り値はRIGHT_TO_LEFT、LEFT_TO_RIGHTです。 HEIGHT　ビューの高さが返されます。コンテント・ビューでは、ビューの高さはビューが現在表示されているウィンドウの実際の高さです。高さの単位は、そのフォーム・モジュールに対して「座標システム」プロパティをどのように定義したかによって異なります。 VIEWPORT_X_POS　スタック・キャンバスの場合は、ウィンドウの現コンテント・キャンバスの左上隅に対するビューの左上隅の現行の位置を示すx座標が返されます。コンテント・ビューの場合は、0が返されます。この値は、VARCHAR2のデータ型で返され、フォーム・モジュールの「座標システム」プロパティに設定された単位で返されます。 VIEWPORT_Y_POS　スタック・キャンバスの場合は、ウィンドウの現コンテント・キャンバスの左上隅に対するビューの左上隅の現行の位置を示すy座標が返されます。コンテント・ビューの場合は、0が返されます。この値は、VARCHAR2のデータ型で返され、フォーム・モジュールの「座標システム」プロパティに設定された単位で返されます。

property（続き）	
	VIEWPORT_X_POS_ON_CANVAS キャンバスの左上隅に対するビューの左上隅の現行の位置を示すx座標が返されます。この値は、データ型VARCHAR2で、また、そのフォーム・モジュールの「座標システム」プロパティに設定されている単位で返されます。
	VIEWPORT_Y_POS_ON_CANVAS キャンバスの左上隅に対するビューの左上隅の現行の位置を示すy座標が返されます。この値は、データ型VARCHAR2で、また、そのフォーム・モジュールの「座標システム」プロパティに設定されている単位で返されます。
	VISIBLE ビューが表示される場合は、VARCHAR2文字列TRUEが返され、表示されない場合はFALSEが返されます。1） ウィンドウの一番手前に表示されている場合、または 2） 他のビューに一部のみが隠れている場合に、ビューは参照可能とみなされます。また、1）コンテント・ビューの後ろに隠れたスタック・ビューがある場合、または 2） 特定のスタック・ビューの後ろにビュー全体が隠れている場合に、ビューは参照不能とみなされます。なお、このプロパティは、現行のウィンドウの表示状態とは無関係です。したがって、ウィンドウが現在隠されていたりアイコン化されていたりしても、ビューは参照可能とみなされることがあります。
	WIDTH ビューの幅が返されます。コンテント・ビューでは、ビューの幅はビューが現在表示されているウィンドウの実際の幅です。高さの単位は、そのフォーム・モジュールに対して「座標システム」プロパティをどのように定義したかによって異なります。
	WINDOW_NAME キャンバスが表示されているウィンドウの名前が返されます。

GET_VIEW_PROPERTYの例

```
/*
** Built-in:GET_VIEW_PROPERTY
** Example:Use the Width, and display position of one
**          stacked view (View1) to determine where to
**          position another one (View2) immediately to its
**          right.
*/
PROCEDURE Anchor_To_Right( View2 VARCHAR2, View1 VARCHAR2) IS
  vw_id1 ViewPort;
  vw_id2 ViewPort;
  x     NUMBER;
  y     NUMBER;
  w     NUMBER;
BEGIN
  /*
```

```
  ** Find View1 and get its (x,y) position, width
  */
  vw_id1 := Find_View(View1);
  x      := Get_View_Property(vw_id1,VIEWPORT_X_POS);
  y      := Get_View_Property(vw_id1,VIEWPORT_Y_POS);
  w      := Get_View_Property(vw_id1,WIDTH);
  /*
  ** Anchor View2 at (x+w,y+h)
  */
  vw_id2 := Find_View(View2);
  Set_View_Property(vw_id2,VIEWPORT_X_POS, x+w );
  Set_View_Property(vw_id2,VIEWPORT_Y_POS, y );
END;
```

GET_WINDOW_PROPERTYビルトイン

説明

指定されたウィンドウの指定されたウィンドウ・プロパティの現行の設定値を返します。

構文

```
FUNCTION GET_WINDOW_PROPERTY
  (window_id  Window,
   property   NUMBER);
FUNCTION GET_WINDOW_PROPERTY
  (window_name  VARCHAR2,
   property     NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

使用上の注意

Microsoft Windows上では、定数FORMS_MDI_WINDOWを使用してMDIアプリケーション・ウィンドウを参照することができます。

パラメータ

window_id	Form Builderによりウィンドウ作成時に割り当てられる一意の ID を指定します。FIND_WINDOWのビルトインを使用して、そのIDを適切な型の変数に戻します。このIDのデータ型はWINDOW型です。
window_name	ウィンドウの作成時に付けた名前を指定します。
property	FIND_WINDOW　ビルトインの使用例のように、値を取得するプロパティごとにGET_WINDOW_PROPERTYビルトインをコールする必要があります。プロパティの現行の値や状態を取得する次の定数をどれか1つ指定します。 BACKGROUND_COLOR　オブジェクトのバックグラウンド領域のカラー。 DIRECTION　両方向オブジェクトのレイアウト方向が返されます。有効な戻り値はRIGHT_TO_LEFT、LEFT_TO_RIGHTです。 FILL_PATTERN　オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME　オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE　ポイント数で指定されるフォントのサイズ。 FONT_SPACING　フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE　フォントのスタイル。 FONT_WEIGHT　フォントの太さ。 FOREGROUND_COLOR　オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 HEIGHT　ウィンドウの高さが返されます。 HIDE_ON_EXIT　ウィンドウの「終了時に隠す」プロパティが「はい」に設定されていればVARCHAR2値TRUEが返され、「いいえ」に設定されていればFALSEが返されます。 ICON_NAME　最小化時にウインドウ項目に関連するアイコン・リソースのファイル名が返されます。 TITLE　ウィンドウのタイトルが返されます。 VISIBLE　ビューが表示される場合は、VARCHAR2値TRUEが返され、表示されない場合はFALSEが返されます。ウィンドウが現在画面にマップされていれば、ウィンドウ全体が別のウィンドウの後ろに隠れていたりアイコン化（最小化）されていたりしても、ウィンドウは参照可能とみなされます。

property（続き）	WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	WIDTH	ウィンドウの幅が返されます。
	WINDOW_HANDLE	オブジェクトの参照に使用される一意の内部VARCHAR2定数が返されます。プラットフォームがMicrosoft Windowsでない場合、数値 0 が返されます。
	WIDOW_SIZE	ウィンドウの幅と高さがコンマで区切られた文字列で返されます。
	WINDOW_STATE	ウィンドウの現行の表示状態が返されます。ウィンドウの表示状態は、VARCHAR2文字列NORMAL、MAXIMIZEまたはMINIMIZEです。
	X_POS	画面上でのウィンドウの現行の表示位置を示すx座標が返されます。
	Y_POS	画面上でのウィンドウの現行の表示位置を示すy座標が返されます。

GO_BLOCKビルトイン

説明

GO_BLOCKビルトインは、指定されたブロックにナビゲートします。ターゲット・ブロックが入力不可ブロックの場合は、エラーが発生します。

構文

```
PROCEDURE GO_BLOCK
  (block_name  VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

block_name	ユーザーがブロックを定義したときにそのブロックに付けた名前を指定します。その名前のデータ型はVARCHAR2です。

GO_BLOCKの例

```
/*
** Built-in:GO_BLOCK
** Example:Navigate to a block by name.Make sure to check
**          that the Go_Block succeeds by checking FORM_SUCCESS.
*/
BEGIN
  IF :Global.Flag_Indicator = 'NIGHT' THEN
    Go_Block('Night_Schedule');
    /*
    ** One method of checking for block navigation success.
    */
    IF NOT FORM_SUCCESS THEN
      RAISE Form_Trigger_Failure;
    END IF;
  ELSIF :Global.Flag_Indicator = 'DAY' THEN
    Go_Block('Day_Schedule');
    /*
    ** Another way of checking that block navigation
    ** succeeds.If the block the cursor is in hasn't
    ** changed after a block navigation, something went
    ** wrong.This method is more reliable than simply
    ** checking FORM_SUCCESS.
    */
    IF :System.Trigger_Block = :System.Cursor_Block THEN
      RAISE Form_Trigger_Failure;
    END IF;
  END IF;
  Execute_Query;
  Go_Block('Main');
END;
```

GO_FORMビルトイン

説明

マルチフォーム・アプリケーションで、現フォームから指定されたターゲット・フォームにナビゲートします。GO_FORMビルトインを使用してナビゲートすると、妥当性検査は一切行われず、また、ナビゲーションを開始したフォームに対して起動されるWHEN-WINDOW-DEACTIVATEDとターゲット・フォーム内のターゲット・ウィンドウに対して起動されるWHEN-WINDOW-ACTIVATEDを除けば、トリガーもまったく起動されません。

まだオープンされていないフォームにナビゲートしようとすると、エラーが発生します。

構文

```
PROCEDURE GO_FORM
  (form_id  FORMMODULE);
PROCEDURE GO_FORM
  (form_name  VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

form_id	実行時にインスタンス化されたときにフォームに動的に割り当てられる、一意のID。FIND_FORMビルトインを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型はFORMMODULEです。
form_name	ターゲット・フォームの名前。この名前のデータ型はVARCHAR2です。GO_FORMビルトインは、.fmxファイルの名前ではなく、フォームのモジュール名を検索します。

GO_FORMの制限事項

CALL_FORMがある別のフォームを起動した結果として現在使用禁止となっているフォームはターゲット・フォームにはできません。

GO_ITEM ビルトイン

説明

GO_ITEMを使用して、指示された項目にナビゲートします。GO_ITEMビルトインは、ターゲット項目の「キーボードで移動可能」プロパティが「いいえ」に設定されている場合も使用できます。

構文

```
PROCEDURE GO_ITEM
  (item_id  Item);
PROCEDURE GO_ITEM
  (item_name  VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。IDのデータ型はITEMです。
item_name	ユーザーが設計時に項目の名前として定義した文字列を指定します。その名前のデータ型はVARCHAR2です。

GO_ITEMの制限事項

```
GO_ITEM('emp.ename');
```

- 「問合せ入力」モードで、別のブロック内の項目にナビゲートする場合は、GO_ITEMを使用できません。
- 表示項目やチャート項目などのナビゲート不可能な項目にナビゲートする場合は、GO_ITEMを使用できません。

GO_ITEMの例

```
/*
** Built-in:GO_ITEM
** Example:Invoke a dialog window by navigating to
**          an item which is on the canvas which the window
**          displays.
*/
PROCEDURE Open_Preference_Dialog IS
BEGIN
 Go_Item('pref_dialog.printer_name');
END;
```

GO_RECORDビルトイン

説明

指定されたレコード番号のレコードにナビゲートします。

構文

```
PROCEDURE GO_RECORD
  (record_number  NUMBER);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

record_number	**PL/SQLで数値に評価される任意の整数値を指定します。この整数値には、TO_NUMBER(:SYSTEM.TRIGGER_RECORD) + 8などのシステム変数に対するコールから導出された値が含まれます。**

システム変数SYSTEM.CURSOR_RECORDまたはSYSTEM.TRIGGER_RECORDを使用し、レコードの順序番号を調べることができます。

GO_RECORDの制限事項

- 問合せがオープンされており、指定されたレコード番号がすでにフェッチされたレコード数より大きい場合は、このビルトイン・コールを満足するためにForm Builderによってさらにレコードがフェッチされます。

GO_RECORDの例

```
/*
** Built-in:GO_RECORD
** Example:Navigate to a record in the current block
**          by record number.Also see FIRST_RECORD and
**          LAST_RECORD built-ins.
*/
BEGIN
  Go_Record( :control.last_record_number );
END;
```

HELPビルトイン

説明

現項目のヒント・メッセージをメッセージ行に表示します。ヒント・メッセージが既に表示されている場合、HELPではその項目の詳細ヘルプ画面が表示されます。

構文

```
PROCEDURE HELP;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

HELPの例

```
/*
** Built-in:HELP
** Example:Gives item-level hint/help.
*/
BEGIN
  Help;
END;
```

HIDE_MENUビルトイン

説明

文字モードのプラットフォームで現メニューが表示中であればそれを消して、そのメニューの背後に隠れているフォーム表示をすべて表示します。このメニューは、SHOW_MENUビルトインの起動時、またはオペレータが[Menu]を押したときに再表示されます。

構文

```
PROCEDURE HIDE_MENU;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

HIDE_MENUの例

```
/*
** Built-in:HIDE_MENU
** Example:Hides the menu from view on character-mode or
**         block-mode devices
*/
BEGIN
  Hide_Menu;
END;
```

HIDE_VIEWビルトイン

説明

指示されたキャンバスを隠します。

構文

```
PROCEDURE HIDE_VIEW
  (view_id  ViewPort);
PROCEDURE HIDE_VIEW
  (view_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

説明

指示されたキャンバスを隠します。

パラメータ

view_id	Form Builderによりビュー作成時に割り当てられる一意のIDを指定します。FIND_VIEWのビルトインを使用して、そのIDを適切なタイプの変数に戻します。このIDのデータ型はVIEWPORT型です。
view_name	ユーザーがビュー作成時にそのビューに与えた名前を指定します。

HIDE_VIEWの例

```
/*
** Built-in:HIDE_VIEW
** Example:Programmatically dismiss a stacked view from the
**          operator's sight.
*/
PROCEDURE Hide_Button_Bar IS
BEGIN
  Hide_View('Button_Bar');
END;
```

HIDE_WINDOWビルトイン

説明

ウィンドウを隠します。HIDE_WINDOWは、SET_WINDOW_PROPERTYをコールすることによりVISIBLEを「No」に設定するのと同じです。

構文

```
PROCEDURE HIDE_WINDOW
(window_id  Window);
PROCEDURE HIDE_WINDOW
(window_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

window_id	Form Builderによりウィンドウ作成時に割り当てられる一意のIDを指定します。FIND_WINDOWのビルトインを使用して、そのIDを適切な型の変数に戻します。このIDのデータ型はWINDOW型です。
window_name	ウィンドウの作成時に付けた名前を指定します。

HIDE_WINDOWの例

```
/*
** Built-in:HIDE_WINDOW
** Example:When a main window is closed, hide other
**          "subordinate" windows automatically.To
**          establish this window hierarchy we might define
**          a static record group in the form called
**          'WINDOW_HIERARCHY' with a structure of:
**
**             Parent_Window     Child_Window
**             -------------     -------------
**               MAIN             DETAIL1
**               MAIN             DETAIL2
**               DETAIL1           DETAIL3
**               DETAIL1           DETAIL4
**               DETAIL2           DETAIL5
**               DETAIL3           DETAIL6
**
**         We also have to make sure we navigate to some
**         item not on any of the canvases shown in the
**         windows we are closing, or else that window
**         will automatically be re-displayed by forms
**         since it has input focus.
*/
PROCEDURE Close_Window( wn_name VARCHAR2,
         dest_item VARCHAR2 ) IS
  rg_id       RecordGroup;
  gc_parent    GroupColumn;
  gc_child    GroupColumn;
  the_Rowcount NUMBER;
  /*
  ** Local function called recursively to close children at
  ** all levels of the hierarchy.
  */
  PROCEDURE Close_Win_With_Children( parent_win VARCHAR2 ) IS
    the_child  VARCHAR2(40);
    the_parent VARCHAR2(40);
```

```
  BEGIN
    FOR j IN 1..the_Rowcount LOOP
      the_parent := Get_Group_Char_Cell(gc_parent,j);
      /* If we find a matching parent in the table */
      IF UPPER(the_parent) = UPPER(parent_win) THEN
    the_child := Get_Group_Char_Cell(gc_child,j);
    /*
    ** Close this child and any of its children
    */
    Close_Win_With_Children( the_child );
      END IF;
    END LOOP;
    /*
    ** Close the Parent
    */
    Hide_Window( parent_win );
  END;
BEGIN
  /*
  ** Setup
  */
  rg_id       := Find_Group('WINDOW_HIERARCHY');
  gc_parent    := Find_Column('WINDOW_HIERARCHY.PARENT_WINDOW');
  gc_child     := Find_Column('WINDOW_HIERARCHY.CHILD_WINDOW');
  the_Rowcount := Get_Group_Row_Count(rg_id);
  /* Close all the child windows of 'wn_name' */
  Close_Win_With_Children( wn_name );
  /* Navigate to the Destination Item supplied by the caller */
  Go_Item( dest_item );
END;
```

HOSTビルトイン

説明

指示されたオペレーティング・システム・コマンドを実行します。

構文

```
PROCEDURE HOST
  (system_command_string  VARCHAR2);
```

```
PROCEDURE HOST
  (system_command_string  VARCHAR2,
   screen_action          NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

system_command_string	**特定のオペレーティング・システムに渡すシステム・コマンドを指定します。**
screen_action	**次のいずれかの定数を指定します。** **パラメータなし** Form Builder**に次のことを実行するよう指定します。** **■スクリーンを消去します。** **■オペレータにそのコマンドから戻ることを求めます。** NO_PROMPT Form Builder**に次のことを実行するよう指定します。** **■スクリーンを消去します（オペレータにそのコマンドから戻ることを求めません）。** NO_SCREEN Form Builder**に次のことを実行するよう指定します。** **■スクリーンを消去しません。** **■オペレータに、システム・コマンドから戻るよう求めません。（この「HOST」コマンドは、NO_SCREEN パラメータの使用時、スクリーンに出力を送信しないでください。）**

使用上の注意

- このscreen_actionパラメータは、「Host」コマンドの出力がフォームと同一ウィンドウに表示される文字モードで実行されるアプリケーションにのみ関連します。GUIアプリケーションでは、「Host」コマンドの出力は別のウィンドウに表示されます。
- Microsoft Windows NTのコマンド・インタープリタはcmdですが、Windows 95上ではcommandとなるので注意してください。HOSTビルトインを使用して外部コマンドを実行する前に、必ずオペレーティング・システムをチェックして、適切なコマンド文字列を渡してください。
- Microsoft Windows NTでは、HOSTを使用して16ビット・アプリケーションを実行するときに、FORM_SUCCESSビルトインは、そのアプリケーションが正常に実行されても失敗しても、TRUEを戻します。これは、Microsoft Win32の問題です。32ビット・アプリケーションおよびOSコマンドは、正常に実行されるとTRUEを正しく戻し、失敗するとFALSEを戻します。無効なコマンドは、FALSEを戻します。
- Windows 95プラットフォーム上では、FORM_SUCCESSビルトインは失敗したHOSTコマンド

に対して常にTRUEを戻します。この中には、command.comまたはOSファンクションに対するコール、任意の16ビットDOSアプリケーションまたはGUIアプリケーション、または無効なコマンドが含まれます。32ビット・アプリケーションは、正常に実行されるとTRUEを正しく戻し、失敗するとFALSEを戻します。

HOSTの例

```
/*
** built-in:HOST
** Example:Execute an operating system command in a
**          subprocess or subshell.Uses the
**          'Get_Connect_Info' procedure from the
**          GET_APPLICATION_PROPERTY example.
*/
PROCEDURE Mail_Warning( send_to VARCHAR2) IS
  the_username VARCHAR2(40);
  the_password VARCHAR2(40);
  the_connect  VARCHAR2(40);
  the_command  VARCHAR2(2000);
BEGIN
  /*
  ** Get Username, Password, Connect information
  */
  Get_Connect_Info(the_username,the_password,the_connect);
  /*
  ** Concatenate together the static text and values of
  ** local variables to prepare the operating system command
  ** string.
  */
  the_command := 'orasend '||
      ' to='||send_to||
      ' std_warn.txt '||
      ' subject="## LATE PAYMENT ##"'||
      ' user='||the_username||
      ' password='||the_password||
      ' connect='||the_connect;

  Message('Sending Message...', NO_ACKNOWLEDGE);
  Synchronize;
  /*
  ** Execute the command string as an O/S command The
  ** NO_SCREEN option tells forms not to clear the screen
  ** while we do our work at the O/S level "silently".
  */
```

```
  Host( the_command, NO_SCREEN );
  /*
  ** Check whether the command succeeded or not
  */
  IF NOT Form_Success THEN
    Message('Error -- Message not sent.');
  ELSE
    Message('Message Sent.');
  END IF;
END;
```

ID_NULLビルトイン

説明

オブジェクトIDが変更可能かどうかを示すBOOLEAN値を戻します。

構文

```
FUNCTION ID_NULL
  (Alert  BOOLEAN);
FUNCTION ID_NULL
  (Block  BOOLEAN);
FUNCTION ID_NULL
  (Canvas  BOOLEAN);
FUNCTION ID_NULL
  (Editor  BOOLEAN);
FUNCTION ID_NULL
  (FormModule  BOOLEAN);
FUNCTION ID_NULL
  (GroupColumn  BOOLEAN);
FUNCTION ID_NULL
  (Item  BOOLEAN);
FUNCTION ID_NULL
  (LOV  BOOLEAN);
FUNCTION ID_NULL
  (MenuItem  BOOLEAN);
FUNCTION ID_NULL
  (ParamList  BOOLEAN);
FUNCTION ID_NULL
  (RecordGroup  BOOLEAN);
FUNCTION ID_NULL
  (Relation  BOOLEAN);
```

```
FUNCTION ID_NULL
  (Timer  BOOLEAN);
FUNCTION ID_NULL
  (Viewport  BOOLEAN);
FUNCTION ID_NULL
  (Window  BOOLEAN);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

BOOLEAN

問合せ入力モード

可

パラメータ

object_id	このファンクションをコールして、次のオブジェクトIDタイプの結果をテストできます。 ■ ALERT ■ BLOCK ■ CANVAS ■ EDITOR ■ FORMMODULE ■ GROUPCOLUMN ■ ITEM ■ LOV ■ MENUITEM ■ PARAMLIST ■ RECORDGROUP ■ RELATION ■ TIMER ■ VIEWPORT ■ WINDOW

使用上の注意

実行時に動的に作成されたオブジェクトが存在するかどうかをチェックするにはID_NULLを使用します。たとえば、特定のレコード・グループが既に存在する場合、そのレコード・グループの作成を試みると、エラー・メッセージが出されます。これをチェックするには、次の一般プロセスに従ってください。

- オブジェクトIDを取得するには適切なFIND_のビルトイン項目を使用します。
- そのIDを持つオブジェクトが既に存在するかどうかをチェックするには、ID_NULLを使用します。
- そのオブジェクトが存在していなければ、それを作成します。

オブジェクトが存在するかどうかをさまざまな時点で（実行中に複数回）テストする場合、ID_NULLを使用する前に必ず適切なFIND_を再発行する必要があります。

ID_NULLの例

CREATE_GROUPビルトインを参照。

IMAGE_SCROLLビルトイン

説明

指定されたオフセット（x座標とy座標）のできるだけ近い場所にイメージ項目をスクロールします。イメージがイメージ項目より大きい場合に使用すると便利です。

構文

```
PROCEDURE IMAGE_SCROLL
  (item_name VARCHAR2,
   X NUMBER,
   Y NUMBER
);
PROCEDURE IMAGE_SCROLL
  (item_id ITEM,
   X NUMBER,
   Y NUMBER
);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_id	Form Builderによりイメージ項目作成時に割り当てられる一意のIDを指定します。
item_name	ユーザーがイメージ定義時にそのイメージに与えた名前を指定します。
X	オフセットのX座標。
y	オフセットのY座標。

IMAGE_SCROLLの例

たとえば、イメージのサイズがイメージ項目のサイズの2倍あると仮定します。つまり、イメージの座標が0,200で、項目の座標が0,100であるとします。イメージを大まかに中央に配置するには、IMAGE_SCROLL XおよびYの座標を50と50に設定します。これによって項目の左上の座標は0,0の代わりに50,50に設定され、イメージが座標50から150に表示されるようにオフセットされます。

IMAGE_ZOOMビルトイン

説明

zoom_typeで指定した効果およびzoom_factorで指定した量により、そのイメージをズームインまたはズームアウトします。

構文

```
PROCEDURE IMAGE_ZOOM
  (image_id   ITEM,
   zoom_type  NUMBER);
PROCEDURE IMAGE_ZOOM
  (image_name  VARCHAR2,
   zoom_type   NUMBER);
PROCEDURE IMAGE_ZOOM
  (image_id     ITEM,
   zoom_type    NUMBER,
   zoom_factor  NUMBER);
PROCEDURE IMAGE_ZOOM
  (image_name   VARCHAR2,
   zoom_type    NUMBER,
   zoom_factor  NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

image_id	Form Builderによりイメージ項目作成時に割り当てられる一意のIDを指定します。このIDのデータ型はITEMです。
image_name	ユーザーがイメージ定義時にそのイメージに与えた名前を指定します。
zoom_type	表示されるイメージに与える効果を記述するには、次のいずれかの定数を指定します。 ADJUST_TO_FIT　表示四角形の中にそのイメージが納まるよう倍率を調整します。イメージ全体を参照でき、またそのイメージには最大限多くのイメージ項目がゆがみなく納められます。 SELECTION_RECTANGLE　選択領域が完全にイメージ項目を満たすように、イメージの倍率を調整します。 ZOOM_IN_FACTOR　zoom_factorに従って、イメージを拡大します。 ZOOM_OUT_FACTOR　zoom_factorに従ってイメージを縮小します。 ZOOM_PERCENT　*zoom_factor*に指示されたパーセンテージに従ってイメージ倍率を変更します。
zoom_factor	イメージ・ズームの要因またはパーセンテージを指定します。この引数はINTEGERで指定します。

使用上の注意

- zoom_factorが妥当であるかチェックします。たとえば、ZOOM_IN_FACTORに 100を指定すると、イメージ・サイズは100倍になり、アプリケーションにメモリー不足が起こる場合があります。
- ZOOM_IN_FACTORまたはZOOM_OUT_FACTORを指定する場合、zoom_factorには、正の整数値であれば何でも使用できますが、2、4または8のときに最適なパフォーマンスが得られます。
- ZOOM_PERCENTを指定する場合、zoom_factorには、正の整数値であれば何でも使用できます。イメージを拡大するには、100より大きいパーセンテージを指定します。
- SELECTION_RECTANGLEを指定する前に、オペレータは、マウスを使用して領域を選択してください。これを行わないと、Form Builderによってエラー・メッセージが戻されます。
- 設計上の注意として、SELECTION_RECTANGLEを使用するイメージ上にはスクロール・バーが必要です。
- カラー・イメージおよびモノクロ・イメージの両方に有効です。

IMAGE_ZOOMの例

次の例では、When-Button-Pressedトリガーを示しています。このボタンが押されるたびに、イメージ・サイズが2倍になります。

```
Image_Zoom('my_image', zoom_in_factor, 2);
```

INIT_OLEARGSビルトイン

説明

OLEオブジェクトのメソッドに渡される引数の数を定義します。

構文

```
PROCEDURE INIT_OLEARGS (num_args NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

num_args	メソッドに渡される引数の数プラス1。

使用上の注意

- このビルトインは、ADD_OLEARGを使用して引数の型と値を定義する前にコールする必要があります。
- このビルトインとADD_OLEARGは、プロパティがアクセスされる場合（たとえば索引によって）、GET_OLE_*コール用に使用されます。
- GET_OLE_*コールがOLEパラメータを取らない場合、このコールの前にINIT_OLEARGSを使用する必要はありません。
- num_argsで指定する数字は、実際の引数の数に1を足した数であることに注意してください。（4つの引数を渡す場合はnum_argに5を設定します）。1を足す操作はGET_OLE_*コールについてのみ必要であり、CALL_OLEの場合は必要ありません。ただし、CALL_OLEについてこの操作を行ってもなんら支障はありません。

INITIALIZE_CONTAINERビルトイン

説明

OLEオブジェクトをサーバー互換ファイルからOLEコンテナに挿入します。

構文

```
PROCEDURE INITIALIZE_CONTAINER
  (item_id    Item,
   file_name  VARCHAR2);
PROCEDURE INITIALIZE_CONTAINER
  (item_name  VARCHAR2,
   file_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
file_name	OLEコンテナに挿入するためのオブジェクトが入ったファイル名を指定します。ファイル位置のパスを挿入します。

INITIALIZE_CONTAINERの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効です。

INITIALIZE_CONTAINERの例

```
/*
** Built-in:INITIALIZE_CONTAINER
** Example:Initializes an OLE container by inserting an object
**         from a specified file into an OLE container.
** Trigger:When-Button-Pressed
*/
DECLARE
 item_id  ITEM;
 item_name VARCHAR(25) := 'OLEITM';
```

```
BEGIN
 item_id := Find_Item(item_name);
 IF Id_Null(item_id) THEN
  message('No such item:'||item_name);
 ELSE
  Initialize_Container(item_id,'c:¥OLE¥oleobj.xls');
 END IF;
END;
```

INSERT_RECORDビルトイン

説明

On-Insertトリガーからコールされると、転記トランザクションおよびコミット・トランザクション処理中に、現レコードをそのデータベースに挿入します。このビルトインは主に、Oracle以外のデータ・ソースに対して実行されるアプリケーション用に組み込まれているものです。

構文

```
PROCEDURE INSERT_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

INSERT_RECORDの制限事項

On-Insertトリガーの場合のみ有効。

INSERT_RECORDの例

```
/*
** Built-in:INSERT_RECORD
** Example :Perform Form Builder standard insert processing
**          based on a global flag setup at startup by the
**          form, perhaps based on a parameter.
** Trigger:On-Insert
```

```
*/
BEGIN
  /*
  ** Check the global flag we setup at form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    User_Exit('my_insrec block=EMP');
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Insert_Record;
  END IF;
END;
```

ISSUE_ROLLBACKビルトイン

説明

On-Rollbackトリガーからコールされると、デフォルトのForm Builder処理を起動し、指示されたセーブポイントにロールバックします。

このビルトインは主に、Oracle以外のデータ・ソースに対して実行されるアプリケーション用に組み込まれているものです。

構文

```
PROCEDURE ISSUE_ROLLBACK
(savepoint_name VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

savepoint_name	ロールバックするセーブポイント名。savepoint_nameがNULLの場合、完全なロールバックとなります。

ISSUE_ROLLBACKの制限事項

ISSUE_ROLLBACKがOn-Rollbackトリガーの外部で使用された場合、または GET_APPLICATION_PROPERTY(SAVEPOINT_NAME)へのコールにより提供されるセーブポイント以外のセーブポイントと共に使用された場合は、予期しない結果になることがあります。

ISSUE_ROLLBACKの例

```
/*
** Built-in:ISSUE_ROLLBACK
** Example:Perform Form Builder standard Rollback processing.
**          Decide whether to use this built-in based on a
**          global flag setup at startup by the form.
**          perhaps based on a parameter.
** Trigger:On-Rollback
*/
DECLARE
  sp_name VARCHAR2(80);
BEGIN
  /*
  ** Get the name of the savepoint to which Form Builder needs to
  ** rollback.(NULL = Full Rollback)
  */
  sp_name := Get_Application_Property(SAVEPOINT_NAME);
  /*
  ** Check the global flag we setup at form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    User_Exit('my_rollbk name='||sp_name);
  ELSE
    Issue_Rollback(sp_name);
 END IF;
END;
```

ISSUE_SAVEPOINTビルトイン

説明

On-Savepointトリガーからコールされると、ISSUE_SAVEPOINTではセーブポイント発行のためのデフォルト処理が起動されます。On-Savepointトリガーが存在しない場合には、GET_APPLICATION_PROPERTY (SAVEPOINT_NAME)を使用して、Form Builderによりデフォルトで発行されるセーブポイント名を判別できます。

このビルトインは主に、Oracle以外のデータ・ソースに対して実行されるアプリケーション用に組み込まれているものです。

構文

```
PROCEDURE ISSUE_SAVEPOINT
(savepoint_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

savepoint _name	発行するセーブポイント名。

ISSUE_SAVEPOINTの制限事項

FM_<number>という名前のセーブポイントは、セーブポイント名がGET_APPLICATION_PROPERTYへのコールによって与えられない限り発行しないでください。これを発行すると、Form Builderによって発行されるセーブポイントと衝突を起こす場合があります。

ISSUE_SAVEPOINTの例

```
/*
** Built-in:ISSUE_SAVEPOINT
** Example:Perform Form Builder standard savepoint processing.
**          Decide whether to use this built-in based on a
**          global flag setup at startup by the form,
**          perhaps based on a parameter.
** Trigger:On-Savepoint
*/
DECLARE
 sp_name VARCHAR2(80);
BEGIN
  /*  Get the name of the savepoint Form Builder needs to issue
  */
  sp_name := Get_Application_Property(SAVEPOINT_NAME);
  /*  Check the global flag we setup at form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    User_Exit('my_savept name='||sp_name);
```

```
  /*  Otherwise, do the right thing.
  */
  ELSE
    Issue_Savepoint(sp_name);
  END IF;
END;
```

ITEM_ENABLEDビルトイン

説明

そのメニュー項目が変更可能であればブール値TRUEを戻します。そのメニュー項目が使用禁止の場合には、ブール値FALSEを戻します。

注意: ITEM_ENABLEDは、GET_MENU_ITEM_PROPERTY (MENU_ITEM, ENABLED)と等価です。

構文

```
FUNCTION ITEM_ENABLED
  (mnunam  VARCHAR2,
   itmnam  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

BOOLEAN

問合せ入力モード

可

パラメータ

mnunam	**そのメニューのVARCHAR2名を指定します。**
itmnam	**そのメニュー項目のVARCHAR2名を指定します。**

LAST_OLE_ERRORビルトイン

説明

最も新しいOLEエラー条件の識別番号を返します。

構文

```
FUNCTION LAST_OLE_ERROR RETURN number;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

パラメータ

なし

使用上の注意

- この関数は、大部分のエラー条件について使用できます。ただし、エラーがPL/SQL例外の場合はLAST_OLE_EXCEPTION関数を代わりに使用します。
- エラーの値と意味の詳細は、winerror.hを参照してください。winerror.hは、ご使用のCコンパイラのベンダーから提供されています。

LAST_OLE_EXCEPTIONビルトイン

説明

コールされたオブジェクト内で発生した最も新しいOLE例外の識別番号を返します。

構文

```
FUNCTION LAST_OLE_EXCEPTION
  (source OUT VARCHAR2, description OUT VARCHAR2,
   helpfile OUT VARCHAR2,
   helpcontextid OUT PLS_INTEGER)
RETURN errornumber PLS_INTEGER;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

OLEサーバーによってこの例外条件に割り当てられたエラー番号

パラメータ

source	例外条件を呼び出したOLEサーバーの名前。
description	エラー・メッセージのテキスト。
helpfile	OLEサーバーが追加のエラー情報を格納するファイルの名前。
helpcontextid	前述ヘルプ・ファイル内の特定ドキュメントのID。

使用上の注意

この関数は、OLEオブジェクト・サーバーをコールした結果、PL/SQL FORM_OLE_FAILURE例外が発生した後で使用できます。他のタイプの例外（PL/SQL例外以外）の詳細は、LAST_OLE_ERROR関数を使用します。

LAST_RECORDビルトイン

説明

ブロックのレコード・リストの最後のレコードへナビゲートします。ブロックで問合せがオープンしていれば、Form Builderは、ブロックのレコード・リストで選択されている残りのレコードをフェッチし、問合せをクローズします。

構文

```
PROCEDURE LAST_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

LAST_RECORDの例

FIRST_RECORDビルトインを参照

LIST_VALUESビルトイン

説明

LIST_VALUESは、付加された値リストがあるテキスト項目内に入力フォーカスがある限り、現項目の値リストを表示します。その値リストは、オペレータがその値リストを消すか、値を選択するまで表示されたままとなります。

デフォルトでは、LIST_VALUESはNO_RESTRICTパラメータを使用します。このパラメータを使用すると、Form Builderでは自動検索と完了機能が使用されないようになります。RESTRICTパラメータを使用すると、Form Builderでは、自動検索と完了機能が使用されます。

自動検索および完成機能

自動検索および完成機能により、値リストはテキスト項目の現在の値を検索値として評価します。つまり、オペレータが値リストのあるテキスト項目内の「List」を押すと、Form Builderではその項目に値が含まれているかどうかチェックされます。

テキスト項目に値が入っていると、Form Builderは、ちょうどオペレータがリストを絞り込むためにその値を値リストの検索フィールドに入れて「List」を押したかのように、自動的にその値を使用します。

そのリスト項目数が1つになるまでリストの内容を縮小した場合には、Form Builderではその値リストが表示されず、自動的に正しい値がフィールドに読み込まれます。

構文

```
PROCEDURE LIST_VALUES
(kwd  NUMBER);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

kwd	次のいずれかの定数を指定します。 NO_RESTRICT　Form Builderにより自動検索と完了機能が使用されないよう指定します。 RESTRICT　Form Builderにより自動検索と完了機能が使用されるよう指定します。

LOCK_RECORDビルトイン

説明

データベース内の現レコードに対応する行をロックしようとします。LOCK_RECORDでは、「ロック・モード」ブロック・プロパティが「即時」（デフォルト）または「遅延」のどちらに設定されているかにかかわらず、そのレコードが即時ロックされます。

LOCK_RECORDをOn-Lockトリガーの内部から実行すると、デフォルトのデータベース・ロックが起動されます。次の例でこのテクニックについて説明します。

構文

```
PROCEDURE LOCK_RECORD;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

なし

LOCK_RECORDの例

```
/*
** Built-in:LOCK_RECORD
** Example:Perform Form Builder standard record locking on the
**          queried record which has just been deleted or
**          updated.Decide whether to use default
**          processing or a user exit by consulting a
**          global flag setup at startup by the form,
**          perhaps based on a parameter.
** Trigger:On-Lock
```

```
*/
BEGIN
  /*
  ** Check the global flag we set up at form startup
  */
  IF :Global.Non_Oracle_Datasource = 'TRUE' THEN
    User_Exit('my_lockrec block=EMP');
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Lock_Record;
  END IF;
END;
```

LOGONビルトイン

説明

示されたユーザー名およびパスワードにより、デフォルトのForm Builderログイン処理が実行されます。デフォルトのログイン処理を補強するときには、On-Logonトリガーからこのプロシージャをコールします。

構文

```
PROCEDURE LOGON
  (username  VARCHAR2,
   password  VARCHAR2);
PROCEDURE LOGON
  (username  VARCHAR2,
   password  VARCHAR2,
   logon_screen_on_error  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

このビルトインは次の引数を取ります。

username	最大80文字までの有効なユーザー名。
password	データベース接続文字列を含む最大80文字までの有効なパスワード。
logon_screen_on_error	オプションのBOOLEANパラメータ。これがTRUE（デフォルト）に設定されていると、指定されたログインが（ユーザー名やパスワードの誤りなどの理由により）失敗した場合に、Form Builderに自動的にそのログイン・スクリーンを表示させます。*logon_screen_on_error*がFALSEに設定され、ログインが失敗すると、そのログイン・スクリーンは表示されず、FORM_FAILUREがTRUEに設定され、設計者は適切な方法でその条件を取り扱えるようになります。

使用上の注意:

LOGONを使用してOPS$データベースに接続する場合、パスワードのユーザー名とデータベース名にスラッシュ'/'を使用します。

LOGONの制限事項

- リモート・データベースを識別した場合、そのデータベースとのNet8接続が実行時に存在している必要があります。
- Form Builderでは1度に接続できるデータベースは1つのみです。しかし、データベース・リンクを使用すると、単一接続で複数のデータベースにアクセスできます。

LOGONの例

```
/*
** Built-in:LOGON
** Example:Perform Form Builder standard logon to the ORACLE
**          database.Decide whether to use Form Builder
**          built-in processing or a user exit by consulting a
**          global flag setup at startup by the form,
**          perhaps based on a parameter.This example
**          uses the 'Get_Connect_Info' procedure from the
**          GET_APPLICATION_PROPERTY example.
** Trigger:On-Logon
*/
DECLARE
  un  VARCHAR2(80);
  pw  VARCHAR2(80);
  cn  VARCHAR2(80);
BEGIN
  /*
  ** Get the connection info
  */
```

```
  Get_Connect_Info(un,pw,cn);
  /*
  ** If at startup we set the flag to tell our form that we
  ** are not running against ORACLE, then call our
  ** appropriate MY_LOGON userexit to logon.
  */
  IF :Global.Non_Oracle_Datasource = 'TRUE' THEN
    User_Exit('my_logon username='||un||' password='||pw);
  /*
  ** Otherwise, call the LOGON built-in
  */
  ELSE
  /*
** Use the following to place a slash in the username field for OPS$ logon
  */
IF un IS NULL THEN
un:='/';
END IF
IF cn IS NOT NULL THEN
     LOGON(un,pw||'@'||cn);
   ELSE
     LOGON(un,pw);
   END IF;
  END IF;
END;
```

LOGON_SCREENビルトイン

説明

デフォルトのForm Builderログイン・スクリーンを表示し、有効なユーザー名およびパスワードを要求します。ごく一般的には、On-Logonトリガーにこのビルトイン・サブプログラムを組み込んで、非Oracleデータ・ソースに接続することになります。

構文

```
PROCEDURE LOGON_SCREEN;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

LOGON_SCREENの制限事項

使用しているデータ・ソースとの接続を作成するには、LOGONのビルトインを発行する必要があります。

LOGON_SCREENの例

```
/*
** Built-in:LOGON_SCREEN
** Example:Use the default Form Builder logon screen to prompt
**          for username and password before logging on to
**          the database.This uses the 'Get_Connect_Info'
**          procedure from the GET_APPLICATION_PROPERTY
**          example.
*/
DECLARE
  un  VARCHAR2(80);
  pw  VARCHAR2(80);
  cn  VARCHAR2(80);
BEGIN
  /*
  ** Bring up the logon screen
  */
  Logon_Screen;
  /*
  ** Get the username, password and
  ** connect string.
  */
  Get_Connect_Info( un, pw, cn );
  /*
  ** Log the user onto the database
  */
  IF cn IS NOT NULL THEN
    LOGON(un,pw||'@'||cn);
  ELSE
    LOGON(un,pw);
```

```
  END IF;
END;
```

LOGOUTビルトイン

説明

ORACLE RDBMSとアプリケーションの接続を切断します。LOGOUTビルトインをコールすると、オープンされているカーソルはすべて自動的にクローズされます。LOGONによりプログラミングで再度ログインできます。複数の接続がある複数フォーム・アプリケーションからLOGOUTすると、Form Builderでは、ユーザーが次にLOGONを実行したときに、その接続がすべて再確立されます。

構文

```
PROCEDURE LOGOUT;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

LOGOUTの例

```
/*
** Built-in:LOGOUT
** Example:Perform Form Builder standard logout.Decide
**          whether to use Form Builder built-in processing or a
**          user exit by consulting a global flag setup at
**          startup by the form, perhaps based on a
**          parameter.
** Trigger:On-Logout
*/
BEGIN
  /*
  ** Check the flag we setup at form startup
  */
  IF :Global.Non_Oracle_Datasource = 'TRUE' THEN
```

```
    User_Exit('my_logout');
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Logout;
  END IF;
END;
```

MENU_CLEAR_FIELDビルトイン

説明

MENU_CLEAR_FIELDは、現フィールドの値をカーソル位置からフィールドの末尾まで消去します。現カーソル位置が最後の非ブランク文字の右側にある場合、MENU_CLEAR_FIELDでは、そのフィールド全体が消去され、その値がNULLになります。

構文

```
PROCEDURE MENU_CLEAR_FIELD;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

MENU_CLEAR_FIELDの制限事項

「パラメータ値入力」ダイアログ内でのみ有効です。

MENU_NEXT_FIELDビルトイン

説明

MENU_NEXT_FIELDを使用して、「パラメータ値入力」ダイアログ内の次のフィールドにナビゲートします。

構文

```
PROCEDURE MENU_NEXT_FIELD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

MENU_NEXT_FIELDの制限事項

「パラメータ値入力」ダイアログ内でのみ有効です。

MENU_PARAMETERビルトイン

説明

MENU_PARAMETERを使用して、「パラメータ値入力」ダイアログ・ボックス内に、現メニューに関係するすべてのパラメータおよびそれらの現行の設定値を表示します。

構文

```
PROCEDURE MENU_PARAMETER;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

MENU_PARAMETERの制限事項

全画面表示形式で実行されるメニューにのみ有効です。

MENU_PREVIOUS_FIELDビルトイン

説明

MENU_PREVIOUS_FIELDを使用して「パラメータ値入力」ダイアログ内の直前フィールドに戻ります。

構文

```
PROCEDURE MENU_PREVIOUS_FIELD;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

MENU_PREVIOUS_FIELDの制限事項

「パラメータ値入力」ダイアログ・ボックスでのみ有効です。

MENU_REDISPLAYビルトイン

説明

このプロシージャはメニュー内のスクリーンを再び描画します。

構文

```
PROCEDURE MENU_REDISPLAY;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

MENU_REDISPLAYの制限事項

キャラクタ・モードまたはブロック・モード・プラットフォーム上でのみ有効です。

MENU_SHOW_KEYSビルトイン

説明

MENU_SHOW_KEYSは、実行時にメニュー・モジュール用の「キー」画面を表示します。

構文

```
PROCEDURE MENU_SHOW_KEYS;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

MENU_SHOW_KEYSの制限事項

MENU_SHOW_KEYSはどのようなコンテキストでも使用可能です。

MESSAGEビルトイン

説明

指定されたテキストをメッセージ行に表示します。

構文

```
PROCEDURE MESSAGE
(message_string VARCHAR2,
   user_response   NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

message_string	文字列を指定し引用符で囲むか、VARCHAR2データ型の変数を指定します。
user_response	次のいずれかの定数を指定します。 ACKNOWLEDGE　2つの連続するメッセージが発行されるときは必ず、オペータが明示的に消す必要のあるモーダル警告をForm Builder表示するよう指定します。ACKNOWLEDGEでは、必ず最初のメッージの確認を強制的に要求し、それがなければ、2番目のメッージは表示できません。これがデフォルトです。 NO_ACKNOWLEDGE　2つの連続するメッセージが発行されるとき、オペレータが示された最初のメッセージに応答しなくても、Form Builder2番目のメッセージを表示するよう指定します。2番目のメッージは、最初のメッセージの確認をオペレータに求めることく即時に最初のメッセージに上書きされるため、NO_ACKNOWLEDGEを使用すると、オペレータが最初のメッセーを見ない危険性があります。

MESSAGEの制限事項

message_stringの最大長は200文字です。しかし、表示可能な最大文字数には、現在のフォントやランタイム・ウィンドウ・マネージャの制限などのいくつかの要因が影響を及ぼすことに注意してください。

MESSAGEの例

```
/*
** Built-in:MESSAGE
** Example:Display several messages to the command line
**          throughout the progress of a particular
**          subprogram.By using the NO_ACKNOWLEDGE parameter,
**          we can avoid the operator's having to
**          acknowledge each message explicitly.
*/
PROCEDURE Do_Large_Series_Of_Updates IS
BEGIN
  Message('Working...(0%)', NO_ACKNOWLEDGE);
  /*
  ** Long-running update statement goes here
  */
```

```
SYNCHRONIZE;
  Message('Working...(30%)', NO_ACKNOWLEDGE);
  /*
  ** Another long-running update statement goes here
  */
  Message('Working...(80%)', NO_ACKNOWLEDGE);
  /*
  ** Last long-running statement here
  */
  Message('Done...', NO_ACKNOWLEDGE);
END;
```

MESSAGE_CODEビルトイン

説明

Form Builderが現行ランフォーム・セッション中に生成した最新のメッセージのメッセージ番号を戻します。MESSAGE_CODEは、Form Builderがメッセージを生成する前やセッションの開始時点ではゼロを戻します。

MESSAGE_CODEを使用して、ユーザー・アクション（例: キーを押す）の結果をテストし、On-Messageトリガー内部の処理を判別します。

構文

```
FUNCTION MESSAGE_CODE;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

問合せ入力モード

可

パラメータ

なし

MESSAGE_CODEの例

```
/*
```

```
** Built-in:MESSAGE_CODE,MESSAGE_TEXT,MESSAGE_TYPE
** Example:Reword certain FRM message messages by checking
**          the Message_Code in an ON-MESSAGE trigger
** Trigger:On-Message
*/
DECLARE
  msgnum NUMBER      := MESSAGE_CODE;
  msgtxt VARCHAR2(80) := MESSAGE_TEXT;
  msgtyp VARCHAR2(3)  := MESSAGE_TYPE;
BEGIN
  IF msgnum = 40400 THEN
    Message('Your changes have been made permanent.');
  ELSIF msgnum = 40401 THEN
    Message('You have no unsaved changes outstanding.');
  ELSE
    /*
    ** Print the Normal Message that would have appeared
    **
    **  FRM-12345:Message Text Goes Here
    */
    Message(msgtyp||'-'||TO_CHAR(msgnum)||':'||msgtxt);
  END IF;
END;
```

MESSAGE_TEXTビルトイン

説明

Form Builderが現行のランフォーム・セッション中に生成した最新のメッセージのメッセージ・テキストを戻します。MESSAGE_TEXTは、Form Builderがメッセージを生成する前やセッションの開始時点ではNULLを戻します。

MESSAGE_TEXTを使用してユーザー・アクション（例: キーを押す）の結果をテストし、On-Messageトリガー内部の処理を判別します。

注意: ユーザーのアプリケーションが複数の言語によってサポートされる必要がある場合、MESSAGE_TEXTビルトインの代わりにMESSAGE_CODEビルトインを使用します。メッセージ・テキストではなくメッセージ・コードを参照することは、各国語に対応しているアプリケーションにおいて有効です。

構文

```
FUNCTION MESSAGE_TEXT;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

なし

MESSAGE_TEXTの例

```
/*
** Built-in:MESSAGE_CODE,MESSAGE_TEXT,MESSAGE_TYPE
** Example:Reword certain FRM message messages by checking
**          the Message_Code in an ON-MESSAGE trigger
** Trigger:On-Message
*/
DECLARE
  msgnum NUMBER      := MESSAGE_CODE;
  msgtxt VARCHAR2(80) := MESSAGE_TEXT;
  msgtyp VARCHAR2(3)  := MESSAGE_TYPE;
BEGIN
  IF msgnum = 40400 THEN
    Message('Your changes have been made permanent.');
  ELSIF msgnum = 40401 THEN
    Message('You have no unsaved changes outstanding.');
  ELSE
    /*
    ** Print the Normal Message that would have appeared
    **
    **  FRM-12345:Message Text Goes Here
    */
    Message(msgtyp||'-'||TO_CHAR(msgnum)||':'||msgtxt);
  END IF;
END;
```

MESSAGE_TYPEビルトイン

説明

Form Builderが現行のランフォーム・セッション中に生成した最新のメッセージのメッセージ・タイプを戻します。

MESSAGE_TYPEを使用してユーザー・アクション（例: キーを押す）の結果をテストし、On-Messageトリガー内部の処理を判別します。

構文

```
FUNCTION MESSAGE_TYPE;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

MESSAGE_TYPEでは、そのメッセージ・タイプについて3つのうちのいずれかの値を戻します。

- FRM　Form Builderメッセージが生成されたことを示す。
- ORA　Oracleメッセージが生成されたことを示す。
- NULL　Form Builderによりセッション中にメッセージが発行されていないことを示す。

問合せ入力モード

可

パラメータ

なし

MESSAGE_TYPEの例

```
/*
** Built-in:MESSAGE_CODE,MESSAGE_TEXT,MESSAGE_TYPE
** Example:Reword certain FRM message messages by checking
**          the Message_Code in an ON-MESSAGE trigger
** Trigger:   On-Message
*/
DECLARE
```

```
  msgnum NUMBER      := MESSAGE_CODE;
  msgtxt VARCHAR2(80) := MESSAGE_TEXT;
  msgtyp VARCHAR2(3)  := MESSAGE_TYPE;
BEGIN
  IF msgnum = 40400 THEN
    Message('Your changes have been made permanent.');
  ELSIF msgnum = 40401 THEN
    Message('You have no unsaved changes outstanding.');
  ELSE
    /*
    ** Print the Normal Message that would have appeared
    **
    **  FRM-12345:Message Text Goes Here
    */
    Message(msgtyp||'-'||TO_CHAR(msgnum)||':'||msgtxt);
  END IF;
END;
```

MOVE_WINDOWビルトイン

説明

座標で指定された位置にウィンドウを移動します。

フォーム・プロパティ「座標システム」を「文字」と指定している場合の*x, y* 座標は文字で指定されます。「座標システム」を「実際単位」と指定している場合の*x, y* 座標は、ピクセルまたはインチ、センチメートル、ポイントのうち、ユーザーが選択した実際単位で指定されます。

構文

```
FUNCTION MOVE_WINDOW
  (window_id  Window,
   x         NUMBER,
   y         NUMBER);
FUNCTION MOVE_WINDOW
  (window_name  VARCHAR2,
   x           NUMBER,
   y           NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

パラメータ

window_id	Form Builderがウィンドウ作成時にそのウィンドウに割り当てられる一意のIDを指定します。FIND_WINDOWのビルトインを使用して、そのIDを適切な型の変数に戻します。このIDのデータ型はWindow型です。
window_name	ウィンドウの作成時に付けた名前を指定します。
x	ウィンドウの左上コーナーを配置する画面上のx座標を指定します。
y	ウィンドウの左上コーナーを配置する画面上のy座標を指定します。

MOVE_WINDOWの例

```
/*
** Built-in:MOVE_WINDOW
** Example:Move window2 to be anchored at the bottom right
**          corner of window1.
*/
PROCEDURE Anchor_Bottom_Right2( Window2 VARCHAR2, Window1 VARCHAR2) IS
  wn_id1 Window;
  wn_id2 Window;
  x     NUMBER;
  y     NUMBER;
  w     NUMBER;
  h     NUMBER;
BEGIN
  /*
  ** Find Window1 and get its (x,y) position, width, and
  ** height.
  */
  wn_id1 := Find_Window(Window1);
  x     := Get_Window_Property(wn_id1,X_POS);
  y     := Get_Window_Property(wn_id1,Y_POS);
  w     := Get_Window_Property(wn_id1,WIDTH);
  h     := Get_Window_Property(wn_id1,HEIGHT);
  /*
  ** Anchor Window2 at (x+w,y+h)
  */
  wn_id2 := Find_Window(Window2);
  Move_Window( wn_id2, x+w, y+h );
END;
```

NAME_INビルトイン

説明

指示された変数の値を戻します。

戻り値は文字列の形になっています。ただし、NAME_INでは、数値や日付を文字列に戻してから、その文字列を適切なデータ型に変換できます。この戻り値は、実行可能文内で使用する他の値と同様に使用できます。

NAME_IN関数をネストすると、Form Builderではもっとも内側のNAME_IN関数からもっとも外側のNAME_IN関数までが個々に評価されます。

構文

```
FUNCTION NAME_IN
(variable_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

variable_name	**有効な変数またはテキスト項目を指定します。その名前のデータ型はVARCHAR2です。**

使用上の注意

戻り値がDATE型文字列の場合、NAME_INはBUILTIN_DATE_FORMATプロパティで指定されたフォーマット・マスクを使用します。DATE_FORMAT_COMPATIBILITY_MODEプロパティが4.5に設定されると、デフォルトのフォーマット・マスクを使用して戻された文字列が設定されます。

NAME_INの例

```
/*
** Built-in:NAME_IN
** Example:Simple implementation of a Last-In-First-Out
```

```
**          stack mechanism using Global variables.
**          For each named stack, a global variable
**          GLOBAL.<stackname>_PTR points to the largest
**          element on the stack.PUSH increments this
**          value as new elements are added.Values
**          PUSH'ed on or POP'ed off the named stack are
**          actually stored in GLOBAL variables of a
**          conveniently formed name:GLOBAL.<stackname>nnn
**          where 'nnn' is the number of the element on the
**          stack.
**
**          Usage:
**          Push('MYSTACKNAME', '1');
**          Push('MYSTACKNAME', '2');
**
**          str_var := Pop('MYSTACKNAME');  -- Gets '2'
**          str_var := Pop('MYSTACKNAME');  -- Gets '1'
**          str_var := Pop('MYSTACKNAME');  -- Gets 'EOS'
**
*/
PROCEDURE Push ( the_stackname VARCHAR2,
        the_value    VARCHAR2 ) IS

  ptr_name VARCHAR2(40); -- This stack's pointer name
  prefix   VARCHAR2(40); -- Common prefix for storage vars
  elt_name VARCHAR2(40); -- Name of storage element
  new_idx  VARCHAR2(4) ; -- New stack pointer value
BEGIN
  /*
  ** For any named stack that we reference, the global
  ** variables used for storing the stack's values and the
  ** stack's pointer all begin with a common prefix:
  ** GLOBAL.<stackname>
  */
  prefix := 'GLOBAL.'|| the_stackname;
  /*
  ** This named stack's pointer resides in
  ** GLOBAL.<stackname>_PTR Remember that this is the *name*
  ** of the pointer.
  */
  ptr_name := prefix || '_PTR';
  /*
  ** Initialize the stack pointer with a default value of
```

```
  ** zero if the stack pointer did not exist previously, ie
  ** the GLOBAL.<stackname>_PTR had yet to be created.
  */
  Default_Value( '0', ptr_name );
  /*
  ** Since we're PUSH'ing a new element on the stack,
  ** increment the stack pointer to reflect this new
  ** element's position.Remember that GLOBAL variables are
  ** always of type VARCHAR2, so we must convert them TO_NUMBER
  ** before any calculations.
  */
  new_idx  := TO_CHAR( TO_NUMBER( Name_In( ptr_name ) ) + 1 ) ;
  Copy( new_idx   , ptr_name );
  /*
  ** Determine the name of the global variable which will
  ** store the value passed in, GLOBAL.<stackname><new_idx>.
  ** This is simply the prefix concatenated to the new index
  ** number we just calculated above.
  */
  elt_name := prefix||new_idx;
  Copy( the_value , elt_name );
END;

FUNCTION Pop ( the_stackname VARCHAR2)
RETURN VARCHAR2 IS
  ptr_name VARCHAR2(40); -- This stack's pointer name
  prefix   VARCHAR2(40); -- Common prefix for storage vars
  elt_name VARCHAR2(40); -- Name of storage element
  new_idx  VARCHAR2(4) ; -- New stack pointer value
  cur_idx  VARCHAR2(4) ; -- Current stack pointer value
  the_val  VARCHAR2(255);

  EMPTY_STACK   CONSTANT VARCHAR2(3) := 'EOS';
  NO_SUCH_STACK CONSTANT VARCHAR2(3) := 'NSS';
BEGIN
  /*
  ** For any named stack that we reference, the global
  ** variables used for storing the stack's values and the
  ** stack's pointer all begin with a common prefix:
  ** GLOBAL.<stackname>
  */
  prefix   := 'GLOBAL.'|| the_stackname;
  /*
```

```
 ** This named stack's pointer resides in
 ** GLOBAL.<stackname>_PTR Remember that this is the *name*
 ** of the pointer.
 */
 ptr_name := prefix || '_PTR';
 /*
 ** Force a default value of NULL so we can test if the
 ** pointer exists (as a global variable).If it does not
 ** exist, we can test in a moment for the NULL, and avoid
 ** the typical error due to referencing non-existent
 ** global variables.
 */
 Default_Value( NULL, ptr_name );
 /*
 ** If the *value* contained in the pointer is NULL, then
 ** the pointer must not have existed prior to the
 ** Default_Value statement above.Return the constant
 ** NO_SUCH_STACK in this case and erase the global
 ** variable that the Default_Value implicitly created.
 */
IF Name_In( ptr_name ) IS NULL THEN
  the_val := NO_SUCH_STACK;
 Erase( ptr_name );
/*
** Otherwise, the named stack already exists.Get the
** index of the largest stack element from this stack's
** pointer.
*/
ELSE
  cur_idx := Name_In( ptr_name ) ;
  /*
  ** If the index is zero, then the named stack is already
  ** empty, so return the constant EMPTY_STACK, and leave
  ** the stack's pointer around for later use, ie don't
  ** ERASE it.
  **
  ** Note that a stack can only be empty if some values
  ** have been PUSH'ed and then all values subsequently
  ** POP'ed.  If no values were ever PUSH'ed on this named
  ** stack, then no associated stack pointer would have
  ** been created, and we would flag that error with the
  ** NO_SUCH_STACK case above.
  */
```

```
   IF cur_idx = '0' THEN
    the_val := EMPTY_STACK;
   /*
   ** If the index is non-zero, then:
   ** (1) Determine the name of the global variable in
   **    which the value to be POP'ed is stored,
   **    GLOBAL.<stackname><cur_idx>
   ** (2) Get the value of the (cur_idx)-th element to
   **    return
   ** (3) Decrement the stack pointer
   ** (4) Erase the global variable which was used for
   **    value storage
   */
   ELSE
     elt_name:= prefix || cur_idx;
     the_val := Name_In( elt_name );
     new_idx := TO_CHAR( TO_NUMBER( Name_In(ptr_name) ) - 1 ) ;
     Copy( new_idx , ptr_name );
     Erase( elt_name );
   END IF;
  END IF;
  RETURN the_val;
END;
```

NEW_FORMビルトイン

説明

現フォームを終了し、指定されたフォームに入ります。コール側フォームは親フォームとして終了します。コール側フォームがそれより上位のフォームによってコールされていた場合、Form Builderではその上位のコールをアクティブに保ったまま、それを新規フォームのコールとして扱います。Form Builderにより、その終了したフォームが使用していたメモリー（データベース・カーソルなど）が解放されます。

Form Builderでは、これと同じ「ランフォーム」オプションのある新規フォームが親フォームとして実行されます。その親フォームがコール先フォームであった場合、Form Builderではそれと同じオプションのある新規フォームが親フォームとして実行されます。

構文

```
PROCEDURE NEW_FORM
  (formmodule_name  VARCHAR2);
```

```
PROCEDURE NEW_FORM
  (formmodule_name  VARCHAR2,
   rollback_mode    NUMBER);
PROCEDURE NEW_FORM
  (formmodule_name  VARCHAR2,
   rollback_mode    NUMBER,
   query_mode       NUMBER);
PROCEDURE NEW_FORM
  (formmodule_name  VARCHAR2,
   rollback_mode    NUMBER,
   query_mode       NUMBER,
   data_mode        NUMBER);
PROCEDURE NEW_FORM
  (formmodule_name  VARCHAR2,
   rollback_mode    NUMBER,
   query_mode       NUMBER,
   paramlist_id     PARAMLIST);
PROCEDURE NEW_FORM
  (formmodule_name  VARCHAR2,
   rollback_mode    NUMBER,
   query_mode       NUMBER,
   paramlist_name   VARCHAR2);
PROCEDURE NEW_FORM
  (formmodule_name  VARCHAR2,
   rollback_mode    NUMBER,
   query_mode       NUMBER,
   data_mode        NUMBER,
   paramlist_id     PARAMLIST);
PROCEDURE NEW_FORM
  (formmodule_name  VARCHAR2,
   rollback_mode    NUMBER,
   query_mode       NUMBER,
   data_mode        NUMBER,
   paramlist_name   VARCHAR2);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

formmodule_name	コール先フォームの名前（引用符（'）で囲む）。データ型はVARCHAR2です。
rollback_mode	TO_SAVEPOINT（デフォルト） Form Builderでは、コミットされていないすべての変更（ポストされた変更を含む）が現行フォームのセーブポイントにロール バックされます。 NO_ROLLBACK　Form Builderでは、セーブポイントにロールバックせずに、現フォームが終了します。ユーザーは、ロールバックを実行せずに最上位のフォームを離れられます。これは、ユーザーが、NEW_FORM操作全体にロックを保持することを意味します。このロックは、外部3GLプログラムからForm Builderを起動する場合にも発生します。このロックは、ユーザーがForm Builderから制御を取り戻したときにも有効です。 FULL_ROLLBACK　Form Builderでは、現ランフォーム・セッション中に加えられたコミットされていないすべての変更内容（ポスト済みのものを含む）がロールバックされます。ポスト専用モードで実行されているフォームからFULL_ROLLBACKを指定することはできません。（フォームにポストしていないレコードがあるときに別のフォームをコールすると、ポスト専用モードになる可能性があります。コール側フォームが発行したロックの喪失を避けるため、Form Builderではコール先フォーム内のコミット処理が封じられます。
query_mode	NO_QUERY_ONLY（デフォルト） 指示されたフォームを通常モードで実行します。エンド・ユーザーは、そのフォーム内で挿入、更新および削除ができます。 QUERY_ONLY　指示されたフォームを問合せ専用モードで実行します。エンド・ユーザーはレコードの問合せはできますが、挿入、更新、削除はできません。
data_mode	NO_SHARE_LIBRARY_DATA（デフォルト） Form Builderは（設計時に）同一のライブラリが付加されているフォームの間でデータを実行時に共有します。 SHARE_LIBRARY_DATA 実行時に、Form Builderは（設計時に）同一のライブラリが付加されているフォームの間でデータを共有します。
paramlist_id	Form Builderがパラメータ・リストを作成するときに割り当てる一意のID。コール側フォームから新規フォームにパラメータを渡すときのパラメータ・リストを指定します。データ型はPARAMLISTです。NEW_FORMを介してフォームに渡されたパラメータ・リストには、DATA_PARAMETER型（レコード・グループへのポインタ）を含めることはできません。
paramlist_name	ユーザーがパラメータ・リスト・オブジェクトを定義するときに付けた名前。データ型はVARCHAR2です。NEW_FORMを介してフォームに渡されたパラメータ・リストには、DATA_PARAMETER型（レコード・グループへのポインタ）を含めることはできません。

NEW_FORMの例

```
/* Create a generic procedure that will invoke the
```

```
** formname passed-in using the method indicated by
** the 'newform' and 'queryonly' parameters.
*/
PROCEDURE GENERIC_CALL(formname   VARCHAR2,
                       newform    VARCHAR2,
                       queryonly  VARCHAR2) IS

  msglvl         VARCHAR2(2);
  error_occurred  EXCEPTION;
BEGIN
  /*
  ** Remember the current message level and temporarily
  ** set it to 10 to suppress errors if an incorrect
  ** formname is called
  */
  msglvl := :SYSTEM.MESSAGE_LEVEL;
  :SYSTEM.MESSAGE_LEVEL := '10';

  IF newform = 'Y' THEN
     IF queryonly = 'Y' THEN
        NEW_FORM(formname, to_savepoint, query_only);
     ELSIF queryonly = 'N' THEN
        NEW_FORM(formname);
     END IF;
  ELSIF newform = 'N' THEN
     IF queryonly = 'Y' THEN
        CALL_FORM(formname, hide, no_replace, query_only);
     ELSIF queryonly = 'N' THEN
        CALL_FORM(formname);
     END IF;
  END IF;
  IF NOT form_success THEN
     MESSAGE('Cannot call form '||UPPER(formname)||
             '.Please contact your SysAdmin for help.');
     RAISE error_occurred;
  END IF;
  :SYSTEM.MESSAGE_LEVEL := msglvl;
EXCEPTION
  WHEN error_occurred THEN
     :SYSTEM.MESSAGE_LEVEL := msglvl;
    RAISE form_trigger_failure;
END;
```

NEXT_BLOCKビルトイン

説明

ナビゲーション順序内の次の入力可能なブロックにある最初のナビゲート可能な項目にナビゲートします。デフォルトでは、ナビゲーション順序内の次のブロックは、順序番号が次に大きなブロックであり、オブジェクト・ナビゲータ内のブロックの順序により定義されます。ただし、「次のナビゲーション・データ・ブロック」ブロック・プロパティは、異なるブロックを、ナビゲーションのための次のブロックとして指定するよう設定できます。

それより上位の順序の入力可能なブロックがない場合、NEXT_BLOCKは、順序番号が最も低い入力可能なブロックにナビゲートします。

構文

```
PROCEDURE NEXT_BLOCK;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

NEXT_BLOCKの例

```
/*
** Built-in:NEXT_BLOCK
** Example:If the current item is the last item in the
**          block, then skip to the next block instead of
**          the default of going back to the first item in
**          the same block
** Trigger:Key-Next-Item
*/
DECLARE
  cur_itm VARCHAR2(80) := :System.Cursor_Item;
  cur_blk VARCHAR2(80) := :System.Cursor_Block;
  lst_itm VARCHAR2(80);
BEGIN
  lst_itm := cur_blk||'.'||Get_Block_Property(cur_blk,LAST_ITEM);
  IF cur_itm = lst_itm THEN
```

```
    Next_Block;
  ELSE
    Next_Item;
  END IF;
END;
```

NEXT_FORMビルトイン

説明

複数フォーム・アプリケーションでは、順序番号が次に高い独立フォームにナビゲートします。（フォームは実行時に起動された順序で順序付けされます。） 順序番号がそれより高いフォームがない場合には、NEXT_FORMを使用して、順序番号がもっとも低いフォームにナビゲートします。そのようなフォームがない場合には、現フォームは現フォームのままとなります。

NEXT_FORMとともにナビゲートするときには、妥当性チェックは行われず、ナビゲーションを開始するフォームで起動されるWHEN-WINDOW-DEACTIVATEDトリガー、およびターゲット・フォームで起動されるWHEN-WINDOW-ACTIVATEDトリガー以外には、トリガーが起動しません。

構文

```
PROCEDURE NEXT_FORM;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

NEXT_FORMの制限事項

CALL_FORMがある別のフォームを起動した結果として現在使用禁止となっているフォームはターゲット・フォームにはできません。

NEXT_ITEMビルトイン

説明

順序番号が現項目の次に高いナビゲート可能な項目にナビゲートします。そのような項目がない場合、NEXT_ITEMは順序番号がもっとも低い項目にナビゲートします。順序番号が低い項目がない場合、NEXT_ITEMは現項目にナビゲートします。

妥当性チェックの単位が項目である場合、NEXT_ITEMを使用して、順序番号が現項目より大きいかターゲット項目より小さいフィールドの妥当性チェックを行います。

ブロック内で最後のナビゲート可能な項目からのNEXT_ITEMの機能は、「ナビゲーション形式」ブロック・プロパティの設定により変わります。「ナビゲーション形式」の有効設定値としては、次のようなものがあります。

同一レコード（デフォルト）:ブロックの最後の項目からの「次の項目へ」操作は、入力フォーカスを、同一レコード内の、ブロックの最初にナビゲート可能な項目へ移します。

レコード変更:ブロックの最後の項目から「次の項目へ」操作すると、次のレコード内の最初のナビゲート可能項目へ入力フォーカスが移ります。現レコードがそのブロック内の最後のレコードであり、現在実行中の問合せがなければ、Form Builderにより新規レコードが作成されます。そのブロックに現在実行中の問合せがある（そのブロックに問合せ対象レコードが含まれている）場合には、Oracleフォームにより必要に応じて追加レコードが取り出されます。

ブロック変更: ブロックの最後の項目から「次の項目へ」操作をすると、次のブロックの最初のレコードの最初のナビゲート可能項目に入力フォーカスが移ります。

構文

```
PROCEDURE NEXT_ITEM;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

NEXT_ITEMの例

NEXT_BLOCKの例を参照してください。

NEXT_KEYビルトイン

説明

順序番号が現項目の次に高い、変更可能でナビゲート可能な主キー項目にナビゲートします。そのような項目がない場合、NEXT_KEYを使用して、順序番号がもっとも低い変更可能でナビゲート可能な主キー項目にナビゲートします。現ブロック内に主キー項目がない場合はエラーとなります。

妥当性チェックの単位が項目である場合、NEXT_KEYを使用して、順序番号が現項目より大きいかターゲット項目より小さいフィールドの妥当性チェックを行います。

構文

```
PROCEDURE NEXT_KEY;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

NEXT_KEYの例

```
/*
** Built-in:NEXT_KEY
** Example:Jump the cursor to the next primary key item in
**           in the current block.
*/
BEGIN
  Next_Key;
END;
```

NEXT_MENU_ITEMビルトイン

説明

現メニュー内の次のメニュー項目にナビゲートします。

構文

```
PROCEDURE NEXT_MENU_ITEM;
```

ビルトイン・タイプ

制限付きプロシージャ

パラメータ

なし

NEXT_MENU_ITEMの制限事項

NEXT_MENU_ITEMは、全画面メニュー表示形式で実行中のカスタム・メニュー内でのみ有効です。

NEXT_RECORDビルトイン

説明

順序番号が現レコードの次に高いレコード内の最初の変更可能でナビゲート可能な項目にナビゲートします。そのようなレコードがない場合は、Form Builderではレコードがフェッチされるか、作成されます。現レコードが新規レコードである場合、NEXT_RECORDは失敗します。

構文

```
PROCEDURE NEXT_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

NEXT_RECORDの制限事項

問合せ入力モードでは許可されません。

NEXT_RECORDの例

```
/*
** Built-in:NEXT_RECORD
** Example:If the current item is the last item in the
**         block, then skip to the next record instead of
**         the default of going back to the first item in
**         the same block
** Trigger:Key-Next-Item
*/
DECLARE
  cur_itm VARCHAR2(80) := :System.Cursor_Item;
  cur_blk VARCHAR2(80) := :System.Cursor_Block;
  lst_itm VARCHAR2(80);
BEGIN
  lst_itm := cur_blk||'.'||Get_Block_Property(cur_blk,LAST_ITEM);
  IF cur_itm = lst_itm THEN
    Next_Record;
  ELSE
    Next_Item;
  END IF;
END;
```

NEXT_SETビルトイン

説明

データベースから次のレコード・セットをフェッチし、そのフェッチにより取り出される最初のレコードにナビゲートします。NEXT_SETは、問合せが現ブロック内でオープンされている場合においてのみ成功します。

構文

```
PROCEDURE NEXT_SET;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

NEXT_SETの例

```
/*
** Built-in:NEXT_SET
** Example:Fetch the next set of records from the database
**          when a button is pressed.
** Trigger:When-Button-Pressed
*/
BEGIN
  Next_Set;
END;
```

OLEVAR_EMPTYビルトイン

説明

VT_EMPTY型のOLEバリアントです。

構文

```
OLEVAR_EMPTY OLEVAR;
```

使用上の注意

これは設定できない変数です。OLEコールに対し、空のまたは存在しない引数を渡すときに使用すると便利です。

OPEN_FORMビルトイン

説明

指示されたフォームをオープンします。OPEN_FORMを使用すると、複数フォーム・アプリケーション、つまり同時に複数のフォームをオープンするアプリケーションを作成できます。

構文

```
PROCEDURE OPEN_FORM
  (form_name  VARCHAR2);
PROCEDURE OPEN_FORM
  (form_name     VARCHAR2,
   activate_mode  NUMBER);
```

```
PROCEDURE OPEN_FORM
  (form_name     VARCHAR2,
   activate_mode  NUMBER,
   session_mode   NUMBER);
PROCEDURE OPEN_FORM
  (form_name     VARCHAR2,
   activate_mode  NUMBER,
   session_mode   NUMBER,
   data_mode     NUMBER);
PROCEDURE OPEN_FORM
  (form_name      VARCHAR2,
   activate_mode   NUMBER,
   session_mode    NUMBER,
   paramlist_name  VARCHAR2);
PROCEDURE OPEN_FORM
  (form_name     VARCHAR2,
   activate_mode  NUMBER,
   session_mode   NUMBER,
   paramlist_id   PARAMLIST);
PROCEDURE OPEN_FORM
  (form_name      VARCHAR2,
   activate_mode   NUMBER,
   session_mode    NUMBER,
   data_mode      NUMBER,
   paramlist_name  VARCHAR2);
PROCEDURE OPEN_FORM
  (form_name     VARCHAR2,
   activate_mode  NUMBER,
   session_mode   NUMBER,
   data_mode     NUMBER,
   paramlist_id   PARAMLIST);
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

form_name	**オープンするフォームの名前です。データ型はVARCHAR2です。*必須*。**

activate_mode	ACTIVATE（デフォルト）	フォームにフォーカスを設定し、アプリケーションの中でそれをアクティブ・フォームにします。
	NO_ACTIVATE	フォームをオープンしますが、そのフォームにフォーカスを設定しません。現フォームは現フォームのままとなります。
session_mode	NO_SESSION（デフォルト）	オープンされたフォームが、現フォームと同じデータベース・セッションを共有するよう指定します。どのフォームにおいてもPOST操作またはCOMMIT操作を行うと、同一セッション内で実行中のすべてのフォームについて、ポスト、妥当性チェックおよびコミット処理が行われます。
	SESSION	オープンされたフォームについて新規の別のデータベース・セッションが作成されるよう指定します。
data_mode	NO_SHARE_LIBRARY_DATA（デフォルト）	実行時に、Form Builderは（設計時に）同一のライブラリが付加されているフォームの間でデータを共有します。
	SHARE_LIBRARY_DATA	実行時に、Form Builderは（設計時に）同一のライブラリが付加されているフォームの間でデータを共有します。
paramlist_name	オープンされたフォームに渡されるパラメータ・リストの名前。データ型はVARCHAR2です。	
paramlist_id	Form Builderによって、パラメータ・リスト作成時に割り当てられる一意のID。GET_PARAMETER_LISTビルトインを使用して、PARAMLIST型の変数にIDを戻します。	

使用上の注意

- ACTIVATEを指定したフォームをオープンしても、またNO_ACTIVATEを指定したフォームをオープンしても、通常起動する起動トリガーはオープンされたフォーム内で実行されます。（ただし、以下に示すSESSION指定に関する使用上の注意を参照してください。
- ACTIVATEを指定したフォームをオープン（デフォルト）すると、そのオープンされたフォームは即時にフォーカスを受け取ります。OPEN_FORMのコールに続くトリガー文は実行されません。
- NO_ACTIVATEを指定したフォームをオープンすると、OPEN_FORMのコールに続くトリガー文は、そのオープンされたフォームがメモリーにロードされ、その初期起動トリガーが起動した後に実行されます。
- SESSIONを指定したフォームをオープンすると、PRE-LOGON、ON-LOGONおよびPOST-LOGONの各トリガーは起動しません。
- OPEN_FORMビルトインを発行するフォームがQUERY_ONLYモードで実行されている場合、オープンされたフォームもQUERY_ONLYモードで実行されます。
- Microsoft Windows上では、独立フォームをオープンするフォーム内のウィンドウが最大化さ

れると、そのオープンされたフォームで表示される最初のウィンドウも、その元の設計時の設定にかかわらず、最大化されます。（ウィンドウのGUI表示状態は、Window_Stateプロパティにより制御されます。）

- ほとんどのアプリケーションにおいては、ルート・ウィンドウを含むフォームに対するOPEN_FORMの使用は避けてください。1度に表示できるルート・ウィンドウは1つのみなので、現フォーム内およびオープンされたフォーム内のルート・ウィンドウに割り当てられるキャンバスは同一ウィンドウ内に表示されます。これにより、オープンしているフォームが、オリジナルのフォームからルート・ウィンドウを"引継ぐ"ことになり、オリジナル・フォームのキャンバスを部分的または完全に隠します。

OPEN_FORMの制限事項

- FORMSnn_SESSION環境変数を TRUEに設定すれば、すべてのランフォーム起動のセッションを「オン」に設定できます。FORMSnn_SESSION変数を設定すると、ランフォームのコマンド・ラインから「Session」オプションを設定して上書きしない限り、すべてのランフォームは、その設定を継承します。
- OPEN_FORMを使用して複数フォーム・アプリケーションを作成するときに*session_mode*を「SESSION」に設定すると、*data_mode*をSHARE_LIBRARY_DATAに設定できません（Form Builderによりランタイム・エラー・メッセージが表示されます）。

PASTE_REGIONビルトイン

説明

クリップボードの内容（つまり、最後にカットまたはコピーされた選択領域）をペーストし、ペースト領域の左上コーナーをカーソル位置に配置します。

構文

```
PROCEDURE PASTE_REGION;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

使用上の注意

PASTE_REGIONは、その他の編集機能と同じくテキスト項目およびイメージ項目上でのみ使用します。カット・アンド・コピー機能では選択済み領域はシステム・クリップボードに転送され、ペースト・ターゲットが指定されるまでシステム・クリップボード内にあります。ペースト・ターゲットが指定されると、カットまたはコピーされた内容がターゲット位置にペーストされます。

PAUSEビルトイン

説明

エンド・ユーザーがファンクション・キーを押すまで処理を中断します。PAUSEでは警告が表示される場合があります。

構文

```
PROCEDURE PAUSE;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

説明

エンド・ユーザーがファンクション・キーを押すまで処理を中断します。PAUSEでは警告が表示される場合があります。

パラメータ

なし

PLAY_SOUNDビルトイン

説明

指定されたサウンド項目内のサウンド・オブジェクトを再生します。

構文

```
PLAY_SOUND(item_id ITEM);
```

```
PLAY_SOUND(item_name VARCHAR2);
```

ビルトイン・タイプ

制限

問合せ入力モード

不可

パラメータ

item_id	ユーザーがサウンド項目作成時に、Form Builderにより与えられた一意のID。
item_name	ユーザーがサウンド項目作成時に与えた名前。

PLAY_SOUNDの例

```
/* Example 1:This procedure call (attached to a menu item)
** plays a sound object from the specified sound item:
*/
GO_ITEM('about.abc_inc');
PLAY_SOUND('about.abc_inc');

/* Example 2:These procedure calls (attached to a
** When-Button-Pressed trigger) read a sound object from the
** file system and play it.Note:since an item must have focus
** in order to play a sound, the trigger code includes a call
** to the built-in procedure GO_ITEM:
*/
BEGIN
  IF :clerks.last_name EQ 'BARNES' THEN
    GO_ITEM('orders.filled_by');
    READ_SOUND_FILE('t:¥orders¥clerk¥barnes.wav',
                 'wave',
                 'orders.filled_by');
    PLAY_SOUND('orders.filled_by');
  END IF;
END;
```

POPULATE_GROUPビルトイン

説明

レコード・グループに関連する問合せを実行し、その問合せの成功または失敗を表わす数値を戻します。問合せが成功すると、POPULATE_GROUPにより0（ゼロ）が戻されます。問合せが失敗すると、そのSELECT文の失敗に対応するORACLEエラー番号が生成されます。問合せ成功の結果、取り出される行により、そのグループ内に存在する行が置換されます。

注意: POPULATE_GROUP配列では、1度に100レコードがフェッチされることに注意してください。ネットワーク・パフォーマンスを向上させるために、問合せを制限し、ネットワーク通信量を制限する場合もあります。

構文

```
FUNCTION POPULATE_GROUP
  (recordgroup_id  RecordGroup);
FUNCTION POPULATE_GROUP
(recordgroup_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

問合せ入力モード

可

パラメータ

recordgroup_id	**Form Builderがグループを作成するときに割り当てる一意のID。そのIDのデータ型はRECORDGROUPです。**
recordgroup_name	**ユーザーがレコード・グループを作成するときに付けた名前。その名前のデータ型はVARCHAR2です。**

POPULATE_GROUPの制限事項

次のレコード・グループに対してのみ有効です。

- 設計時に問合せとともに作成されたレコード・グループ
- CREATE_GROUP_FROM_QUERYビルトインのコールにより作成されたレコード・グループ
- それまでに、POPULATE_GROUP_WITH_QUERYビルトイン（問合せをそのレコード・グルー

プに関連付ける）が移入されているレコード・グループ

POPULATE_GROUPの例

GET_GROUP_ROW_COUNTまたはCREATE_GROUP_FROM_QUERYの例を参照してください。

POPULATE_GROUP_FROM_TREEビルトイン

説明

階層ツリーのデータを使用してレコード・グループを生成します。

構文

```
PROCEDURE POPULATE_GROUP_FROM_TREE
  (group_name VARCHAR2,
   item_name VARCHAR2,
   node NODE);
PROCEDURE POPULATE_GROUP_FROM_TREE
  (group_name VARCHAR2,
   item_id ITEM,
   node NODE);
PROCEDURE POPULATE_GROUP_FROM_TREE
  (group_id RECORDGROUP,
   item_name VARCHAR2,
   node NODE);
PROCEDURE POPULATE_GROUP_FROM_TREE
  (group_id RECORDGROUP,
   item_id ITEM,
   node NODE);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

group_name	グループの名前を指定します。
group_id	グループに割り当てられたIDを指定します。

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
node	有効なノードを指定します。指定された場合、指定ノードも含め、レコード・グループの生成に使用されたサブツリーを示します。

使用上の注意

レコード・グループは、階層ツリーのデータ・セットを挿入する前に消去されます。

POPULATE_GROUP_FROM_TREEの例

```
/*
** Built-in:POPULATE_GROUP_FROM_TREE
*/

-- This code will transfer all the data from a hierarchical tree
-- that is parented by the node with a label of "Zetie" to a
-- pre-created record group.Please see the documentation
-- for the structure of the required record group.

DECLARE
   htree        ITEM;
   find_node     NODE;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Find the node with a label "Zetie".
   find_node := Ftree.Find_Tree_Node(htree, 'Zetie', Ftree.FIND_NEXT,
   Ftree.NODE_LABEL, Ftree.ROOT_NODE, Ftree.ROOT_NODE);

   -- Populate the record group with the tree data.
   -- The record group must already exist.
   Ftree.Populate_Group_From_Tree('tree_data_rg', htree, find_node);
END;
```

POPULATE_GROUP_WITH_QUERYビルトイン

説明

所定の問合せでレコード・グループへ挿入を行います。そのレコード・グループは消去され、フェッチされた行によりそのレコード・グループ内の既存の行が置換されます。

SELECT文が失敗すると、Form BuilderによりORACLEエラー番号が戻されます。問合せが成功すれば、このビルトインにより0（ゼロ）が戻されます。

このビルトインは、次のいずれかのコールによって作成されたレコード・グループの挿入に使用できます。

■ CREATE_GROUPビルトイン

■ CREATE_GROUP_FROM_QUERYビルトイン

このビルトインを使用すると､指示された問合せがそのグループのデフォルトの問合せとなり、POPULATE_GROUPのビルトイン項目が後にコールされるたびに実行されます。

注意: POPULATE_GROUP_WITH_QUERY配列では、1度に20レコードがフェッチされることに注意してください。ネットワーク・パフォーマンスを向上させるために、問合せを制限し、ネットワーク通信量を制限する場合もあります。

構文

```
FUNCTION POPULATE_GROUP_WITH_QUERY
(recordgroup_id  RecordGroup,
 query      VARCHAR2);
FUNCTION POPULATE_GROUP_WITH_QUERY
  (recordgroup_name  VARCHAR2,
   query             VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

NUMBER

問合せ入力モード

可

パラメータ

recordgroup_id	Form Builderがグループを作成するときに割り当てる一意のID。そのIDのデータ型はRECORDGROUPです。
recordgroup_name	ユーザーがレコード・グループを作成するときに付けた名前。その名前のデータ型はVARCHAR2です。
query	引用符（'）に囲まれた有効なSELECT文。問合せの結果として取り出された列はすべて、表の中の列のデータ型をとります。問合せを表内の列のサブセットに制限すると、レコード・グループにはそれらの列のみが作成されます。問合せのデータ型はVARCHAR2です。

POPULATE_GROUP_WITH_QUERYの制限事項

- SELECT文に指定された列の数値および型がレコード・グループ列に一致している必要があります。

POPULATE_GROUP_WITH_QUERYの例

CREATE_GROUPの例を参照してください。

POPULATE_LISTビルトイン

説明

現リストの内容を削除し、レコード・グループの値をそのリストに挿入します。そのレコード・グループは実行時に作成してください。また、次の2つの列（VARCHAR2）構造を持っている必要があります。

列1: **列2:**

リスト・ラベル　リスト値

構文

```
PROCEDURE POPULATE_LIST
  (list_id    ITEM,
   recgrp_id  RecordGroup);
PROCEDURE POPULATE_LIST
  (list_id     ITEM,
   recgrp_name  VARCHAR2);
PROCEDURE POPULATE_LIST
  (list_name  VARCHAR2,
   recgrp_id  RecordGroup);
PROCEDURE POPULATE_LIST
```

```
(list_name    VARCHAR2,
 recgrp_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

list_id	Form Builderがリスト項目を作成するときに割り当てる一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
list_name	ユーザーがリスト項目を作成するときに付けた名前。その名前のデータ型はVARCHAR2です。
recgrp_id	Form Builderによりレコード・グループ作成時に割り当てられる一意のIDを指定します。そのIDのデータ型はRECORDGROUPです。
recgrp_name	ユーザーがレコード・グループ作成時に与えたVARCHAR2名。

使用上の注意

- 「ほかの値のマッピング」プロパティが定義されていて、そのブロック内に問合せ対象レコードがある場合は、POPULATE_LISTビルトインを使用しないでください。このような場合にCLEAR_LISTビルトインを使用すると、すでにフェッチされたレコードを表示できなくなる場合があります。

  たとえば、値A、BおよびCを含んでいるリスト項目があり、「ほかの値のマッピング」プロパティが定義されているとします。また、これらの値はデータベースからフェッチされたものであるとします（問合せはオープンしている）。ここで、ユーザーがPOPULATE_LISTを使用してこのリストを挿入すると、エラーが起こります。その理由は、Form Builderではそれまでにフェッチされた値（A、BおよびC）を表示しようとしますが、これらの値がリストから除去され、新しい値に置換されているため、表示できないからです。

- リストを挿入する前に、オープンされている問合せをクローズします。オープンされた問合せのクローズには、ABORT_QUERYビルトインを使用します。

POPULATE_LISTの制限事項

POPULATE_LISTによって、次の場合に、「エラーFRM-41337: レコード・グループからリストを占有できません。」が戻されます。

- このレコード・グループにデフォルト値の要素もその他の値要素も含まれておらず、このリ

ストが、DELETE_LIST_ELEMENTでこれらの要素を削除する際の基準に適合しない場合。詳細は、DELETE_LIST_ELEMENTの制限事項を参照してください。

- そのレコード・グループに別の値の要素が含まれているが、そのリストが、ADD_LIST_ELEMENTにより別の値の要素を追加する際の基準に適合しない場合。詳細は、ADD_LIST_ELEMENTの制限事項を参照してください。

POPULATE_LISTの例

```
/*
**  Built-in:POPULATE_LIST
**  Example:Retrieves the values from the current list item
**           into record group one, clears the list, and
**           populates the list with values from record group
**           two when a button is pressed.
**  Trigger:When-Button-Pressed
*/
BEGIN
   Retrieve_List(list_id, 'RECGRP_ONE');
   Clear_List(list_id);
   Populate_List(list_id, 'RECGRP_TWO');
END;
```

POPULATE_TREEビルトイン

説明

階層ツリー内にある既存データを消去し、「レコード・グループ」プロパティまたは「データ問合せ」プロパティによって指定されたデータ・セットを取得します。

構文

```
PROCEDURE POPULATE_TREE
  (item_name VARCHAR2);
PROCEDURE POPULATE_TREE
  (item_id ITEM);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。

POPULATE_TREEの例

```
/*
** Built-in:POPULATE_TREE
*/

-- This code will cause a tree to be re-populated using
-- either the record group or query already specified
-- for the hierarchical tree.

DECLARE
htree        ITEM;
   top_node     FTREE.NODE;
   find_node    FTREE.NODE;
BEGIN
-- Find the tree itself.
htree := Find_Item('tree_block.htree3');

-- Populate the tree with data.
Ftree.Populate_Tree(htree);
END;
```

POSTビルトイン

説明

フォームのデータをデータベースに書き込みますが、データベース・コミットは実行しません。Form Builderでは最初にフォームの妥当性チェックが実行されます。データベースにポストする変更内容がある場合、フォームの各ブロックについて、そのデータベースに書込み、削除、挿入および更新が行われます。

データベースにポストするデータはすべて、現ランフォーム・セッションで実行される次のCOMMIT_FORMにより、そのデータベースに対してコミットされます。また、このデータは次のCLEAR_FORMによってロールバックもできます。

構文

```
PROCEDURE POST;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

使用上の注意

このフォームがNO_SESSIONパラメータを指定したOPEN_FORMによってコールされた場合、POSTはデータの妥当性チェックを行い、このフォームとコール側フォームの両方にデータを書き込みます。

POSTの例

```
/*
** Built-in:POST and EXIT_FORM
** Example:Leave the called form, without rolling back the
**          posted changes so they may be posted and
**          committed by the calling form as part of the
**          same transaction.
*/
BEGIN
  Post;

  /*
  ** Form_Status should be 'QUERY' if all records were
  ** successfully posted.
  */
  IF :System.Form_Status <> 'QUERY' THEN
    Message('An error prevented the system from posting changes');
    RAISE Form_Trigger_Failure;
  END IF;
  /*
  ** By default, Exit_Form asks to commit and performs a
  ** rollback to savepoint.We've already posted, so we do
  ** not need to commit, and we don't want the posted changes
```

```
  ** to be rolled back.
  */
  Exit_Form(NO_COMMIT, NO_ROLLBACK);
END;
```

PREVIOUS_BLOCKビルトイン

説明

ナビゲーション順序内の前の入力可能なブロックにある最初のナビゲート可能な項目にナビゲートします。デフォルトでは、ナビゲーション順序内の前のブロックは、オブジェクト・ナビゲータ内のブロック順序で定義した通り、順序番号が次に低いブロックです。ただし、「前のナビゲーション・データ・ブロック」ブロック・プロパティの設定では、ナビゲーションを目的として、異なるブロックを前のブロックとして指定できます。

それより順序の低い入力可能なブロックがない場合、PREVIOUS_BLOCKを使用して、順序番号が最大の入力可能なブロックにナビゲートします。

構文

```
PROCEDURE PREVIOUS_BLOCK;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

PREVIOUS_BLOCKの例

```
/*
** Built-in:PREVIOUS_BLOCK
** Example:If the current item is the first item in the
**          block, then skip back the previous block
**          instead of the default of going to the last
**          item in the same block
** Trigger:Key-Previous-Item
*/
DECLARE
```

```
  cur_itm VARCHAR2(80) := :System.Cursor_Item;
  cur_blk VARCHAR2(80) := :System.Cursor_Block;
  frs_itm VARCHAR2(80);
BEGIN
  frs_itm := cur_blk||'.'||Get_Block_Property(cur_blk,FIRST_ITEM);
  IF cur_itm = frs_itm THEN
    Previous_Block;
  ELSE
    Previous_Item;
  END IF;
END;
```

PREVIOUS_FORMビルトイン

説明

複数フォーム・アプリケーションでは、順序番号が次に低いフォームにナビゲートします。（フォームは実行時に起動された順序で順序付けされます。）それより順序番号が低いフォームがない場合、PREVIOUS_FORMは順序番号が最大のフォームにナビゲートします。そのようなフォームがない場合には、現フォームは現フォームのままとなります。

PREVIOUS_FORMでナビゲートするときには、妥当性チェックは行われず、ナビゲーションを開始するフォームで起動されるWHEN-WINDOW-DEACTIVATEDトリガー、およびターゲット・フォームで起動されるWHEN-WINDOW-ACTIVATEDトリガー以外には、トリガーが起動しません。

構文

```
PROCEDURE PREVIOUS_FORM;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

PREVIOUS_FORMの制限事項

CALL_FORMがある別のフォームを起動した結果として現在使用禁止となっているフォームはターゲット・フォームにはできません。

PREVIOUS_ITEMビルトイン

説明

順序番号が現項目の次に低いナビゲート可能な項目にナビゲートします。該当項目がない場合、PREVIOUS_ITEMは順序番号が最大のナビゲート可能な項目にナビゲートします。そのような項目がない場合、PREVIOUS_ITEMは現項目にナビゲートします。

そのブロック内の最初のナビゲート可能な項目からのPREVIOUS_ITEM機能は、「ナビゲーション形式」ブロック・プロパティの設定によって変わります。「ナビゲーション形式」の有効設定値としては、次のようなものがあります。

同一レコード（デフォルト）**:** ブロックの最初の項目から「前の項目へ」操作を行うと、同一レコード内の、ブロックの最後にナビゲート可能な項目へ入力フォーカスが移ります。

レコード変更: ブロックの最初の項目から「前の項目へ」操作を行うと、前のレコード内の、ブロックの最後にナビゲート可能な項目へ入力フォーカスが移ります。

ブロック変更: ブロックの最初の項目から「前の項目」操作を行うと、前のブロック内の、現レコード内の、ナビゲート可能な最後の項目へ入力フォーカスが移ります。

構文

```
PROCEDURE PREVIOUS_ITEM;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

PREVIOUS_ITEMの例

PREVIOUS_BLOCKの例を参照してください。

PREVIOUS_MENUビルトイン

説明

PREVIOUS_MENUを使用して、前のメニュー内で前にアクティブであった項目にナビゲートします。

構文

```
PROCEDURE PREVIOUS_MENU;
```

ビルトイン・タイプ

制限付きプロシージャ

パラメータ

なし

PREVIOUS_MENUの制限事項

PREVIOUS_MENUは、全画面メニューおよびバー・メニューにのみ適用されます。

PREVIOUS_MENU_ITEMビルトイン

説明

PREVIOUS_MENU_ITEMを使用して、現メニュー内の前のメニュー項目にナビゲートします。

構文

```
PROCEDURE PREVIOUS_MENU_ITEM;
```

ビルトイン・タイプ

制限付きプロシージャ

パラメータ

なし

PREVIOUS_MENU_ITEMの制限事項

PREVIOUS_MENU_ITEMは、全画面メニューにのみ適用されます。

PREVIOUS_RECORDビルトイン

説明

順序番号が現レコードの次に低いレコード内の最初の変更可能でナビゲート可能な項目にナビゲートします。

構文

```
PROCEDURE PREVIOUS_RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

PREVIOUS_RECORDの例

```
/*
** Built-in:PREVIOUS_RECORD
** Example:If the current item is the first item in the
**          block, then skip back to the previous record
**          instead of the default of going to the last
**          item in the same block
** Trigger:Key-Previous-Item
*/
DECLARE
  cur_itm VARCHAR2(80) := :System.Cursor_Item;
  cur_blk VARCHAR2(80) := :System.Cursor_Block;
  frs_itm VARCHAR2(80);
BEGIN
  frs_itm := cur_blk||'.'||Get_Block_Property(cur_blk,FIRST_ITEM);
  IF cur_itm = frs_itm THEN
    Previous_Record;
  ELSE
    Previous_Item;
  END IF;
END;
```

PRINTビルトイン

説明

現ウィンドウをファイルまたはプリンタに印刷します。

構文

```
PROCEDURE PRINT;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

PRINTの例

```
/*
** Built-in:PRINT
** Example:Print the current window.
*/
BEGIN
  Print;
END;
```

PTR_TO_VARビルトイン

説明

最初に、入力されたアドレスを含む型VT_PTRのOLEバリアントを作成します。次に、そのバリアントを渡し、関数VARPTR_TO_VARによって入力します。

構文

```
FUNCTION PTR_TO_VAR
  (address PLS_INTEGER, vtype VT_TYPE)
RETURN newvar OLEVAR;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

作成および変換されたOLEバリアント

パラメータ

address	値がアドレスである変数。
vtype	変更行のOLEバリアントに付与される型（VARPTR_TO_VARによって処理された後）。

使用上の注意

大部分のアプリケーションでは、このファンクションを使用する必要はありません。ファンクションを使用する場合、正しいアドレス値が新しいバリアント内に配置されるように注意する必要があります。

QUERY_PARAMETERビルトイン

説明

指定された置換パラメータの現行の設定値を示す「問合せパラメータ」ダイアログを表示します。エンド・ユーザーは、ユーザーがリストに組み込むパラメータの値を設定できます。

「問合せパラメータ」ダイアログはモーダル・ダイアログであり、制御は、エンド・ユーザーがそのダイアログを受け入れるか取り消すまで、制御はそのコール・トリガーまたはプロシージャに戻りません。このため、QUERY_PARAMETERのコールに引き続くPL/SQL文は、「問合せパラメータ」ダイアログが消されるまで実行されません。

構文

```
PROCEDURE QUERY_PARAMETER
(parameter_string  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

parameter_string	メニュー項目用の置換パラメータの文字列を指定します。parameter_stringパラメータを指定する構文には、アンパサンドを伴う&parm_nameが必要です。置換パラメータは、すべてのバインド変数に使用されるPL/SQLのコロン構文":param_name"で参照されます。

QUERY_PARAMETERの例

```
/*
** Built-in:QUERY_PARAMETER
** Example:Prompt for several menu parameters
**          programmatically, validating their contents.
*/
PROCEDURE Update_Warehouse IS
  validation_Err BOOLEAN;
BEGIN
  WHILE TRUE LOOP
    Query_Parameter('&p1 &q2 &z6');
    /*
    ** If the user did not Cancel the box the Menu_Success
    ** function will return boolean TRUE.
    */
    IF Menu_Success THEN
      IF TO_NUMBER( :q2 ) NOT BETWEEN 100 AND 5000 THEN
      Message('Qty must be in the range 100..5000');
      Bell;
      Validation_Err := TRUE;
        END IF;
    /*
    ** Start a sub-block so we can catch a Value_Error
    ** exception in a local handler
    */
      BEGIN
    IF TO_DATE( :z6 ) < SYSDATE THEN
      Message('Target Date must name a day in the future.');
      Bell;
      Validation_Err := TRUE;
    END IF;
      EXCEPTION
    WHEN VALUE_ERROR THEN
      Message('Target Date must be of the form DD-MON-YY');
      Bell;
      Validation_Err := TRUE;
      END;
```

```
      /*
      ** If we get here, all parameters were valid so do the
      ** Update Statement.
      */
      IF NOT Validation_Err THEN
    UPDATE WAREHOUSE
      SET QTY_TO_ORDER = QTY_TO_ORDER*0.18
    WHERE TARGET_DATE = TO_DATE(:z6)
      AND QTY_ON_HAND > TO_NUMBER(:q2)
      AND COST_CODE LIKE :p1||'%';
      END IF;
    ELSE
      /*
      ** If Menu_Success is boolean false, then return back
      ** from the procedure since user cancelled the dialog
      */
      RETURN;
    END IF;
  END LOOP;
END;
```

READ_IMAGE_FILEビルトイン

説明

所定のファイルから所定のタイプのイメージを読み込んで、Form Builderのイメージ項目にそれを表示します。

構文

```
PROCEDURE READ_IMAGE_FILE
  (file_name  VARCHAR2,
   file_type  VARCHAR2,
   item_id    ITEM);
PROCEDURE READ_IMAGE_FILE
  (file_name  VARCHAR2,
   file_type  VARCHAR2,
   item_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

file_name	有効なファイル名。ファイル名指定には、ご使用のオペレーティング・システムに固有のフルパス文を組み込めます。
file_type	有効なイメージ・ファイル・タイプは、BMP、CALS、GIF、JFIF、JPG、PICT、RAS、TIFF、またはTPICです。（注意: ファイル・タイプは、Form Builderがソース・イメージ・ファイルから推測するため、任意指定です。ただし、パフォーマンスを最適化するには、ファイル・タイプを指定してください。）
item_id	Form Builderによりイメージ項目の作成時に割り当てられる一意のID。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。データ型はITEMです。
item_name	ユーザーがイメージ項目の作成時に与えた名前。データ型はVARCHAR2です。

使用上の注意

Form Builderは.FMXファイル検索時に、同一のデフォルト・パス上のイメージ・ファイルを検索します。使用しているプラットフォームの固有の検索パスの詳細は、使用しているオペレーティング・システムのForm Builderのマニュアルを参照してください。

READ_IMAGE_FILEの例

```
/* Read an image from the filesystem into an image item on the
** form.In this example, the scanned picture identification
** for each employee is NOT saved to the database, but is
** stored on the filesystem.An employee's  photo is a TIFF
** image stored in a file named <Userid>.TIF Each employee's
** Userid is unique.
** Trigger:Post-Query
*/
DECLARE
  tiff_image_dir VARCHAR2(80) := '/usr/staff/photos/';
  photo_filename VARCHAR2(80);
BEGIN
  /*
  ** Set the message level high so we can gracefully handle
  ** an error reading the file if it occurs
  */
  :System.Message_Level := '25';
  /*
  ** After fetching an employee record, take the employee's
  ** Userid and concatenate the '.TIF' extension to derive
  ** the filename from which to load the TIFF image.The EMP
```

```
  ** record has a non-database image item named 'EMP_PHOTO'
  ** into which we read the image.
  */
  photo_filename := tiff_image_dir||LOWER(:emp.userid)||'.tif';

  /*
  ** For example 'photo_filename' might look like:
  **
  **    /usr/staff/photos/jgetty.tif
  **                     ------
  **
  ** Now, read in the appropriate image.
  */

  READ_IMAGE_FILE(photo_filename, 'TIFF', 'emp.emp_photo');
  IF NOT FORM_SUCCESS THEN
    MESSAGE('This employee does not have a photo on file.');
  END IF;
  :SYSTEM.MESSAGE_LEVEL := '0';
END;
```

READ_SOUND_FILEビルトイン

説明

指定ファイルから、指定サウンド項目にサウンド・オブジェクトを読み込みます。

構文

```
READ_SOUND_FILE(file_name VARCHAR2,
          file_type VARCHAR2,
          item_id ITEM);
READ_SOUND_FILE(file_name VARCHAR2,
          file_type VARCHAR2,
          item_name VARCHAR2);
```

ビルトイン・タイプ

非制限

問合せ入力モード

可

パラメータ

file_name	読込み対象のサウンド・オブジェクトを含むファイルの完全修飾ファイル名。
file_type	そのサウンド・データ・ファイルのファイル・タイプ。有効な値は、AU、AIFF、AIFF-C、およびWAVEです（注意: ファイル・タイプは任意指定ですが、すでにわかっていれば、パフォーマンス向上のために指定してください）。
item_id	ユーザーがサウンド項目作成時に、Form Builderにより与えられた一意のID。
item_name	ユーザーがサウンド項目作成時に与えた名前。

使用上の注意

■ サウンド・ファイルのファイル・タイプ指定は任意指定です。ただし、ファイル・タイプを知っていれば、それを指定することでパフォーマンスを向上させることができます。

READ_SOUND_FILEの例

```
/* These procedure calls (attached to a When-Button-Pressed
** trigger) reads a sound object from the file system and plays
** it.Note:since a sound item must have focus in order to play
** a sound object, the trigger code includes a call to the
** built-in procedure GO_ITEM:
*/
BEGIN
  IF :clerks.last_name EQ 'BARNES' THEN
    GO_ITEM('orders.filled_by');
    READ_SOUND_FILE('t:¥orders¥clerk¥barnes.wav',
                    'wave',
                    'orders.filled_by');
    PLAY_SOUND('orders.filled_by');
  END IF;
END;
```

RECALCULATEビルトイン

説明

指定された計算項目（そのブロックの各レコード内）の値を再計算のためにマークします。一般的に、その計算式（またはそれによって起動されるファンクションまたはプロシージャ）により、異なる値を戻すシステム変数またはビルトイン関数が参照される場合に、このビルトインが起動されることになります。

実際の計算は即時には行われないことに注意してください。計算は、項目がマークされた後に、計算済み項目が参照されるか、またはエンド・ユーザーに表示される前に行われます。使用し

ているアプリケーション・ロジックは、特定時刻に行われる計算項目の再計算には依存しません。

構文

```
PROCEDURE RECALCULATE
  (item_name  VARCHAR2);
PROCEDURE RECALCULATE
  (item_id  Item);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_name	ユーザーが項目定義時に与えた名前。データ型はVARCHAR2です。
item_id	Form Builderによりその項目の作成時に割り当てられた一意のID。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。データ型はITEMです。

RECALCULATEの制限事項

RECALCULATEビルトインを使用して、計算項目のみを再計算することができます。RECALCULATEの引数としてサマリー項目（または非計算項目）を指定すると、Form Builderにより次のエラー・メッセージが戻されます。

FRM-41379:非式項目: <block_name.item_name>を再計算できません。

REDISPLAYビルトイン

説明

スクリーンを再描画します。これにより、スクリーンに表示されている既存のシステム・メッセージが消去されます。

構文

```
PROCEDURE REDISPLAY;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

RELEASE_OBJビルトイン

説明

OLEオブジェクトへの接続を解除します。

構文

```
PROCEDURE RELEASE_OBJ
  (obj OLEOBJ, kill_persistence_boolean := NULL);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

obj	解除するOLEオブジェクトへのポインタ。
Kill_persistence_boolean	ブール値NULLによってオブジェクトが解除され、その永続性は失われます。 ブール値TRUEによって永続オブジェクトのみが解除されます。永続オブジェクトへのポインタがない場合、コードは誤動作します。 ブール値FALSEによって非永続オブジェクトのみが解除されます。非永続オブジェクトへのポインタがない場合、エラーFRM-40935が表示されます。 これはオプションのパラメータです。入力しない場合のデフォルト値はNULLです（オブジェクトを無条件に解除します）。

使用上の注意

一般的に、オブジェクトを解除した後は、アクセスしないでください。

このプロシージャの条件付きフォーム（ブール値のTRUEまたはFALSE）が使用されるのは、オブジェクトの2つのインスタンスが作成され、各インスタンスが異なる永続性値を持ち、ポインタが曖昧であるといった、稀な場合のみです。プロシージャによって2つのオブジェクトのいずれかが解除され、もう一方が唯一のインスタンスになります。

REPLACE_CONTENT_VIEWビルトイン

説明

指示されたウィンドウに現在表示中の内容キャンバスを異なる内容キャンバスで置換します。

構文

```
PROCEDURE REPLACE_CONTENT_VIEW
  (window_id  Window,
   view_id    ViewPort);
PROCEDURE REPLACE_CONTENT_VIEW
  (window_name  VARCHAR2,
   view_id      ViewPort);
PROCEDURE REPLACE_CONTENT_VIEW
  (window_id  Window,
   view_name  VARCHAR2);
PROCEDURE REPLACE_CONTENT_VIEW
  (window_name  VARCHAR2,
   view_name    VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

window_id	Form Builderがウィンドウ作成時にそのウィンドウに割り当てられる一意のIDを指定します。FIND_WINDOWのビルトインを使用して、そのIDを適切なタイプの変数に戻します。このIDのデータ型はWINDOW型です。
window_name	ウィンドウの作成時に付けた名前を指定します。その名前のデータ型はVARCHAR2です。
view_id	Form Builderによりオブジェクト作成時にそのビューに割り当てられる一意のIDを指定します。FIND_VIEWのビルトインを使用して、そのIDを適切な型の変数に戻します。このIDのデータ型はVIEWPORT型です。
view_name	ユーザーがオブジェクト定義時にそのオブジェクトに与えた名前を指定します。その名前のデータ型はVARCHAR2です。

REPLACE_CONTENT_VIEWの制限事項

- ウィンドウの現コンテント・キャンバスに取って代わるキャンバスは、すでに設計時にそのウィンドウに割り当てられている必要があります。つまり、あるウィンドウのコンテント・ビューを別のウィンドウからのコンテント・ビューで置き換えることはできません。

■ 現在フォーカスを持つ項目を含むコンテント・キャンバスを置換すると、Form Builderでは即時にその置換が取り消され、フォーカス項目がエンド・ユーザーに見える状態が保持されます。

REPLACE_CONTENT_VIEWの例

```
/*
** Built-in:REPLACE_CONTENT_VIEW
** Example:Replace the 'salary' view with the 'history'
**          view in the 'employee_status' window.
*/
BEGIN
  Replace_Content_View('employee_status','history');
END;
```

REPLACE_MENUビルトイン

説明

現メニューを指定メニューで置換しますが、新規メニューをアクティブにしません。また、REPLACE_MENUを使用すると、そのメニューの表示方法およびロールを変更できます。

REPLACE_MENUでは、新規メニューをアクティブにしないため、Form Builderではそのメニューがアクティブ・キャンバスのどの部分であれ、覆い隠されないようにします。したがって、アクティブ・キャンバスがメニューを覆うと、メニュー全体または一部分が画面上で見えなくなります。

構文

```
REPLACE_MENU;

PROCEDURE REPLACE_MENU
  (menu_module_name       VARCHAR2);
PROCEDURE REPLACE_MENU
  (menu_module_name       VARCHAR2,
   menu_type              NUMBER);
PROCEDURE REPLACE_MENU
  (menu_module_name       VARCHAR2,
   menu_type              NUMBER,
   starting_menu_name     VARCHAR2);
PROCEDURE REPLACE_MENU
  (menu_module_name       VARCHAR2,
   menu_type              NUMBER,
   starting_menu          VARCHAR2,
   group_name             VARCHAR2);
PROCEDURE REPLACE_MENU
  (menu_module_name       VARCHAR2,
```

```
    menu_type               NUMBER,
    starting_menu           VARCHAR2,
    group_name              VARCHAR2,
    use_file                BOOLEAN);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

使用上の注意

REPLACE_MENUを使用して、アプリケーション内のすべてのウィンドウについてそのメニューが置換されます。ユーザーがCALL_FORMを使用している場合、REPLACE_MENUでは、コール側フォームとコール先フォームの両方のメニューが指定メニューで置換されます。

パラメータ

menu_module _name	現メニュー・モジュールを置換すべきメニュー・モジュールの名前。データ型はVARCHAR2です。このパラメータは任意です。省略すると、Form Builderはメニューなしのフォームを実行します。
menu_type	そのメニューの表示形式。次の定数は、このパラメータの引数として渡せます。 PULL_DOWN　ほとんどのGUIプラットフォームおよび一定の文字モードのプラットフォームに特徴的なプルダウン形式でメニューを表示するときに、これを指定します。 BAR　ルート・ウィンドウの1番上に水平にまたがるバー形式でメニューを表示するときに、これを指定します。 FULL_SCREEN　全画面形式でメニューを表示するときに、これを指定します。
starting_menu	Form Builderにより開始メニューとして使用されるべきメニュー・モジュール内のメニューを指定します。その名前のデータ型はVARCHAR2です。
group_name	Form Builderにより使用されるべきセキュリティ・ロールを指定します。ロール名を指定しない場合は、Form Builderでは現ユーザー名を使用してロールが判別されます。

use_file	Form Builderにより実行対象のメニュー.MMXファイルを探す方法が指示されます。「メニュー・ソース」フォーム・モジュール・プロパティに対応します。*use_file*のデータ型はBOOLEANです。 NULL　Form Builderが現メニューの「ソース・プロパティ」から読み込みを行い、それに応じてREPLACE_MENUを実行するよう指定します。たとえば、フォーム・モジュール「メニュー・ソース」プロパティが現フォームについて「はい」に設定されている場合、Form Builderでは、*use_file*実パラメータがTRUEであるかのように REPLACE_MENUが実行されます。 FALSE　Form Builderにより、「メニュー・モジュール」値がデータベース内の.MMB（バイナリ）メニュー・モジュールへの参照値として扱われ、このモジュールに対する問合せが実行され.MMX（実行可能）の実名が取得されるよう指定します。 TRUE　Form Builderによりその「メニュー・モジュール」値が、そのファイル・システム内の.MMXメニュー実行ファイルへの直接参照として扱われるよう指定します。

REPLACE_MENUの例

```
/*
** Built-in:REPLACE_MENU
** Example:Use a standard procedure to change which root
**             menu in the current menu application appears in
**             the menu bar.A single menu application may
**             have multiple "root-menus" which an application
**             can dynamically set at runtime.
*/
PROCEDURE Change_Root_To(root_menu_name VARCHAR2) IS
BEGIN
  Replace_Menu('MYAPPLSTD', PULL_DOWN, root_menu_name);
END;
```

REPORT_OBJECT_STATUSビルトイン

説明

フォーム内でRUN_REPORT_OBJECTビルトインによって実行されるレポート・オブジェクトのステータスを提供します。

構文

```
FUNCTION REPORT_OBJECT_STATUS
  (report_id VARCHAR2(20)
);
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

パラメータ

report_id	RUN_REPORT_OBJECTビルトインによって返されるVARCHAR2型の値。この値によって、現在ローカルで実行されているまたはリモートのレポート・サーバー上で実行されているレポートが一意に識別されます。

使用上の注意

このビルトインには、finished、running、canceled、opening_report、enqueued、invalid_job、terminated_with_errorおよびcrashedという、8種類の戻り値があります

REPORT_OBJECT_STATUSの例

```
DECLARE
   repid REPORT_OBJECT;
   v_rep  VARCHAR2(100);
   rep_status varchar2(20);
BEGIN
   repid := find_report_object('report4');
   v_rep := RUN_REPORT_OBJECT(repid);
   rep_status := REPORT_OBJECT_STATUS(v_rep);

   if rep_status = 'FINISHED' then
      message('Report Completed');
      copy_report_object_output(v_rep,'d:¥temp¥local.pdf');
      host('netscape d:¥temp¥local.pdf');
   else
      message('Error when running report.');
   end if;
END;
```

RESET_GROUP_SELECTIONビルトイン

説明

所定のグループ内の選択行を選択解除します。このビルトインは、個々の行にSET_GROUP_SELECTIONを実行することによりプログラム的に選択済みとしてマークされたすべてのレコード・グループ行を選択解除するのに使用します。

構文

```
PROCEDURE RESET_GROUP_SELECTION
  (recordgroup_id  RecordGroup);
PROCEDURE RESET_GROUP_SELECTION
(recordgroup_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

recordgroup_id	Form Builderがグループを作成するときに割り当てる一意のID。そのIDのデータ型はRECORDGROUPです。
recordgroup_name	ユーザーがレコード・グループを作成するときに付けた名前。その名前のデータ型はVARCHAR2です。

RESET_GROUP_SELECTIONの例

```
/*
** Built-in:RESET_GROUP_SELECTION
** Example:If the user presses the (Cancel) button, forget
**          all of the records in the 'USERSEL' record
**          group that we may have previously marked as
**          selected records.
** Trigger:When-Button-Pressed
*/
BEGIN
  Reset_Group_Selection( 'usersel' );
END;
```

RESIZE_WINDOWビルトイン

説明

ウィンドウのサイズを所定の幅および高さに変更します。RESIZE_WINDOWのコールにより、ウィンドウが現在表示されていなくても、そのウィンドウの幅と高さが設定されます。スクリーン上のウィンドウの左上角のxおよびy座標によって指定されているので、RESIZE_WINDOWは、ウィンドウの位置を変更しません。

Microsoft Windowsでは、定数FORMS_MDI_WINDOWをウィンドウ名に指定すれば、MDIアプリケーション・ウィンドウのサイズを変更できます。

また、SET_WINDOW_PROPERTYを使用してもウィンドウのサイズを変更できます。

構文

```
PROCEDURE RESIZE_WINDOW
  (window_id  Window,
   width      NUMBER,
   height     NUMBER);
PROCEDURE RESIZE_WINDOW
  (window_name  VARCHAR2,
   width        NUMBER,
   height       NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

window_id	Form Builderがウィンドウ作成時にそのウィンドウに割り当てられる一意のIDを指定します。FIND_WINDOWのビルトインを使用して、そのIDを適切な型の変数に戻します。このIDのデータ型はWindow型です。
window_name	ウィンドウの作成時に付けた名前を指定します。その名前のデータ型はVARCHAR2です。
width	そのウィンドウの新しい幅をフォーム座標単位で指定します。
height	そのウィンドウの新しい高さをフォーム座標単位で指定します。

RESIZE_WINDOWの例

```
/*
** Built-in:RESIZE_WINDOW
** Example:Set Window2 to be the same size as Window1
```

```
*/
PROCEDURE Make_Same_Size_Win( Window1 VARCHAR2, Window2 VARCHAR2) IS
  wn_id1 Window;
  w      NUMBER;
  h      NUMBER;
BEGIN
  /*
  ** Find Window1 and get it's width and height.
  */
  wn_id1 := Find_Window(Window1);
  w      := Get_Window_Property(wn_id1,WIDTH);
  h      := Get_Window_Property(wn_id1,HEIGHT);
  /*
  ** Resize Window2 to the same size
  */
  Resize_Window( Window2, w, h );
END;
```

RETRIEVE_LISTビルトイン

説明

現リストの内容を取り出し、指定されたレコード・グループに格納します。ターゲット・レコードには、次の2列（VARCHAR2）構造体が必要です。

列1: **列2:**

リスト・ラベル　リスト値

リスト項目の内容を保存しておくと、その内容でリストを回復できるようになります。

構文

```
PROCEDURE RETRIEVE_LIST
  (list_id     ITEM,
   recgrp_name  VARCHAR2);
PROCEDURE RETRIEVE_LIST
  (list_id    ITEM,
   recgrp_id  RecordGroup);
PROCEDURE RETRIEVE_LIST
  (list_name  VARCHAR2,
   recgrp_id  RecordGroup);
PROCEDURE RETRIEVE_LIST
```

```
(list_name    VARCHAR2,
 recgrp_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

list_id	Form Builderがリスト項目を作成するときに割り当てる一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
list_name	ユーザーがリスト項目を作成するときに付けた名前。その名前のデータ型はVARCHAR2です。
recgrp_id	Form Builderによりレコード・グループ作成時に割り当てられる一意のIDを指定します。そのIDのデータ型はRECORDGROUPです。
recgrp_name	ユーザーがレコード・グループ作成時に与えたVARCHAR2名です。

RETRIEVE_LISTの例

POPULATE_LISTの例を参照してください。

RUN_PRODUCTビルトイン

説明

サポートされているOracle Toolsのひとつを起動して、モジュール名または実行対象のモジュール名を指定します。コールされた製品がそのコール時に使用不能である場合、Form Builderではエンド・ユーザーにメッセージが戻されます。

パラメータ・リストを作成し、次にそれをRUN_PRODUCTのコールで参照すると、そのフォームは、コールされた製品に対して、コマンド・ライン・パラメータ、およびバインドまたは字句参照、名前付き問合せの値を示すテキストおよびデータ・パラメータを渡せるようになります。DATA_PARAMETER型のパラメータは、Form Builder内のレコード・グループのポインタ

です。DATA_PARAMETERはReport BuilderおよびGraphics Builderには渡せますが、Form BuilderやOracle Bookには渡せません。

フォーム内からレポートを実行するには、専用のレポート統合ビルトインRUN_REPORT_OBJECTを使用することもできます。

構文

```
PROCEDURE RUN_PRODUCT
  (product         NUMBER,
   module          VARCHAR2,
   commmode        NUMBER,
   execmode        NUMBER,
   location        NUMBER,
   paramlist_id    VARCHAR2,
   display         VARCHAR2);
PROCEDURE RUN_PRODUCT
  (product         NUMBER,
   module          VARCHAR2,
   commmode        NUMBER,
   execmode        NUMBER,
   location        NUMBER,
   paramlist_name  VARCHAR2,
   display         VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

product	起動するOracle製品の数値定数を指定します。FORMSはランフォーム・セッションを指定します。GRAPHICSはGraphics Builderを指定します。REPORTSはReportBuilderを指定します。BOOKはOracle Bookを指定します。
module	そのモジュール、またはコールされた製品により実行されるモジュールのVARCHAR2名を指定します。有効値は、フォーム・モジュールまたはレポート、Graphics Builder表示、Oracle Bookモジュールの名前です。アプリケーションは、そのモジュール、またはコールされた製品に定義されたデフォルト・パス内のモジュールを探します。

commmode	コールされた製品の実行時に使用する通信モードを指定します。このパラメータの有効な数値定数はSYNCHRONOUSおよびASYNCHRONOUSです。 SYNCHRONOUS　コールされた製品が終了した後にのみ制御がForm Builderに戻されることを指定します。また、コールされるアプリケーションが動作している間、エンド・ユーザーはフォーム内での作業を行うことができません。 ASYNCHRONOUS　コールされたアプリケーションがその表示を完了していない場合でも、制御をコール側アプリケーションに即時戻すことを指定します。
execmode	コールされた製品の実行時に実行モードを使用することを指定します。このパラメータの有効な数値定数はBATCHおよびRUNTIMEです。Report BuilderおよびGraphics Builderの実行時、execmodeはBATCHまたはRUNTIMEのいずれかです。Form Builderの実行時には、常にexecmodeをRUNTIMEに設定してください。
location	そのモジュール、またはコールされた製品に実行させたいモジュール（ファイル・システムまたはデータベース）の位置を指定します。このプロパティの有効定数はFILESYSTEMおよびDBです。
Paramlist_name or paramlist_ID	コールされた製品に渡されるパラメータ・リストを指定します。このパラメータの有効値は、パラメータ・リストのVARCHAR2名パラメータ・リストのIDまたはNULL文字列（''）です。パラメータ・リストIDを指定するには、PARAMLISTタイプの変数を使用します。 コールされた製品へのテキスト・パラメータの引渡しは、SYNCHRONOUSおよびASYNCHRONOUSモードのどちらでも可能です。ただし、DATA_PARAMETERタイプのパラメータ（レコード・グループへのポインタ）を含むパラメータ・リストについては、SYNCHRONOUSモードでReport BuilderおよびGraphics Builderに渡せるのみです。（SYNCHRONOUSモードは、Graphics Builderを起動して、フォーム・チャート項目内に表示されるGraphics Builder表示を戻す際に必要です。 注意: *key*を「LOGON」に設定し、*value*を「いいえ」に設定したパラメータを含むパラメータ・リストを渡すと、Graphics Builderのログオンを防止できます。 注意: DATA_PARAMETERを、Report Builder内の子問合せに渡すことはできません。データ渡しがサポートされるのは、マスター問合せのみです。
display	Graphics Builderが生成する図表（円グラフまたは棒グラフ、グラフなど）を含むForm Builderチャート項目のVARCHAR2名を指定します。チャート項目の名前は、*block_name.item_name*の書式で指定する必要があります。（このパラメータは、フォーム中にGraphics Builderチャート項目を使用している場合にのみ必要です。

RUN_PRODUCTの例

```
/*
** Built-in:RUN_PRODUCT
** Example:Call a Report Builder report, passing the
**          data in record group 'EMP_RECS' to substitute
**          for the report's query named 'EMP_QUERY'.
**          Presumes the Emp_Recs record group already
```

```
**          exists and has the same column/data type
**          structure as the report's Emp_Query query.
*/
PROCEDURE Run_Emp_Report IS
  pl_id ParamList;
BEGIN
  /*
  ** Check to see if the 'tmpdata' parameter list exists.
  */
  pl_id := Get_Parameter_List('tmpdata');
  /*
  ** If it does, then delete it before we create it again in
  ** case it contains parameters that are not useful for our
  ** purposes here.
  */
  IF NOT Id_Null(pl_id) THEN
    Destroy_Parameter_List( pl_id );
  END IF;
  /*
  ** Create the 'tmpdata' parameter list afresh.
  */
  pl_id := Create_Parameter_List('tmpdata');
  /*
  ** Add a data parameter to this parameter list that will
  ** establish the relationship between the named query
  ** 'EMP_QUERY' in the report, and the record group named
  ** 'EMP_RECS' in the form.
  */
  Add_Parameter(pl_id,'EMP_QUERY',DATA_PARAMETER,'EMP_RECS');
  /*
  **Pass a Parameter into PARAMFORM so that a parameter dialog will not
appear
  **for the parameters being passing in.
  */
Add_Parameter(pl_id, 'PARAMFORM', TEXT_PARAMETER, 'NO');
/*
  ** Run the report synchronously, passing the parameter list
  */
  Run_Product(REPORTS, 'empreport', SYNCHRONOUS, RUNTIME,
          FILESYSTEM, pl_id, NULL);
END;
```

RUN_REPORT_OBJECTビルトイン

説明

フォーム内からレポートを実行するには、このビルトインを使用します。ローカルまたはリモートのデータベース・サーバーに対してレポートを実行できます。このビルトインを実行することは、レポート上でRUN_PRODUCTビルトインを使用することと似ています。

構文

```
FUNCTION RUN_REPORT_OBJECT
  (report_id REPORT_OBJECT
);
```

ビルトイン・タイプ

制限なしプロシージャ

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

report_id	実行するレポートのIDを指定します。FIND_REPORT_OBJECTビルトインを使用すると、特定レポートのレポートIDを取得できます。

使用上の注意

- ローカルで実行しているまたはリモートのレポート・サーバー上で実行しているレポートを一意に識別するVARCHAR2値を返します。このレポートID文字列を、REPORT_OBJECT_STATUS、COPY_REPORT_OBJECTおよびCANCEL_REPORT_OBJECTへのパラメータとして使用できます。

  空の「レポート・サーバー」プロパティを持つRUN_REPORT_OBJECTを呼び出すと、戻り値はNULLになります。その場合、ビルトインREPORT_OBJECT_STATUSおよびCOPY_REPORT_OBJECT_OUTPUTには実際のID値が必要となるため、これらのビルトインは使用できません。

RUN_REPORT_OBJECTの例

```
DECLARE
   repid REPORT_OBJECT;
   v_rep  VARCHAR2(100);
```

```
   rep_status varchar2(20);
BEGIN
   repid := find_report_object('report4');
   v_rep := RUN_REPORT_OBJECT(repid);
   ......
END;
```

SCROLL_DOWNビルトイン

説明

前に隠されたレコードのうち、より高い順序番号を持つレコードが表示されるよう、現在のブロックのリストをスクロールします。変更可能なレコードが存在し、問合せがブロック内でオープンされている場合は、SCROLL_DOWN処理中にForm Builderによってレコードがフェッチされます。単一行ブロックで、SCROLL_DOWNはレコードのブロックのリスト内の次のレコードを表示します。SCROLL_DOWNは、表示されているレコードのうち順序番号が最も小さいレコード内の現項目のインスタンスに入力フォーカスを移動します。

構文

```
PROCEDURE SCROLL_DOWN;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

SCROLL_DOWNの例

```
/*
** Built-in:SCROLL_DOWN
** Example:Scroll records down some.
*/
BEGIN
Scroll_Down;
END;
```

SCROLL_UPビルトイン

説明

前に隠されたレコードのうち、より低い順序番号を持つレコードが表示されるよう、現在のブロックのリストをスクロールします。このアクセスはブロックの表示よりも"上"にあったレコードを表示します。

SCROLL_UPは、表示されているレコードのうち順序番号が最も大きいレコード内の現項目のインスタンスに入力フォーカスを移動します。

構文

```
PROCEDURE SCROLL_UP;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

SCROLL_UPの例

```
/*
** Built-in:SCROLL_UP
** Example:Scroll records up some.
*/
BEGIN
  Scroll_Up;
END;
```

SCROLL_VIEWビルトイン

説明

「キャンバス上のビューポートのX位置」プロパティと「キャンバス上のビューポートのY位置」プロパティを変更して、ビューをキャンバス上の別の位置に移動します。ビューを移動すると、オペレータにキャンバスの別の領域を見せることができますが、ウィンドウ内のビューの位置は変わりません。

注意: コンテント・キャンバスやツールバー・キャンバスでは、キャンバスが表示されるウィンドウはそのキャンバスのビューを表します。また、スタック・キャンバスでは、「幅」プロパティと「高さ」プロパティを設定して、ビューのサイズを制御します。

構文

```
PROCEDURE SCROLL_VIEW
(view_id  ViewPort,
 x     NUMBER,<HP> y     NUMBER);
PROCEDURE SCROLL_VIEW
  (view_name  VARCHAR2,
   x         NUMBER,
   y         NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

view_id	Form Builderによりオブジェクト作成時にそのビューに割り当てられる一意のIDを指定します。FIND_VIEWビルトインを使用して、そのIDを適切なタイプの変数に戻します。このIDのデータ型はVIEWPORT型です。
view_name	ユーザーがオブジェクト定義時にそのオブジェクトに与えた名前を指定します。その名前のデータ型はVARCHAR2です。
x	キャンバスの左上角からのビューの左上角のX座標を指定します。
y	キャンバスの左上角からのビューの左上角のY座標を指定します。

SCROLL_VIEWの例

```
/*
** Built-in:SCROLL_VIEW
** Example:Scroll the view whose name is passed in 10% to
**           the right or left depending on the 'direction'
**           parameter.
*/
PROCEDURE Scroll_Ten_Percent( viewname VARCHAR2,
                direction VARCHAR2 ) IS
  vw_id       ViewPort;
  vw_wid      NUMBER;
  vw_x        NUMBER;
  cn_id       Canvas;
  cn_wid      NUMBER;
```

```
  ten_percent  NUMBER;
  new_x       NUMBER;
  old_y       NUMBER;
BEGIN
  /*
  ** Get the id's for the View and its corresponding canvas
  */
  vw_id := Find_View( viewname );
  cn_id := Find_Canvas( viewname );

  /*
  ** Determine the view width and corresponding canvas
  ** width.
  */
  vw_wid := Get_View_Property(vw_id,WIDTH);
  cn_wid := Get_Canvas_Property(cn_id,WIDTH);
  /*
  ** Calculate how many units of canvas width are outside of
  ** view, and determine 10% of that.
  */
  ten_percent := 0.10 * (cn_wid - vw_wid);
  /*
  ** Determine at what horizontal position the view
  ** currently is on the corresponding canvas
  */
  vw_x:= Get_View_Property(vw_id,VIEWPORT_X_POS_ON_CANVAS);
  /*
  ** Calculate the new x position of the view on its canvas
  ** to effect the 10% scroll in the proper direction.
  ** Closer than ten percent of the distance to the edge
  ** towards which we are moving, then position the view
  ** against that edge.
  */
  IF direction='LEFT' THEN
    IF vw_x > ten_percent THEN
      new_x := vw_x - ten_percent;
    ELSE
      new_x := 0;
    END IF;
  ELSIF direction='RIGHT' THEN
    IF vw_x < cn_wid - vw_wid - ten_percent THEN
      new_x := vw_x + ten_percent;
    ELSE
```

```
      new_x := cn_wid - vw_wid;
    END IF;
  END IF;
  /*
  ** Scroll the view that much horizontally
  */
  old_y := Get_View_Property(vw_id,VIEWPORT_Y_POS_ON_CANVAS);
  Scroll_View( vw_id, new_x , old_y );
END;
```

SELECT_ALLビルトイン

説明

現項目内のテキストを選択します。テキスト項目の内容全体を切り取る、またはコピーする場合は、CUT_REGIONやCOPY_REGIONをコールする前に、このプロシージャをコールします。

構文

```
PROCEDURE SELECT_ALL;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

可

パラメータ

なし

SELECT_RECORDSビルトイン

説明

On-Selectトリガー内からこのビルトインをコールすると、Form BuilderのデフォルトのSELECT処理が起動されます。Oracle以外のデータ・ソースに対して実行するアプリケーションにおいて、Form Builderのデフォルト・トランザクション処理の代わりにトランザクション・トリガーを実行する場合に、このビルトイン処理をインクルードするのが主な用途です。

構文

```
PROCEDURE SELECT_RECORDS;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

SELECT_RECORDSの制限事項

On-Selectトリガー内でのみ有効です。

SELECT_RECORDSの例

```
/*
** Built-in:SELECT_RECORDS
** Example:Perform Form Builder standard SELECT processing
**          based on a global flag setup at startup by the
**          form, perhaps based on a parameter.
** Trigger:On-Select
*/
BEGIN
  /*
  ** Check the flag variable we setup at form startup
  */
  IF :Global.Using_Transactional_Triggers = 'TRUE' THEN
    User_Exit('my_select block=EMP');
  /*
  ** Otherwise, do the right thing.
  */
  ELSE
    Select_Records;
  END IF;
END;
```

SERVER_ACTIVEビルトイン

説明

所定のOLEコンテナと対応付けられているOLEサーバーが実行中かどうかを示します。

■ OLEサーバーが実行中の場合は、TRUEが返ります。

■ OLEサーバーが実行していない場合は 、FALSEが返ります。

適切な型を持つ変数を定義して返り値を受け取ります。

構文

```
FUNCTION SERVER_ACTIVE
  (item_id  Item);
FUNCTION SERVER_ACTIVE
  (item_name  VARCHAR2);
```

戻り値

BOOLEAN

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

不可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。

SERVER_ACTIVEの制限事項

Microsoft WindowsおよびMacintosh上でのみ有効。

SERVER_ACTIVEの例

```
/*
** Built-in:SERVER_ACTIVE
** Example:Checks to see if the OLE server is active.
** Trigger:When-Button-Pressed
```

```
*/
DECLARE
 item_id  ITEM;
 item_name VARCHAR(25) := 'OLEITM';
 active_serv BOOLEAN;
BEGIN
 item_id := Find_Item(item_name);
 IF Id_Null(item_id) THEN
  message('No such item:'||item_name);
 ELSE
  active_serv := Forms_OLE.Server_Active(item_id);
  IF active_serv = FALSE THEN
   Forms_OLE.Activate_Server(item_id);
  END IF;
 END IF;
END;
```

SET_ALERT_BUTTON_PROPERTYビルトイン

説明

警告内のボタンの1つに割り当てられたラベルを変更します。

構文

```
PROCEDURE SET_ALERT_BUTTON_PROPERTY
  (alert_id    ALERT,
   button      NUMBER,
   property    VARCHAR2,
   value       VARCHAR2);
PROCEDURE SET_ALERT_BUTTON_PROPERTY
  (alert_name  VARCHAR2,
   button      NUMBER,
   property    VARCHAR2,
   value       VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

alert_id	警告作成時にForm Builderによってその警告に割り当てられた一意のID（データ型ALERT）を指定します。FIND_ALERTビルトインを使用すると、このIDを適切な型の変数に返すことができます。
alert_name	警告の名前をVARCHAR2型で指定します。
button	変更する警告ボタンを指定する定数。ALERT_BUTTON1、ALERT_BUTTON2またはALERT_BUTTON3のいずれかです。
property	**LABEL**　警告ボタンのラベル・テキストを指定します。
value	指定したプロパティに適用する値をVARCHAR2型で指定します。

使用上の注意

ラベルがNULLに指定されると、ボタンのラベルは、設計時に指定されたラベルに戻ります。

SET_ALERT_PROPERTYビルトイン

説明

既存の警告のメッセージ・テキストを変更します。

構文

```
SET_ALERT_PROPERTY
  (alert_id    ALERT,
   property    NUMBER,
   message     VARCHAR2);
SET_ALERT_PROPERTY
  (alert_name  VARCHAR2,
   property    NUMBER,
   message     VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

alert_id	警告作成時にForm Builderによってその警告に割り当てられた一意のID（データ型ALERT）を指定します。FIND_ALERTビルトインを使用すると、このIDを適切な型の変数に返すことができます。
alert_name	警告の名前をVARCHAR2型で指定します。

property	設定する「警告」プロパティを特定する次の定数をどれか一つ指定します。 ALERT_MESSAGE_TEXT　警告メッセージのテキストを設定することを指定します。 TITLE　警告のタイトルを指定します。プロパティ値がNULLでなければ、Form Builderで指定された値に上書きします。
message	新しい警告メッセージを指定します。現行の警告メッセージはこのメッセージに置換されます。このメッセージは、引用符で囲まれた文字列として、変数として、あるいは文字列／変数構成体の形で渡されます。

SET_ALERT_PROPERTYの制限事項

メッセージ・ボックスの文字が200文字を超えると、超えた部分は切り捨てられます。

SET_ALERT_PROPERTYの例

```
/*
** Built-in:SET_ALERT_PROPERTY
** Example:Places the error message into a user-defined alert **
named 'My_Error_Alert' and displays the alert.
** Trigger:On-Error
*/
DECLARE
  err_txt   VARCHAR2(80) := Error_Text;
  al_id    Alert;
  al_button Number;
BEGIN
  al_id := Find_Alert('My_Error_Alert');
  Set_Alert_Property(al_id, alert_message_text, err_txt );
  al_button := Show_Alert( al_id );
END;
```

SET_APPLICATION_PROPERTYビルトイン

説明

現行アプリケーションのアプリケーション・プロパティを設定（または再設定）します。

構文

```
SET_APPLICATION_PROPERTY
  (property  NUMBER,
   value     VARCHAR2)
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

property	指定したアプリケーションに設定するプロパティを指定します。設定できるプロパティは次のとおりです。 BUILTIN_DATE_FORMAT ビルトイン日付書式マスクを指定します。 CURSOR_STYLE 与えられたアプリケーションのカーソル・スタイルを指定します。 DATE_FORMAT_COMPATIBILITY_MODE 特定の日付書式変換操作をどのように行うかを指定します。 FLAG_USER_VALUE_TOO_LONG ユーザーの入力した値が項目の最大長プロパティを超過している際の、Form Builderによるその値の処理方法を指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 PLSQL_DATE_FORMAT PL/SQL日付書式マスクを指定します。
value	このプロパティに設定する新しい値。

SET_BLOCK_PROPERTYビルトイン

説明

指定されたブロックに指定されたブロック・プロパティを設定します。

構文

```
SET_BLOCK_PROPERTY
  (block_id   Block,
   property   VARCHAR,
   value      VARCHAR);
SET_BLOCK_PROPERTY
  (block_id   Block,
   property   VARCHAR,
   x    NUMBER);
SET_BLOCK_PROPERTY
  (block_id  Block,
   property VARCHAR,
```

```
x   NUMBER
   y    NUMBER);
SET_BLOCK_PROPERTY
  (block_name VARCHAR2,
   property   VARCHAR,
   value      VARCHAR);
SET_BLOCK_PROPERTY
  (block_name VARCHAR2,
   property   VARCHAR,
   x    NUMBER);
SET_BLOCK_PROPERTY
  (block_name VARCHAR2,
   property   VARCHAR,
   x    NUMBER,
   y    NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

block_id	ブロック作成時にForm Builderによってそのブロックに割り当てられた一意のID。データ型はBLOCKです。
block_name	ブロック作成時にユーザーがそのブロックに付けた名前。データ型はVARCHAR2です。
property	次の定数のいずれかを指定します。 **ALL_RECORDS** 問合せ基準に一致するすべてのレコードを、問合せの実行時にデータ・ブロック内にフェッチするかどうかを指定します。 **BLOCKSCROLLBAR_POSITION** ブロックのスクロール・バーのxおよびy位置を、「座標システム」フォーム・プロパティによって示されるフォーム座標単位で指定します。 **BLOCKSCROLLBAR_X_POS** ブロックのスクロール・バーのx位置を、「座標システム」フォーム・プロパティによって示されるフォーム座標単位で指定します。

property（続き）	BLOCKSCROLLBAR_Y_POS	ブロックのスクロール・バーのy位置を、「座標システム」フォーム・プロパティによって示されるフォーム座標単位で指定します。
	COORDINATION_STATUS	マスター/ディテール・リレーションのディテール・ブロックとしてのブロックが、現在そのマスター・ブロックのすべてと調整が取れているかどうかを示す状態を指定します。すなわち、ディテール・レコードがマスター・ブロックの現マスター・レコードに対応しているかどうかを示す状態を指定します。有効な値は、COORDINATEDまたはNON_COORDINATEDです。
	CURRENT_RECORD_ATTRIBUTE	指定したブロックに関連づけられる名前付き可視属性のVARCHAR2型名を指定します。指定した名前付き可視属性が存在しない場合は、エラー・メッセージが出力されます。
	CURRENT_ROW_BACKGROUND_COLOR	オブジェクトのバックグラウンド領域のカラー。
	CURRENT_ROW_FILL_PATTERN	オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。
	CURRENT_ROW_FONT_NAME	オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。
	CURRENT_ROW_FONT_SIZE	ポイント数で指定されるフォントのサイズ。
	CURRENT_ROW_FONT_SPACING	フォントの幅つまり文字間のスペース（カーニング）。
	CURRENT_ROW_FONT_STYLE	フォントのスタイル。
	CURRENT_ROW_FONT_WEIGHT	フォントの太さ。
	CURRENT_ROW_FOREGROUND_COLOR	オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。
	CURRENT_ROW_WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。

property（続き）	DEFAULT_WHERE	ブロックのデフォルトWHERE句を指定し、既存のWHERE句を上書きします。（注意:データ・ブロックのWHERE句プロパティについて設計時にプロパティ・パレットによって確定された値は上書きされません）。WHERE句は引用符で囲みます。予約語WHEREは指定してもしなくてもどちらでも構いません。デフォルトのWHERE句には、標準のバインド変数構文を使用して、グローバル変数、フォームのパラメータおよび項目の値への参照を含めることができます。
	DELETE_ALLOWED	与えられたブロックのレコードをオペレータまたはアプリケーションが削除してよいかどうかを指定します。有効な値は、PROPERTY_TRUEまたはPROPERTY_FALSEです。
	DML_DATA_TARGET_NAME	ブロックのDMLデータ・ソースの名前を指定します。
	ENFORCE_PRIMARY_KEY	ブロックの挿入または更新されたレコードがデータベースにコミットされるためには、一意特性を持っていなくてはならないことを指定します。有効な値は、PROPERTY_TRUEまたはPROPERTY_FALSEです。
	INSERT_ALLOWED	与えられたブロックにレコードをオペレータまたはアプリケーションが挿入してよいかどうかを指定します。有効な値は、PROPERTY_TRUEまたはPROPERTY_FALSEです。
	KEY_MODE	ブロックのキー・モードを指定します。このプロパティは、Oracle以外のデータ・ソースに対してForm Builderを実行しているときに特に便利です。有効な値は、UPDATEABLE_PRIMARY_KEYとNON_UPDATEABLE_PRIMARY_KEYです。
	LOCKING_MODE	ブロックの「ロックモード」プロパティを指定します。有効な値は、DELAYEDまたはIMMEDIATEです。
	MAX_QUERY_TIME	最大問合せ時間を指定します。オペレータは、問合せの経過時間がこのプロパティを超えたとき問合せを中止できます。
	MAX_RECORDS_FETCHED	フェッチ可能なレコードの最大数を指定します。このプロパティは、「全レコード問合せ」プロパティが「はい」に設定されている場合にのみ有効です。
	NAVIGATION_STYLE	ブロックの「ナビゲーション形式」プロパティを指定します。有効な値は、SAME_RECORD、CHANGE_RECORDまたはCHANGE_BLOCKです。
	NEXT_NAVIGATION_BLOCK	ブロックの次のナビゲーション・ブロックの名前を指定します。デフォルトでは、次のナビゲーション・ブロックは順序番号が次に大きいブロックです。ただし、「次のナビゲーション・データブロック」ブロック・プロパティを設定すると、デフォルトのブロックのナビゲーション順序を上書きできます。
	OPTIMIZER_HINT	問合せ作成時にForm BuilderがRDBMSオプティマイザに渡すヒントを指定します。これにより、ブロック問合せ時にForm designerで最大のパフォーマンスを得ることができます。

property（続き）	ORDER_BY ORDER	ブロックのデフォルトORDER BY句を指定し、既存のORDER BY句に上書きします。引用符で囲みます。ただし実際の単語'ORDER BY'は囲まないでください。"ORDER BY"で与えられた文に、Form Builderが自動的にプレフィックスを付けます。
	PRECOMPUTE_SUMMARIES(現在、定義されていません。)	
	PREVIOUS_NAVIGATION_BLOCK	ブロックの前のナビゲーション・ブロックの名前を指定します。デフォルトでは、前のナビゲーション・ブロックは順序番号が次に小さいブロックです。ただし、「次のナビゲーション・ブロック」ブロック・プロパティを設定すれば、デフォルトのブロックのナビゲーション順序を上書きできます。
	QUERY_ALLOWED	問合せがオペレータまたはプログラムが、ブロックの問合せを発行できるかどうかを指定します。有効な値は、PROPERTY_TRUEまたはPROPERTY_FALSEです。
	QUERY_DATA_SOURCE_NAME	ブロックの問合せデータ・ソースの名前を指定します。注意:ブロックのデータソースがプロシージャの場合、ブロックのQUERY_DATA_SOURCE_NAMEは設定できません。
	QUERY_HITS	COUNT_QUERY操作によって識別されるレコード数を示すNUMBER値を指定します。
	UPDATE_ALLOWED	与えられたブロックのレコードをオペレータまたはアプリケーションが更新してよいかどうかを指定します。有効な値は、PROPERTY_TRUEまたはPROPERTY_FALSEです。
	UPDATE_CHANGED_COLUMNS	オペレータが更新した列のみがデータベースに送信されることを指定します。「変更列のみ更新」プロパティが「いいえ」に設定されていれば、更新されたかどうかに関係なくすべての列が送信されます。この場合、特にブロックにLONGデータ型が含まれていれている場合は、ネットワークの通信量が大幅に増加します。
value	次の定数を前述のプロパティの値に引数として渡すことができます。	
	COORDINATED	マスター/ディテール・リレーションのディテール・ブロックであるブロックの「COORDINATION_STATUS」プロパティをCOORDINATEDに設定することを指定します。
	DELAYED	コミット動作実行時のみForm Builderにディテール・レコードをロックさせることを指定します。
	IMMEDIATE	データベースのレコードが変更されたら、直ちにForm Builderにディテール・レコードをロックさせることを指定します。
	NON_COORDINATED	マスター/ディテール・リレーションのディテール・ブロックとしてのブロックの「COORDINATION_STATUS」プロパティをNON_COORDINATEDに設定することを指定します。

value（続き）	NON_UPDATEABLE_PRIMARY_KEY 基礎を形成するデータ・ソースでは主キーを更新できないという原則に基づいて、Form Builderにブロック内のレコードを処理させることを指定します。 PROPERTY_TRUE プロパティをTRUE状態に設定することを指定します。特に、DELETE_ALLOWED、INSERT_ALLOWED、QUERY_HITSおよびUPDATE_ALLOWEDの値を設定する場合に使用します。 PROPERTY_FALSE プロパティをFALSE状態に設定することを指定します。 UNIQUE_KEY 基礎を形成するデータ・ソースではある形式の一意キーまたはROWIDを使用するという原則に基づいて、Form Builderにブロック内のレコードを処理させることを指定します。 UPDATEABLE_PRIMARY_KEY 基礎を形成するデータ・ソースで主キーの更新を許可するという原則に基づいて、Form Builderにブロック内のレコードを処理させることを指定します。
X	フォーム座標システム単位で指定された軸座標のNUMBER値。xとyの位置を設定する場合、この値はx座標を参照します。y位置のみを設定する場合、この値はy座標を参照します。
Y	フォーム座標システム単位で指定されたy軸座標のNUMBER値。この値はxとyの位置を設定するときに適用され、他のすべてのプロパティでは無視されます。

SET_BLOCK_PROPERTYの例

```
/*
** Built-in:SET_BLOCK_PROPERTY
** Example:Prevent future inserts, updates, and deletes to
**          queried records in the block whose name is
**          passed as an argument to this procedure.
*/
PROCEDURE Make_Block_Query_Only( blk_name IN VARCHAR2 )
IS
  blk_id Block;
BEGIN
/* Lookup the block's internal ID */
  blk_id := Find_Block(blk_name);
  /*
  ** If the block exists (ie the ID is Not NULL) then set
  ** the three properties for this block.Otherwise signal
  ** an error.
  */
  IF NOT Id_Null(blk_id) THEN
    Set_Block_Property(blk_id,INSERT_ALLOWED,PROPERTY_FALSE);
    Set_Block_Property(blk_id,UPDATE_ALLOWED,PROPERTY_FALSE);
    Set_Block_Property(blk_id,DELETE_ALLOWED,PROPERTY_FALSE);
```

```
  ELSE
    Message('Block '||blk_name||' does not exist.');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

BLOCKSCROLLBAR_POSITIONの使用方法:

```
/*
** Built-in:SET_BLOCK_PROPERTY
** Example:** Built-in:  SET_BLOCK_PROPERTY
** Example:   Set the x and y position of the block's scrollbar
**  to the passed x and y coordinates
*/
PROCEDURE Set_Scrollbar_Pos( blk_name IN VARCHAR2, xpos IN
   NUMBER, ypos IN NUMBER )
IS
BEGIN
   Set_Block_Property(blk_name, BLOCKSCROLLBAR_POSITION, xpos, ypos);
END;
```

SET_CANVAS_PROPERTYビルトイン

説明

指定されたキャンバスの指定されたキャンバス・プロパティを設定します。

構文

```
SET_CANVAS_PROPERTY
  (canvas_id    CANVAS,
   property     NUMBER,
   value        VARCHAR2);
SET_CANVAS_PROPERTY
  (canvas_id    CANVAS,
   property     NUMBER,
   x            NUMBER);
SET_CANVAS_PROPERTY
  (canvas_id    CANVAS,
   property     NUMBER,
   x            NUMBER,
   y            NUMBER);
SET_CANVAS_PROPERTY
  (canvas_name  VARCHAR2,
```

```
   property     NUMBER,
   value        VARCHAR2);
SET_CANVAS_PROPERTY
  (canvas_name  VARCHAR2,
   property     NUMBER,
   x            NUMBER);
SET_CANVAS_PROPERTY
  (canvas_name  VARCHAR2,
   property     NUMBER,
   x            NUMBER,
   y            NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

canvas_id	キャンバス・オブジェクト作成時にForm Builderによってそのオブジェクトに割り当てられた一意のID。FIND_CANVASビルトインを使用すると、このIDをCANVAS型の変数に返すことができます。
canvas_name	キャンバス・オブジェクトの定義時に付けた名前。データ型はVARCHAR2です。
property	指定したキャンバスで設定するプロパティ。指定できるプロパティは次のとおりです。 BACKGROUND_COLOR　オブジェクトのバックグラウンド領域のカラー。 CANVAS_SIZE　キャンバスの寸法（幅、高さ）。 FILL_PATTERN　オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME　オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE　ポイント数で指定されるフォントのサイズ。 FONT_SPACING　フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE　フォントのスタイル。 FONT_WEIGHT　フォントの太さ。 FOREGROUND_COLOR　オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 HEIGHT　文字数によるキャンバスの高さ。

property（続き）	TOPMOST_TAB_PAGE	最上部に現れるタブ・ページの名前（タブ・キャンバスの他のすべてのタブ・ページの上に表示します）。
	VISUAL_ATTRIBUTE	現在のフォームに存在する有効な名前付き可視属性、またはForm Builderによってキャンバスへ適用されるランタイム・リソース・ファイルにある論理属性定義の名前。
	WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	WIDTH	キャラクタによるキャンバスの幅。
value	指定したプロパティに設定するVARCHAR2型の値。	
x	指定したプロパティによって、X座標または幅の値をNUMBERで指定します。引数はフォームの座標システムの単位で指定します。	
y	指定したプロパティにより、Y座標または高さのNUMBER値。引数はフォームの座標システムの単位で指定します。	

SET_CANVAS_PROPERTYの制限事項

- 存在しない名前付き可視属性は入力できません。
- Form Builderによって、指定した名前が名前付き可視属性として見つからない場合は、ユーザーのOracle*Terminalリソース・ファイルで表示属性が検索されます。

SET_CANVAS_PROPERTYの例

```
/* Change the "background color" by dynamically setting the
** canvas color at runtime to the name of a visual attribute
** you created:
*/
BEGIN
  SET_CANVAS_PROPERTY('my_cvs', visual_attribute, 'blue_txt');
END;
```

SET_CUSTOM_ITEM_PROPERTYビルトイン

注意: このビルトインは、SET_CUSTOM_PROPERTYビルトインと置き換えられました。新規フォームでは、SET_CUSTOM_PROPERTYビルトインを使用してください。次に記載する情報は、保守のみを目的としています。

説明

Bean領域項目に関連するJavaBeanのプロパティの値を設定します。

構文

このビルトインは、VARCHAR2、NUMBERまたはBOOLEANの各型で使用できます。

```
SET_CUSTOM_ITEM_PROPERTY
  (item,
   prop-name,
   varchar2 value);
SET_CUSTOM_ITEM_PROPERTY
  (item,
   prop-name,
   number value);
SET_CUSTOM_ITEM_PROPERTY
  (item,
   prop-name,
   boolean value);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item	ターゲットJavaBeanに関連付けられたBean領域項目の名前。名前の形式は、VARCHAR2リテラルまたは名前の値に設定される変数です。
prop-name	このBean領域に関連付けられたJavaBeanコンテナの特定のプロパティ。
value	指定されたプロパティの値。値はVARCHAR2、INTEGERまたはBOOLEANのいずれかの型にしてください。

使用上の注意

- JavaBeanコンテナでは、各プロパティのタイプは、ID.registerPropertyを使用して作成されるIDクラスの1つのインスタンスによって表す必要があります。
- フォームで実行される各Set_Custom_Item_Propertyビルトインでは、JavaBeanコンテナのsetPropertyメソッドがコールされます。
- Bean領域項目の名前は、Find_Item('Item_Name')または単に'Item_Name'を介して取得できます。

SET_CUSTOM_PROPERTYビルトイン

説明

プラグイン可能Javaコンポーネント内のユーザー定義プロパティの値を設定します。

構文

このビルトインは、VARCHAR2、NUMBERまたはBOOLEANの各型で使用できます。

```
SET_CUSTOM_PROPERTY
  (item,
   row-number,
   prop-name,
   value   VARCHAR2);
SET_CUSTOM_PROPERTY
  (item,
   row-number,
   prop-name,
   value   NUMBER);
SET_CUSTOM_PROPERTY
  (item,
   row-number,
   prop-name,
   value   BOOLEAN);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item	ターゲットのプラグイン可能Javaコンポーネントに関連付けられた項目の名前またはID。名前の形式は、VARCHAR2リテラルまたは名前の値が設定された変数です。
row-number	設定対象項目のインスタンスの行番号。　（インスタンスの行番号は1から始まります。）すべてのインスタンスを設定する場合は、定数ALL_ROWSを指定します。
prop-name	設定対象Javaコンポーネントのプロパティ名
value	指定されたプロパティの新しい値。値はVARCHAR2、NUMBERまたはBOOLEANのいずれかの型にしてください。

使用上の注意

- プラグイン可能Javaコンポーネントでは、各カスタム・プロパティのタイプは、ID.registerPropertyを使用して作成されるIDクラスの1つのインスタンスによって表す必要があります。
- フォームで実行される各Set_Custom_Propertyビルトインについて、Javaコンポーネントの setPropertyメソッドがコールされます。

■ 項目の名前は、Find_Item('Item_Name')または単に'Item_Name'を介して取得できます。

SET_CUSTOM_PROPERTYビルトインの例

この例では、プラグイン可能JavaコンポーネントはJavaBeanです。ここではコードの一部を示しますが、完全なコードは、完全な例にあります。）

JavaBeanのコンテナ（またはラッパー）では、次のようになります。

```
private static final ID SETRATE = ID.registerProperty("SetAnimationRate");
```

フォームでは、エンド・ユーザー画面の「faster」ボタン上でWhen_Button_Pressedトリガーによってアクティブにされるプログラム単位の一部として、次のようになります。

```
NewAnimationRate := gb.CurAnimationRate + 25
    . . .
Set_Custom_Property('Juggler_Bean', ALL_ROWS, 'SetAnimationRate',
NewAnimationRate);
```

このSet_Custom_Propertyビルトインでは、次の点に注意してください。

■ Juggler_Beanは、フォーム内のBean領域項目の名前です。この項目は、JavaBeanのコンテナと関連付けられています。

■ SetAnimationRateは、JavaBeanのコンテナ内のプロパティです。

■NewAnimationRateは、JavaBeanコンテナに渡されるプロパティの新しい値を保持する変数です。

SET_FORM_PROPERTYビルトイン

説明

指定されたフォームのプロパティを設定します。

構文

```
SET_FORM_PROPERTY
  (formmodule_id  FormModule,
   property      NUMBER,
   value         NUMBER);
SET_FORM_PROPERTY
  (formmodule_name  VARCHAR2,
   property        NUMBER,
   value           NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

formmodule_id	フォーム作成時に FormBuilderによってそのフォームに割り当てられた一意のIDを指定します。このIDのデータ型はFormModule型です。
formmodule_name	ユーザーがフォーム・モジュールを作成したときにそのモジュールに付けた名前を指定します。その名前のデータ型はVARCHAR2です。
property	フォームに設定するプロパティを指定します。 CURRENT_RECORD_ATTRIBUTE 与えられたフォームに関連付けられる名前付き可視属性のVARCHAR2型の名前を指定します。指定した名前付き可視属性が存在しない場合は、エラー・メッセージが出力されます。 CURRENT_ROW_BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 CURRENT_ROW_FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 CURRENT_ROW_FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 CURRENT_ROW_FONT_SIZE ポイント数で指定されるフォントのサイズ。 CURRENT_ROW_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 CURRENT_ROW_FONT_STYLE フォントのスタイル。 CURRENT_ROW_FONT_WEIGHT フォントの太さ。 CURRENT_ROW_FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 CURRENT_ROW_WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。

property（続き）	CURSOR_MODE	Form Builderによって定義を試みる場合のカーソル状態を指定します。このプロパティは、主に、Oracle以外のデータ・ソースに接続するときに使用します。有効な値は、OPEN_AT_COMMITおよびCLOSE_AT_COMMITです。
	DEFER_REQUIRED_ENFORCEMENT	必須項目の妥当性検査を項目レベルからレコード・レベルへ延期するかを指定します。有効な値は、PROPERTY_TRUE、PROPERTY 4 5およびPROPERTY_FALSEです。
	DIRECTION	双方向オブジェクトのレイアウト方向を指定します。有効な値は、DIRECTION_DEFAULT、RIGHT_TO_LEFTおよびLEFT_TO_RIGHTです。
	FIRST_NAVIGATION_BLOCK	Form Builderがフォームの起動時にナビゲートするブロックの名前を復帰します。デフォルトでは、オブジェクト・ナビゲータで定義した最初のブロックが最初のナビゲーション・ブロックになりますが、「最初のナビゲーション・データ・ブロック」ブロック・プロパティを設定すれば、フォーム起動時に別のブロックを最初のナビゲーション・ブロックに指定することができます。
	SAVEPOINT_MODE	Form Builderがセーブポイントを発行するかどうかを指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。
	VALIDATION	Form Builderがデフォルトの妥当性検査を行うかどうかについて指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。
	VALIDATION_UNIT	フォームの妥当性検査の有効範囲を指定します。有効な値は、DEFAULT_SCOPE、FORM_SCOPE、BLOCK_SCOPE、RECORD_SCOPEおよびITEM_SCOPEです。
value	次の定数を前述のプロパティの値に引数として渡すことができます。	
	BLOCK_SCOPE	Form Builderによってブロック・レベルでデータの妥当性を検査するときに指定します。これは、たとえば、ナビゲーション・イベントがそのブロックから別のブロックに移動したために妥当性検査が強制されたときに、Form Builderによってブロック内のすべてレコードの妥当性が検査されることを意味します。
	CLOSE_AT_COMMIT	Oracle以外のデータベースに対してフォームが実行されているときのように、データベース・コミット中にカーソルを閉じておきたい場合に指定します。
	DEFAULT_SCOPE	「有効単位」フォーム・モジュール・プロパティをデフォルト値に設定します。GUIウィンドウ・マネージャ上では、デフォルトの妥当性検査単位は項目です。

value（続き）	FORM_SCOPE	フォーム・レベルのみで妥当性検査を行うときに指定します。
	ITEM_SCOPE	項目レベルで妥当性を検査するときに指定します。これは、たとえば、ナビゲーション・イベントの結果として、ある項目から別の項目にナビゲートしたときに、Form Builderによって変更された各項目の妥当性が検査されることを意味します。
	OPEN_AT_COMMIT	データベース・コミット中にカーソルを開いておく場合に指定します。これは、Oracleに対して実行しているときの標準の設定です。
	PROPERTY_TRUE	プロパティをTRUE状態に設定することを指定します。
	PROPERTY_FALSE	プロパティをFALSE状態に設定することを指定します。
	RECORD_SCOPE	Form Builderによってレコード・レベルで妥当性を検査するときに指定します。これは、たとえば、あるレコードから別のレコードに移動したときに、FormBuilderによって変更された各レコードの妥当性が検査されることを意味します。

SET_FORM_PROPERTYの例

例1

```
/*
** Built-in:SET_FORM_PROPERTY
** Example:Set the Cursor Mode property in the current form
**  to CLOSE_AT_COMMIT and changes the form
**  Validation unit to the Block level.
*/
DECLARE
   fm_id FormModule;
BEGIN
   fm_id := Find_Form(:System.Current_Form);
   Set_Form_Property(fm_id,CURSOR_MODE,CLOSE_AT_COMMIT);
   Set_Form_Property(fm_id,VALIDATION_UNIT,BLOCK_SCOPE);
END;
```

例2

```
/*
** Built-in:SET_FORM_PROPERTY
** Example:Setup form and block properties required to run
**          against a particular non-Oracle datasource.
**          Procedure accepts the appropriate numerical
**          constants like DELAYED as arguments.
```

```
**
** Usage:Setup_Non_Oracle(PROPERTY_FALSE,
**                        CLOSE_AT_COMMIT,
**                        UPDATEABLE_PRIMARY_KEY,
**                        DELAYED);
*/
PROCEDURE Setup_Non_Oracle( the_savepoint_mode NUMBER,
                 the_cursor_mode    NUMBER,
                 the_key_mode      NUMBER,
                 the_locking_mode   NUMBER ) IS
  fm_id   FormModule;
  bk_id   Block;
  bk_name VARCHAR2(40);
BEGIN
  /* ** Validate the settings of the parameters ** */
  IF the_savepoint_mode NOT IN (PROPERTY_TRUE,PROPERTY_FALSE) THEN
    Message('Invalid setting for Savepoint Mode.');
    RAISE Form_Trigger_Failure;
  END IF;
  IF the_cursor_mode NOT IN (CLOSE_AT_COMMIT,OPEN_AT_COMMIT) THEN
    Message('Invalid setting for Cursor Mode.');
    RAISE Form_Trigger_Failure;
  END IF;
  IF the_key_mode NOT IN (UNIQUE_KEY,UPDATEABLE_PRIMARY_KEY,
            NON_UPDATEABLE_PRIMARY_KEY) THEN
    Message('Invalid setting for Key Mode.');
    RAISE Form_Trigger_Failure;
  END IF;
  IF the_locking_mode NOT IN (IMMEDIATE,DELAYED) THEN
    Message('Invalid setting for Locking Mode.');
    RAISE Form_Trigger_Failure;
  END IF;
  /*
  ** Get the id of the current form
  */
  fm_id := Find_Form(:System.Current_Form);
  /*
  ** Set the two form-level properties
  */
  Set_Form_Property(fm_id, SAVEPOINT_MODE, the_savepoint_mode);
  Set_Form_Property(fm_id, CURSOR_MODE, the_cursor_mode);
  /*
  ** Set the block properties for each block in the form
```

```
  */
  bk_name := Get_Form_Property(fm_id,FIRST_BLOCK);
  WHILE bk_name IS NOT NULL LOOP
    bk_id := Find_Block(bk_name);

    Set_Block_Property(bk_id,LOCKING_MODE,the_locking_mode);

    Set_Block_Property(bk_id,KEY_MODE,the_key_mode);
    IF the_key_mode IN (UPDATEABLE_PRIMARY_KEY,
             NON_UPDATEABLE_PRIMARY_KEY) THEN
      Set_Block_Property(bk_id,PRIMARY_KEY,PROPERTY_TRUE);
    END IF;

    bk_name := Get_Block_Property(bk_id, NEXTBLOCK);
  END LOOP;
END;
```

SET_GROUP_CHAR_CELLビルトイン

説明

指定された行および列から特定されるレコード・グループのセルの値を設定します。

構文

```
SET_GROUP_CHAR_CELL
  (groupcolumn_id    GroupColumn,
   row_number        NUMBER,
   cell_value        VARCHAR2);
SET_GROUP_CHAR_CELL
  (groupcolumn_name  VARCHAR2,
   row_number        NUMBER,
   cell_value        VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

groupcolumn_id	レコード・グループの列を作成したときにForm Builderによってその列に割り当てられた一意のID。適切な型の変数にIDを戻すには、FIND_COLUMNビルトインを使用します。このIDのデータ型はGROUPCOLUMN型です。
groupcolumn_name	ユーザーが列を作成したときにその列に付けた名前で、recordgroup_name.groupcolumn_nameのように、レコード・グループ名およびドットが前に付けられます。その名前のデータ型はVARCHAR2です。
row_number	値を設定するセルを含む行番号を指定します。行番号はNUMBER型の整数で指定します。
cell_value	VARCHAR2型の列では、セルに入力するVARCHAR2型の値を、LONG列では、セルに入力するLONG値を指定します。

SET_GROUP_CHAR_CELLの制限事項

- 指定した行のセルの値を設定する前に、その行を作成する必要があります。このビルトインで行を指示しても、Form Builderは新しい行を自動的には作成しません。ADD_GROUP_ROWビルトインを使用して明示的に行を追加するか、POPULATE_GROUPまたはPOPULATE_GROUP_WITH_QUERYのいずれかを使用してグループを挿入します。
- このビルトインは、静的レコード・グループには有効ではありません。静的レコード・グループは設計時に作成されたレコード・グループで、「レコード・グループ・タイプ」プロパティが「スタティック」に設定されています。

SET_GROUP_CHAR_CELLの例

ADD_GROUP_ROWの例を参照してください。

SET_GROUP_DATE_CELLビルトイン

説明

指定された行および列から特定されるレコード・グループのセルの値を設定します。

構文

```
SET_GROUP_DATE_CELL
  (groupcolumn_id    GroupColumn,
   row_number        NUMBER,
   cell_value        DATE);
SET_GROUP_DATE_CELL
  (groupcolumn_name  VARCHAR2,
   row_number        NUMBER,
   cell_value        DATE);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

groupcolumn_id	レコード・グループの列を作成したときにForm Builderによってその列に割り当てられた一意のID。適切な型の変数にIDを戻すには、FIND_COLUMNビルトインを使用します。このIDのデータ型はGroupColumn型です。
groupcolumn_name	ユーザーが列を作成したときにその列に付けた名前で、recordgroup_name.groupcolumn_nameのように、レコード・グループ名およびドットが前に付けられます。その名前のデータ型はVARCHAR2です。
row_number	値を設定するセルを含む行番号を指定します。行番号はNUMBER型の整数で指定します。
cell_value	セルに入力するDATE値を指定します。

SET_GROUP_DATE_CELLの制限事項

- 指定した行のセルの値を設定する前に、その行を作成する必要があります。このビルトインで新しい行を指定しても、Form Builderは新しい行を自動的には作成しません。ADD_GROUP_ROWビルトインを使用して行を明示的に追加するか、POPULATE_GROUPまたはPOPULATE_GROUP_WITH_QUERYを使用してグループを挿入してください。
- このビルトインは、静的レコード・グループには有効ではありません。静的レコード・グループは設計時に作成されたレコード・グループで、「レコード・グループ・タイプ」プロパティが「スタティック」に設定されています。

SET_GROUP_DATE_CELLの例

```
/*
** Built-in:SET_GROUP_DATE_CELL
** Example:Lookup a row in a record group, and set the
**          minimum order date associated with that row in
**          the record group.Uses the 'is_value_in_list'
**          function from the GET_GROUP_CHAR_CELL example.
*/
PROCEDURE Set_Max_Order_Date_Of( part_no  VARCHAR2,
                  new_date DATE ) IS
  fnd_row NUMBER;
BEGIN
  /*
  ** Try to lookup the part number among the temporary part list
  ** record group named 'TMPPART' in its 'PARTNO' column.
```

```
  */
  fnd_row := Is_Value_In_List( part_no, 'TMPPART', 'PARTNO');

  IF fnd_row = 0 THEN
    Message('Part Number '||part_no||' not found.');
    RETURN;
  ELSE
    /*
    ** Set the corresponding Date cell value from the
    ** matching row.
    */
    Set_Group_Date_Cell('TMPPART.MAXORDDATE',fnd_row,new_date );
  END IF;
END;
```

SET_GROUP_NUMBER_CELLビルトイン

説明

指定された行および列から特定されるレコード・グループのセルの値を設定します。

構文

```
SET_GROUP_NUMBER_CELL
  (groupcolumn_id    GroupColumn,
   row_number        NUMBER,
   cell_value        NUMBER);
SET_GROUP_NUMBER_CELL
  (groupcolumn_name  VARCHAR2,
   row_number        NUMBER,
   cell_value        NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

groupcolumn_id	レコード・グループの列を作成したときにForm Builderによってその列に割り当てられた一意のID。適切な型の変数にIDを戻すには、FIND_COLUMNビルトインを使用します。このIDのデータ型はGROUPCOLUMN型です。

groupcolumn_name	ユーザーが列を作成したときにその列に付けた名前で、recordgroup_name.groupcolumn_nameのように、レコード・グループ名およびドットが前に付けられます。その名前のデータ型はVARCHAR2です。
row_number	値を設定するセルを含む行番号を指定します。行番号はNUMBER型の整数で指定します。
cell_value	セルに入力するNUMBER値を指定します。

SET_GROUP_NUMBER_CELLの制限事項

- 指定した行のセルの値を設定する前に、その行を作成する必要があります。ADD_GROUP_ROWビルトインを使用して明示的に行を追加するか、POPULATE_GROUPまたはPOPULATE_GROUP_WITH_QUERYを使用してグループを挿入してください。
- このビルトインは、静的レコード・グループには有効ではありません。静的レコード・グループは設計時に作成されたレコード・グループで、「レコード・グループ・タイプ」プロパティが「スタティック」に設定されています。

SET_GROUP_NUMBER_CELLの例

ADD_GROUP_ROWの例を参照してください。

SET_GROUP_SELECTIONビルトイン

説明

後続のプログラムによる行操作に備えて、指定されたレコード・グループの指定された行にマークを付けます。行番号が1から順に付けられます。たとえば、行3、8および12を選択すると、Form Builderではこれらの行が選択1、2および3とみなされます。RESET_GROUP_SELECTIONビルトインをコールすることによりグループ全体の行選択を取り消せます。

構文

```
SET_GROUP_SELECTION
  (recordgroup_id     RecordGroup,
   row_number         NUMBER);
SET_GROUP_SELECTION
  (recordgroup_name   VARCHAR2,
   row_number         NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

recordgroup_id	Form Builderによりレコード・グループ作成時にそのコード・グループに割り当てられる一意のIDを指定します。FIND_GROUPビルトインを使用して、そのIDを変数に戻します。そのIDのデータ型はRECORDGROUPです。
recordgroup_name	ユーザーがレコード・グループ作成時に与えたレコード・グループ名を指定します。その名前のデータ型はVARCHAR2です。
row_number	ユーザーが選択するレコード・グループ行の番号を指定します。ユーザーが指定する値はNUMBERです。

SET_GROUP_SELECTIONの例

```
/*
** Built-in:SET_GROUP_SELECTION
** Example:Set all of the even rows as selected in the
**          record group whose id is passed-in as a
**          parameter.
*/
PROCEDURE Select_Even_Rows ( rg_id RecordGroup ) IS
BEGIN
  FOR j IN 1..Get_Group_Row_Count(rg_id) LOOP
    IF MOD(j,2)=0 THEN
      Set_Group_Selection( rg_id, j );
    END IF;
  END LOOP;
END;
```

SET_INPUT_FOCUSビルトイン

説明

現フォームのメニュー上に入力フォーカスを設定します。トリガー処理が終了すると、そのメニューが自動的にアクティブになります。

構文

```
SET_INPUT_FOCUS
  (MENU  );
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

MENU

SET_INPUT_FOCUSの制限事項

このビルトインは、文字モードおよびブロック・モード環境でのみ使用できます。

SET_INPUT_FOCUSの例

```
/*
** Built-in:SET_INPUT_FOCUS
** Example:Directs the users input focus to the Menu when
**          used with the only support parameter, MENU.
**          Only has an effect on character-mode or
**          block-mode devices.
*/
BEGIN
  Set_Input_Focus(MENU);
END;
```

SET_ITEM_INSTANCE_PROPERTYビルトイン

説明

指定された項目プロパティを変更することにより、ブロック内の現行の項目インスタンスを変更します。SET_ITEM_INSTANCE_PROPERTYでは、現行インスタンスをミラー化する項目の外観は変更されません。

現行フォーム内の項目はどれでも参照できます。SET_ITEM_INSTANCE_PROPERTYは、項目の現行インスタンスの表示方法のみに影響することに注意してください。指定された項目の他のインスタンスには影響がありません。つまり、マルチレコード・ブロック内にある項目について表示方法の変更を指定した場合、SET_ITEM_INSTANCE_PROPERTYは、ブロックの現行レコードに属する項目のインスタンスのみを変更します。マルチレコード・ブロック内の項目のすべてのインスタンスを変更する場合は、SET_ITEM_PROPERTYを使用します。

SET_ITEM_INSTANCE_PROPERTYによって行われた変更は、次の動作が発生するまで有効です。

■ 同じ項目インスタンスが別のSET_ITEM_INSTANCE_PROPERTYによって参照される場合。

■ 同じ項目インスタンスがDISPLAY_ITEMビルトインによって参照される場合。

■ 項目のインスタンスが削除される場合(CLEAR_RECORDまたは問合せなどによって）。

■ 現行フォームを終了した場合。

構文

```
SET_ITEM_INSTANCE_PROPERTY
  (item_id   ITEM,
   record_number NUMBER,
   property      NUMBER,
   value         VARCHAR2);
SET_ITEM_INSTANCE_PROPERTY
  (item_name     VARCHAR2,
   record_number NUMBER,
   property      NUMBER,
   value         VARCHAR2);
SET_ITEM_INSTANCE_PROPERTY
  (item_name     VARCHAR2,
   record_number NUMBER,
   property      NUMBER,
   value         NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_id	オブジェクト作成時にFormBuilder によってそのオブジェクトに割り当てられた一意のID。FIND_ITEMビルトインを使用すると、このIDをITEM型の変数に返すことができます。
record_number	設定するレコード番号。レコード番号はブロックにおけるレコードの位置です。行番号はNUMBER型で指定します。ブロックの現在のレコードを設定する場合は、CURRENT_RECORDを指定します。
item_name	ユーザーが項目を作成した時にその項目に付けた名前。データ型はVARCHAR2です。
property	指定した項目で設定するプロパティ。指定できるプロパティは次のとおりです。

property（続き）	BORDER_BEVEL	指定された項目インスタンスの項目境界の凹凸を指定します。有効値は、RAISED、LOWERED、PLAIN （凹凸なし）または「" "」で表します。「" "」の場合の値は設計時に項目レベルで指定される値によって、または実行時にSET_ITEM_PROPERTYによって境界線の凹凸が決定されます。 注意: Form Builder内で項目の「凹凸」プロパティが「なし」に設定されている場合、BORDER_BEVELは設定できません。
	INSERT_ALLOWED	データベースから取り出されないレコードのみに対して適用されます。項目インスタンス、項目およびブロック・レベルでPROPERTY_TRUEに設定すると、エンド・ユーザーは項目インスタンスを変更できます。項目インスタンス、項目およびブロック・レベルでPROPERTY_FALSEに設定すると、エンド・ユーザーは項目インスタンスを変更できません。
	NAVIGABLE	項目インスタンスおよび項目のレベルでPROPERTY_TRUEに設定すると、エンド・ユーザーはデフォルトのキーボード・ナビゲーションを使用して項目インスタンスにナビゲートできます。項目インスタンスまたは項目のレベルでPROPERTY_FALSEに設定すると、エンド・ユーザーはデフォルトのキーボード・ナビゲーションを使用して項目インスタンスにナビゲートできません。
	REQUIRED	エンド・ユーザーにその項目インスタンスについて非NULL値を強制的に入力させたい場合、定数PROPERTY_TRUEを指定します。項目インスタンスおよび項目のレベルでPROPERTY_FALSEに設定すると、項目インスタンスが必須でないことを示します。
	UPDATE_ALLOWED	データベースから取り出されるレコードのみに適用されます。項目インスタンス、項目およびブロック・レベルでPROPERTY_TRUEに設定すると、エンド・ユーザーは項目インスタンスを変更できます。インスタンス、項目およびブロック・レベルでPROPERTY_FALSEに設定すると、エンド・ユーザーは項目インスタンスを変更できません。
	VISUAL_ATTRIBUTE	現在のフォームに存在する有効な名前付き可視属性または「''」を指定します。「''」を指定すると、可視属性は項目インスタンスのレベルで未指定のままになります。

使用上の注意

複数のレベル（項目インスタンス、項目およびブロック）で指定されたプロパティを操作する場合は、次のガイドラインを考慮してください。

- 複数のレベルで指定された必須プロパティは、ORで結合されます。
- 複数のレベルで指定されたブール型プロパティは、ANDで結合されます。

項目インスタンス、項目およびブロックの各レベルで指定されたプロパティを組み合せて導出される値は有効値と呼ばれます。前述の2つのルールによる効果の一部を次に示します。

- INSERT_ALLOWEDをTRUEに設定しても、ブロック・レベルおよび項目レベルで同様に設定されていなければ、項目インスタンス・レベルで有効になりません。たとえば、「挿入可」プロパティがインスタンス・レベルで「はい」であり、項目レベルおよびブロック・レベルで「いいえ」の場合、ユーザーは項目インスタンスにデータを入力できません。

- NAVIGABLEをTRUEに設定しても、項目レベルおよび項目インスタンス・レベルで同様に設定されていなければ、項目インスタンス・レベルで有効になりません。

- NAVIGABLEをTRUEに設定すると、ブロックが入力可能であるかどうかに影響する場合があります。ブロックの現行レコードに、その有効値がNAVIGABLEプロパティについてTRUEである項目インスタンスが含まれる場合に限り、ブロックの読込み専用「入力可能」プロパティはTRUEになります。

- REQUIREDをFALSEに設定しても、項目レベルおよび項目インスタンス・レベルで同様に設定されていなければ、項目インスタンス・レベルで有効になりません。

- UPDATE_ALLOWEDをTRUEに設定しても、ブロック・レベル、項目レベルおよび項目インスタンス・レベルで同様に設定されていなければ、項目インスタンス・レベルで有効になりません。

- 項目インスタンス・レベルでBORDER_BEVELを設定すると、項目インスタンスのBORDER_BEVELプロパティが未指定の場合（つまり「" "」に設定した場合）を除き、項目レベルのBORDER_BEVELプロパティに上書きします。

- 項目インスタンス・レベルでVISUAL_ATTRIBUTEを設定すると、部分可視属性を指定しない限り、項目およびブロック・レベルのプロパティに上書きします。この場合、部分可視属性と項目の現行可視属性の間でマージが発生します。項目インスタンス・レベルでVISUAL_ATTRIBUTEに「" "」を設定すると、このプロパティの項目レベルの設定が使用されます。

- 新規レコードが作成されると、その「項目インスタンス」プロパティは、上位レベルで指定された値を上書きしない値に設定されます。たとえば、BORDER_BEVELプロパティとVISUAL_ATTRIBUTEプロパティは「" "」に、REQUIREDはFALSEに、他のブール型プロパティはTRUEに設定されます。

- 項目インスタンスのプロパティを設定しても、指定された項目をミラー化する項目の項目インスタンスのプロパティには影響しません。

- ポップリストのインスタンスは、選択されると、現行の値がNULLの場合、または「必須」プロパティがFALSEに設定されている場合は、さらにNULL値を表示します。テキスト・リスト（Tlist）のインスタンスの現行の値を選択すると、「必須」プロパティがFALSEに設定されている場合は、その値の選択が解除されます（Tlistの値は1つも選択されません）。「必須」プロパティがTRUEに設定されている場合、Tlistインスタンスの現在の値を選択しても効力はあ

りません。つまり、値は選択されたままです。

SET_ITEM_INSTANCE_PROPERTYの例

```
/*
** Built-in:SET_ITEM_INSTANCE_PROPERTY
** Example:Change the visual attribute of each item instance in the
**          current record
*/
DECLARE
  cur_itm   VARCHAR2(80);
  cur_block VARCHAR2(80) := :System.Cursor_Block;
BEGIN
  cur_itm   := Get_Block_Property( cur_block, FIRST_ITEM );
  WHILE ( cur_itm IS NOT NULL ) LOOP
    cur_itm := cur_block||'.'||cur_itm;
    Set_Item_Instance_Property( cur_itm, CURRENT_RECORD,
       VISUAL_ATTRIBUTE,'My_Favorite_Named_Attribute');
    cur_itm := Get_Item_Property( cur_itm, NEXTITEM );
  END LOOP;
END;
```

SET_ITEM_PROPERTYビルトイン

説明

指定された項目のプロパティを変更することによって、ブロック内の項目のインスタンスをすべて変更します。場合によっては、特定のオブジェクト・プロパティの取得はできますが、設定できないことがありますので注意してください。

構文

```
SET_ITEM_PROPERTY
  (item_id     ITEM,
   property    NUMBER,
   value       VARCHAR2);
SET_ITEM_PROPERTY
  (item_name   VARCHAR2,
   property    NUMBER,
   value       VARCHAR2);
SET_ITEM_PROPERTY
  (item_id     ITEM,
   property    NUMBER,
   x           NUMBER);
```

```
SET_ITEM_PROPERTY
  (item_name   VARCHAR2,
   property    NUMBER,
   x           NUMBER);
SET_ITEM_PROPERTY
  (item_id     ITEM,
   property    NUMBER,
   x           NUMBER,
   y           NUMBER);
SET_ITEM_PROPERTY
  (item_name   VARCHAR2,
   property    NUMBER,
   x           NUMBER,
   y           NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_id	オブジェクト作成時にFormBuilder によってそのオブジェクトに割り当てられた一意のID。FIND_ITEMビルトインを使用すると、このIDをITEM型の変数に返すことができます。
item_name	ユーザーが項目を作成した時にその項目に付けた名前。データ型はVARCHAR2です。
property	指定した項目で設定するプロパティ。指定できるプロパティは次のとおりです。 **ALIGNMENT** テキストの整列(テキストおよび表示項目のみ)。有効な値は、ALIGNMENT_START、ALIGNMENT_END、ALIGNMENT_LEFT、ALIGNMENT_ CENTER、ALIGNMENT_RIGHTです。 **AUTO_HINT** 入力フォーカスが指定された項目にある場合、Form Builderによってステータス行にヘルプ・ヒントを表示するかどうかを決定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 **AUTO_SKIP** エンド・ユーザーがテキスト項目に最後の文字を入力する場合、カーソルが次の項目へ自動スキップするかどうかを指定します。このプロパティを指定できるのはテキスト項目のみです。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。

property（続き）	BACKGROUND_COLOR	オブジェクトのバックグラウンド領域のカラー。
	BORDER_BEVEL	指定された項目インスタンスの項目境界の凹凸を指定します。有効な値は、RAISED、LOWEREDまたはPLAIN（凹凸なし）です。 注意:Form Builder内で項目の「凹凸」プロパティが「なし」に設定されている場合、BORDER_BEVELは設定できません。
	CASE_INSENSITIVE_QUERY	項目に入力される問合せ条件が大文字小文字を区別するかどうかを指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。
	CASE_RESTRICTION	指示されたテキスト項目に入力されるすべてのテキストに対して大文字小文字の制限を適用するかどうかを指定します。有効な値は、UPPERCASE、LOWERCASEまたはNONEです。
	COMPRESS	サウンド・オブジェクトのサウンド・データを、Form Builderがデータをファイルに書き込む前に圧縮するかどうかを指定します。有効な値はCOMPRESSION_ON、COMPRESSION_OFFおよびORIGINAL_SETTING（デフォルトのデータ圧縮設定を保持）です。
	CONCEAL_DATA	エンド・ユーザーが値を入力したとき、項目をスペースのままにしておくか、またはあいまいなままにしておきたい場合は、定数PROPERTY_TRUEを指定します。テキスト項目に入力された値をすべて画面に表示する場合は、定数PROPERTY_FALSEを指定します。
	CURRENT_RECORD_ATTRIBUTE	与えられた項目に関連付けられる名前付き可視属性のVARCHAR2名を指定します。指定した名前付き可視属性が存在しない場合は、エラー・メッセージが出力されます。
	CURRENT_ROW_BACKGROUND_COLOR	オブジェクトのバックグラウンド領域のカラー。
	CURRENT_ROW_FILL_PATTERN	オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。
	CURRENT_ROW_FONT_NAME	オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。

property（続き）	CURRENT_ROW_FONT_SIZE ポイント数の100倍で指定されるフォントのサイズ(例えば、フォント･サイズが8ポイントの場合には、設定値は800となります)。 CURRENT_ROW_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 CURRENT_ROW_FONT_STYLE フォントのスタイル。 CURRENT_ROW_FONT_WEIGHT フォントの太さ。 CURRENT_ROW_FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 CURRENT_ROW_WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 DIRECTION　双方向オブジェクトのレイアウト方向を指定します。有効な値は、DIRECTION_DEFAULT、RIGHT_TO_LEFTおよびLEFT_TO_RIGHTです。 DISPLAYED　項目が表示（変更可能に）されるか、非表示（使用不可）にされるかを指定します。 ECHO　エンド・ユーザーがテキスト項目にタイプする文字を可視にすべきかどうかを指定します。「Echo」プロパティが「FALSE」に設定されていれば、入力された文字は画面に表示されません。これは、パスワードを保護する場合に使用されます。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 ENABLED　エンド・ユーザーが項目を操作できるようにすべきかどうか指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 注意: 「Enabled」プロパティを「FALSE」に設定すると、他のプロパティの設定値も変更されます。詳細は「プロパティ変更値の波及」のセクションを参照してください。 FILL_PATTERN　オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。

property（続き）	FIXED_LENGTH	項目の値を、項目の「最大長」プロパティの設定に照らし合わせて検査すべきであるかどうかを指定します。FIXED_LENGTHがTRUEであれば、項目の値の文字数が「最大長」の設定値と等しい場合のみ、その項目は有効とみなされます。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。
	FONT_NAME	オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。
	FONT_SIZE	ポイント数で指定されるフォントのサイズ。
	FONT_SPACING	フォントの幅つまり文字間のスペース（カーニング）。
	FONT_STYLE	フォントのスタイル。
	FONT_WEIGHT	フォントの太さ。
	FOREGROUND_COLOR	オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。
	FORMAT_MASK	テキスト項目のデータの受入れ可能な表示形式と入力を指定します。
	HEIGHT	項目の高さを指定します。
	HINT_TEXT	実行時にメッセージ行に表示される項目特有のヘルプ・テキストを指定します。テキストがNULLの場合、Form Builderに指定された元のヒント・テキストが復元されます。
	ICON_NAME	「アイコン化」プロパティが「はい」に設定されたボタン項目に関連付けられたアイコン・リソースのファイル名を指定します。
	IMAGE_DEPTH	イメージ項目に適用されるカラーの深さを指定します。
	INSERT_ALLOWED	新しいレコードで、PROPERTY_TRUEに設定されている場合、通常はエンド・ユーザーが項目を挿入することを許可します。PROPERTY_FALSEに設定すると、オペレータは項目を変更できませんが、項目は通常通り表示されます（グレー表示されません）。（Insert_Allowedを変更しても「Enabled」プロパティには影響しません。）
	ITEM_IS_VALID	現項目を有効であるとみなすべきであるかどうかを指定します。有効な値は、PROPERTY_TRUEまたはPROPERTY_FALSEです。
	ITEM_SIZE	項目の幅と高さをカンマで区切った2つの数として指定します。x,yを含む構文を使用します。

property（続き）	KEEP_POSITION	「カーソル位置を保持」プロパティを「TRUE」または「FALSE」のどちらにするかを指定します。「カーソル位置を保存」が「TRUE」ならば、カーソルは、テキスト項目を抜け出たときの位置に戻ります。「カーソル位置を保持」が「FALSE」であれば、テキスト項目内のデフォルトの位置にカーソルを配置します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。
	LABEL	項目のラベルとして表示するVARCHAR2文字列を指定します。このプロパティは、ボタンのようにラベルをもつ項目にのみ有効です。
	LOCK_RECORD_ON_CHANGE	この項目が変更されたときレコードをロックする場合は、定数PROPERTY_TRUEを指定します。この項目が変更されたときにレコードをロックしない場合は、定数PROPERTY_FALSEを指定します。この定数は、主に、低レベルのロッキング機能を持たないOracle以外のデータ・ソースに接続するときに使用します。
	LOV_NAME	与えられた項目に関係付けられる値リストのVARCHAR2名を指定します。指定した値リスト名が存在しない場合は、エラー・メッセージが出力されます。
	MERGE_CURRENT_ROW_VA	指定された可視属性の内容を現行行の可視属性とマージします（置換ではない）。
	MERGE_TOOLTIP_ATTRIBUTE	指定された可視属性の内容をツールチップの現行の可視属性とマージします（置換ではない）。
	MERGE_VISUAL_ATTRIBUTE	指定された可視属性の内容をオブジェクトの現行の可視属性とマージします（置換ではない）。
	MOUSE_NAVIGATE	エンド・ユーザーがマウスで項目をアクティブ化したとき、Form Builderがナビゲートしてその項目にフォーカスを設定するかどうかを指定します。エンド・ユーザーがマウスを使用して項目にナビゲートできるようにする場合は、定数PROPERTY_TRUEを指定します。エンド・ユーザーがマウスをクリックしても入力フォーカスを現項目から移動したくない場合は、定数PROPERTY_FALSEを指定します。
	NAVIGABLE	エンド・ユーザーがデフォルトのキーボード・ナビゲーションを使用して項目までナビゲートできるようにする場合、定数PROPERTY_TRUEを指定します。項目へのデフォルトのキーボード・ナビゲーションを使用禁止に設定する場合は、定数PROPERTY_FALSEを指定します。（「Navigable」プロパティは「Enabled」プロパティには影響しません。）

property（続き）	NEXT_NAVIGATION_ITEM この現項目に関して、"次のナビゲーション項目"として定義されている項目の名前を指定します。 POPUPMENU_CONTENT_ITEM OLEポップアップ・メニュー項目のプロパティのいれについても設定値を指定します。 POPUPMENU_COPY_ITEM POPUPMENU_CUT_ITEM POPUPMENU_DELOBJ_ITEM POPUPMENU_INSOBJ_ITEM POPUPMENU_LINKS_ITEM POPUPMENU_OBJECT_ITEM POPUPMENU_PASTE_ITEM POPUPEMNU_PASTESPEC_ITEM OLEポップアップ・メニュー項目をOLEポップアップ・メニューに表示しない場合は、文字列HIDDENを指定します。OLEポップアップ・メニュー項目を表示し、変更可能にする場合は、文字列ENABLEDを指定します。OLEポップアップ・メニュー項目を表示し、使用禁止にする場合は、文字列DISABLEDを指定します。 POSITION　項目のX、Y座標を、カンマによって区切られたNUMBERとして指定します。x,yを含む構文を使用します。 PREVIOUS_NAVIGATION_ITEM この現項目に関して、"前のナビゲーション項目"として定義されている項目の名前を指定します。 PRIMARY_KEY　ブロックに挿入または更新されたレコードをデータベースにコミットするためには、それに一意特性を持たせなくてはならないことを示すために、定数PROPERTY_TRUEを指定します。そうでなければ、定数PROPERTY_FALSEを指定します。 PROMPT_ALIGNMENT_OFFSET 項目とプロンプトの間の距離を決めます。 PROMPT_BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 PROMPT_DISPLAY_STYLE 「プロンプト表示スタイル」プロパティがPROMPT_FIRST_RECORD、PROMPT_HIDDENまたはPROMPT_ALL_RECORDSのいずれであるかを決めます。 PROMPT_EDGE　接続されるエッジ（プロンプトの連結枠）がSTART_EDGE、END_EDGE、TOP_EDGEまたはBOTTOM_EDGEのいずれであるかを決定します。

property（続き）	PROMPT_EDGE_ALIGNMENT プロンプトを整列するエッジが、ALIGNMENT_START、ALIGNMENT_ENDまたはALIGNMENT_CENTERのいずれのであるかを指定します。 PROMPT_EDGE_OFFSET 項目とプロンプトの間の距離をVARCHAR2値として決めます。 PROMPT_FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 PROMPT_FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 PROMPT_FONT_SIZE ポイント数で指定されるフォントのサイズ。 PROMPT_FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 PROMPT_FONT_STYLE フォントのスタイル。 PROMPT_FONT_WEIGHT フォントの太さ。 PROMPT_FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 PROMPT_TEXT 項目について表示されるテキスト・ラベルを決定します。 PROMPT_TEXT_ALIGNMENT プロンプトの揃え方が、ALIGNMENT_START、ALIGNMENT_LEFT、ALIGNMENT_RIGHT、ALIGNMENT_CENTERまたはALIGNMENT_ENDのいずれであるかを指定します。 PROMPT_VISUAL_ATTRIBUTE 実行時にプロンプトに適用されるユーザー命名可視属性を指定します。 PROMPT_WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 QUERYABLE エンド・ユーザーが項目に対して問合せを開始することができるようにする場合、定数PROPERTY_TRUEを指定します。問合せで項目が使用できないようにする場合は、定数PROPERTY_FALSEを指定します。

property（続き）	QUERY_ONLY	問合せ対象の項目を指定します。このとき、項目が挿入または更新文の一部にならないようにします。QUERY_ONLYは、テキスト項目、ラジオ・グループおよびチェックボックスに適用できます。項目の完全修飾名を一重引用符で囲んで指定します。
	REQUIRED	エンド・ユーザーに項目の値を強制的に入力させたければ、定数PROPERTY_TRUEを指定します。項目が必須でない場合は、定数PROPERTY_FALSEを指定します。
	SHOW_FAST_FORWARD_BUTTON	サウンド項目上に  を表示するならば定数PROPERTY_TRUEを、隠すならばPROPERTY_FALSEを指定します。
	SHOW_PLAY_BUTTON	サウンド項目上に  を表示するならば定数PROPERTY_TRUEを、隠すならばPROPERTY_FALSEを指定します。なお、Form Builderで隠せるのは  か  のどちらかのみで、両方隠すことはできません。
	SHOW_RECORD_BUTTON	サウンド項目上に  を表示するならば定数PROPERTY_TRUEを、隠すならばPROPERTY_FALSEを指定します。なお、Form Builderで隠せるのは  か  のどちらかのみで、両方隠すことはできません。
	SHOW_REWIND_BUTTON	サウンド項目上に  を表示するならば定数PROPERTY_TRUEを、隠すならばPROPERTY_FALSEを指定します。
	SHOW_SLIDER	サウンド項目上に  を表示するならば定数PROPERTY_TRUEを、隠すならばPROPERTY_FALSEを指定します。
	SHOW_TIME_INDICATOR	サウンド項目上に  ボタンを表示するならば定数PROPERTY_TRUEを、隠すならばPROPERTY_FALSE を指定します。
	SHOW_VOLUME_CONTROL	サウンド項目上に  を表示するならば定数PROPERTY_TRUEを、隠すならばPROPERTY_FALSEを指定します。
	TOOLTIP_BACKGROUND_COLOR	オブジェクトのバックグラウンド領域のカラー。
	TOOLTIP_FILL_PATTERN	オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。

property（続き）	TOOLTIP_FONT_NAME	オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。
	TOOLTIP_FONT_SIZE	ポイント数で指定されるフォントのサイズ。
	TOOLTIP_FONT_SPACING	フォントの幅つまり文字間のスペース（カーニング）。
	TOOLTIP_FONT_STYLE	フォントのスタイル。
	TOOLTIP_FONT_WEIGHT	フォントの太さ。
	TOOLTIP_FOREGROUND_COLOR	オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。
	TOOLTIP_TEXT	項目のツールチップ・テキストを決めます。
	TOOLTIP_WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	UPDATE_ALLOWED	エンド・ユーザーが項目を更新できるようにする場合は、定数PROPERTY_TRUEを指定します。項目が更新できないようにする場合は、定数PROPERTY_FALSEを指定します。
	UPDATE_COLUMN	この列を更新ありとして扱い、データベースに書き込むために列に組み込む必要がある場合は、定数PROPERTY_TRUEを指定します。この列を更新なしとして扱い、データベースに書き込まれる列に組み込まない場合は、定数PROPERTY_FALSEを指定します。
	UPDAOYNUUPDATE_NULL	項目の値がNULLのときのみエンド・ユーザーがその項目を更新できるようにする場合、定数PROPERTY_TRUEを指定します。エンド・ユーザーが値がNULLかどうかに関係なく項目の値を更新できるようにする場合は、定数PROPERTY_FALSEを指定します。
	UPDATE_PERMISSION	Oracle以外のデータ・ソースに対して実行しているときはUPDATE_ALLOWEDを使用します。項目の「UPDATEABLE」および「UPDATE_NULL」プロパティをオンにするには、定数PROPERTY_TRUEを指定します。項目の「UPDATEABLE」および「UPDATE_NULL」プロパティをオフにするには、PROPERTY_FALSEを指定します。
	VALIDATE_FROM_LIST	PROPERTY_TRUEに設定されている場合、Form Builderは、接続した値リストにある値と照らし合わせてテキスト項目の値を妥当性検査すべきであることを指定します。妥当性検査で値リストを使用しない場合は、定数PROPERTY_FALSEを指定します。

property（続き）	VISIBLE	指示された項目は可視か、あるいは隠すべきかを指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 注意: 「可視」プロパティをFALSEに設定すると、他の項目プロパティの設定値も変更されます。詳細は「プロパティ変更値の波及」の項を参照してください。
	VISUAL_ATTRIBUTE	現在のフォームに存在する有効な名前付き可視属性を指定します。 注意:イメージ項目には可視属性を設定できません。
	WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。
	WIDTH	項目の幅をNUMBERとして指定します。幅の単位は、そのフォームに指定した「座標システム」プロパティの値とデフォルトのフォント・スケーリングによって異なります。
	X_POS	X座標をNUMBERで指定します。
	Y_POS	Y座標をNUMBERで指定します。
value	所定のプロパティに適用される値を指定します。プロパティのデータ型によって、入力する値のデータ型が決まります。たとえば、「Visible」プロパティをTRUEに設定する場合は、このプロパティの値に定数PROPERTY_TRUEを指定します。項目のLABELを変更する場合は、値を、つまり、ラベルをVARCHAR2の文字列で指定します。	
	PROPERTY_TRUE	プロパティをTRUE状態に設定することを指定します。
	PROPERTY_FALSE	プロパティをFALSE状態に設定することを指定します。
	プロパティの値を設計時に確定した元の値にリセットする場合は、2つの一重引用符（''）を間にスペースを入れずに入力します。たとえば SET_ITEM_PROPERTY('DEPTNO', FORMAT_MASK, '');と入力すると、書式マスクは設計時の値にリセットされます。	
x	指定したプロパティの種類に応じて、x座標つまり幅を表すNUMBER値を指定します。引数はフォームの座標システムの単位で指定します。	
y	指定したプロパティの種類に応じて、y座標つまり高さを表すNUMBER値を指定します。引数はフォームの座標システムの単位で指定します。	

使用上の注意

次の2点を考慮して、プロパティの値を項目に適用する方法を決めてください。

■ プロパティの変更値の妥当性検査

■ プロパティの変更値の波及

プロパティの変更値の妥当性検査

SET_ITEM_PROPERTYビルトインを使用して値を変更する場合、プロパティの値が変更される前にForm Builderによって新しい値の妥当性が検査されます。その値が有効であれば、Form Builderによって新しい値に変更され、次にSET_ITEM_PROPERTYで同じプロパティが変更されるまで、または、現フォームが終了するまでその値が有効になります。

無効な設定値

新しい値が無効であれば、Form Builderによってエラー・メッセージが表示されます。ターゲット項目が次の条件を満たしている場合は、SET_ITEM_PROPERTYで次の項目のプロパティをTRUEにもFALSEにも設定できません。

このパラメータを設定することはできません...	制限付き設定値	ターゲット項目の条件
（すべて）	TRUE/FALSE	NULLキャンバス項目（項目のキャンバス・プロパティがNULLです）。
ENABLED	TRUE/FALSE TRUE	現項目。 「可視」項目プロパティが「いいえ」です。
INSERT_ALLOWED	TRUE TRUE	「変更可能」項目プロパティが「いいえ」です。 「可視」項目プロパティが「いいえ」です。
NAVIGABLE	TRUE/FALSE TRUE	現項目。 「可視」項目プロパティが「いいえ」です。
QUERYABLE (Query Allowed)	TRUE	「可視」項目プロパティが「いいえ」です。
UPDATE_ALLOWED	TRUE TRUE	「変更可能」項目プロパティが「いいえ」です。 「データ隠蔽」項目プロパティが「はい」です。
UPDATE_NULL (Update if NULL)	TRUE TRUE	「変更可能」項目プロパティが「いいえ」です。 「データ隠蔽」項目プロパティが「はい」です。
VISIBLE	TRUE/FALSE	現項目。

Form Builderでは、プロパティの設定値を変更する前に、項目の現行の内容が検査されるわけではありません。SET_ITEM_PROPERTYを使用して、項目内のデータの妥当性検査方法を左右するような項目プロパティ（FIXED_LENGTHやREQUIREDなど）を変更すると、妥当性検査の結

果を変更前の値に適用できなくなります。新しい妥当性検査ルールは、通常の環境でForm Builderによって次にその項目の妥当性が検査されるまで、その項目に適用されません。

たとえば、アプリケーションに従業員IDなどの入力必須テキスト項目があるとします。このアプリケーションでは、エンド・ユーザーが（REQUIRED項目で許可されない）この項目からそのまま抜け出せるようにする必要があるため、一時的に「REQUIRED」プロパティを「False」に設定します。この時点では、Form Builderによって既存のNULL値にVALIDのマークが付けられています。この後、アプリケーション内で、もう一度「REQUIRED」プロパティをTRUEに設定すると、Form BuilderではVALID/INVALIDマークが自動的には変更されません。NULL値にINVALIDのマークを付けるには（REQUIRED項目を除く）、FormBuilderでその項目の妥当性が検査されるように、項目内で次のように変更する必要があります。

```
IF :block.item IS NULL
THEN :block.item := NULL;
```

プロパティの変更値の波及

SET_ITEM_PROPERTYビルトインを使用して1度に変更できる項目プロパティは、1つのみです。しかし、目的の変更を完了する、あるいは波及させるために他のプロパティも変更する必要がある場合は、SET_ITEM_PROPERTY文1つで複数の項目プロパティが変更されることがあります。この機能は、本来、旧バージョンとの互換性を確保する目的で用意されたものです。

次の表に、Form Builderによって項目プロパティ間で変更を波及させるSET_ITEM_PROPERTY設定を示します。

次のpropertyパラメータを2列目に示す設定にすると、3列目に示す変更が波及される	設定	波及される変更
ENABLED	いいえ	「Navigable」項目プロパティを「FALSE」に設定します。 「Update_Null」項目プロパティを「FALSE」に設定します。 「Updateable」項目プロパティを「FALSE」に設定します。 「Required」項目プロパティを「FALSE」に設定します。
DISPLAYED	いいえ	「Updateable」および「Navigable」の各項目プロパティを「FALSE」に設定します。 「Updateble」項目プロパティを「FALSE」に設定する。 「Update_Null」項目プロパティを「FALSE」に設定します。 「Required」項目プロパティを「FALSE」に設定します。 「Queryable」項目プロパティを「FALSE」に設定します。
UPDATEABLE	True	「Update_Null」項目プロパティを「FALSE」に設定します。
UPDATE_NULL	True	「Updateable」項目プロパティを「FALSE」に設定します。

SET_ITEM_PROPERTYの例

```
/*
** Built-in:SET_ITEM_PROPERTY
** Example:Change the icon of an iconic button dynamically
**          at runtime by changing its icon_name.The user
**          clicks on this button to go into enter query
**          mode, then clicks on it again (after the icon
**          changed) to execute the query.After the query
**          is executed the user sees the original icon
**          again.
** Trigger:When-Button-Pressed
*/
DECLARE
  it_id Item;
BEGIN
  it_id := Find_Item('CONTROL.QUERY_BUTTON');
  IF :System.Mode = 'ENTER-QUERY' THEN
    /*
    ** Change the icon back to the enter query icon, and
    ** execute the query.
    */
    Set_Item_Property(it_id,ICON_NAME,'entquery');
    Execute_Query;
  ELSE
    /*
** Change the icon to the execute query icon and get
    ** into enter query mode.
    */
    Set_Item_Property(it_id,ICON_NAME,'exequery');
    Enter_Query;
  END IF;
END;
```

SET_LOV_COLUMN_PROPERTYビルトイン

説明

指定された値リストの指定された「値リスト」プロパティを設定します。

構文

```
SET_LOV_COLUMN_PROPERTY
  (lov_id    LOV,
   colnum    NUMBER,
   property  NUMBER,
   value     VARCHAR2);
SET_LOV_COLUMN_PROPERTY
  (lov_name  VARCHAR2,
   colnum    NUMBER,
   property  NUMBER,
   value     VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

lov_id	値リスト作成時にForm Builderによってその値リストに割り当てられた一意のIDを指定します。FIND_LOVビルトインを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型は値リスト型です。
lov_name	値リスト型の名前をVARCHAR2型で指定します。
colnum	変更する列をNUMBER型で指定します。最初の列は列1です。
property	指定した値リストに設定するプロパティを指定します。設定できるプロパティは次のとおりです。 TITLE　値リスト列の上に表示するタイトルを制御する「列タイトル」プロパティを設定します。 注意: 列のタイトルをNULLに設定すると、列タイトルが設計時に指定されたタイトルに再設定されます。 WIDTH　列値を表示するための、値リストに予約される幅を指定します。 注意: 列の幅をNULLに設定すると、その列は非表示列になります。
value	指定したプロパティに設定する値をVARCHAR2またはNUMBERで指定します。

SET_LOV_PROPERTYビルトイン

説明

指定された値リストの指定された「値リスト」プロパティを設定します。

構文

```
SET_LOV_PROPERTY
  (lov_id     LOV,
   property   NUMBER,
   value      NUMBER);
SET_LOV_PROPERTY
  (lov_name   VARCHAR2,
   property   NUMBER,
   value      NUMBER);
SET_LOV_PROPERTY
  (lov_id     LOV,
   property   NUMBER,
   x          NUMBER,
   y          NUMBER);
SET_LOV_PROPERTY
  (lov_name   VARCHAR2,
   property   NUMBER,
   x          NUMBER,
   y          NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

lov_id	値リスト作成時にForm Builderによってその値リストに割り当てられた一意のIDを指定します。FIND_LOVビルトインを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型は値リスト型です。
lov_name	値リスト型の名前をVARCHAR2型で指定します。
property	指定した値リストに設定するプロパティを指定します。設定できるプロパティは次のとおりです。 AUTO_REFRESH 値リストが起動されるたびにForm Builderが問合せを再実行するかどうかを指定します。 GROUP_NAME 値リストが関連付けられるレコード・グループを指定します。 LOV_SIZE 値リストのサイズを示す幅と高さの組み合わせを指定します。 POSITION 値リストの位置を示すxとyの組み合わせを指定します。 TITLE 値リストのタイトルを指定します。プロパティ値がNULLでなければ、Form Builderで指定されている値を上書きします。

value	次の定数のいずれかを指定します。 PROPERTY_TRUE　プロパティをTRUE状態に設定することを指定します。 PROPERTY_FALSE　プロパティをFALSE状態に設定することを指定します。
Recordgroup Name	設定するレコード・グループの名前をVARCHAR2で指定します。このレコード・グループは、SET_LOV_PROPERTYがコールされるときにレコード・グループが存在する限り、Form Builderで、またはプログラミングで作成できます。
x	指定したプロパティにより、X座標または幅を指定します。
y	指定したプロパティにより、Y座標または高さを指定します。

SET_LOV_PROPERTYの制限事項

■ 各ビルトイン・コールで設定できるプロパティは1つのみです。

SET_LOV_PROPERTYの例

```
/*
** Built-in:SET_LOV_PROPERTY
** Example:if LOV is currently base on GROUP1,
**          make LOV use GROUP2
*/
DECLARE
  lov_id     LOV;
BEGIN
  lov_id    := Find_LOV('My_LOV_1');
  IF Get_LOV_Property(lov_id,GROUP_NAME) = 'GROUP1' THEN
    Set_LOV_Property(lov_id,GROUP_NAME,'GROUP2');
  ENDIF;
END;
```

SET_MENU_ITEM_PROPERTYビルトイン

説明

メニュー項目の指定されたプロパティを変更します。

構文

```
SET_MENU_ITEM_PROPERTY
  (menuitem_id              MenuItem,
   property                 NUMBER,
   value                    NUMBER);
SET_MENU_ITEM_PROPERTY
  (menu_name.menuitem_name  VARCHAR2,
```

```
property                    NUMBER,
value                       NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

menuitem_id	メニュー項目作成時にForm Builderによってその項目に割り当てられた一意のIDを指定します。FIND_MENU_ITEMを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型はMenuItem型です。
menu_name.menuitem_name	ユーザーがメニュー項目を定義したときにその項目に付けた名前をVARCHAR2で指定します。名前でメニュー項目を指定する場合は、上の構文のように、メニュー項目を修飾するメニュー名も合わせて指定してください。
property	メニュー項目に関する情報を設定する次の定数をどれか1つ指定します。 CHECKED　チェックボックス・メニュー項目またはラジオ・メニュー項目がチェック・マーク付きであるかどうかを示す、「チェック・マーク付き」プロパティを指定します。 ENABLED　メニュー項目が変更可能（アクティブ）か、使用禁止（グレーになっていてオペレータは使用できない）かどうかを指定します。 ICON_NAME　メニューのアイコンプロパティがTRUEに設定されたメニュー項目に関連するアイコン・リソースのファイル名を指定します。 LABEL　メニュー項目ラベルの文字列を指定します。 VISIBLE　メニュー項目が可視的に表示されるかどうかを指定します。
value	次の定数のいずれかを指定します。 PROPERTY_TRUE　プロパティをTRUE状態に設定することを指定します。 PROPERTY_FALSE　プロパティをFALSE状態に設定することを指定します。
Label	VARCHAR2型のラベル名を指定します。

SET_MENU_ITEM_PROPERTYの制限事項

これらの制限は、メニュー・モジュールの「セキュリティ使用」プロパティが「はい」に設定されている場合のみ適用されます。

- メニュー・モジュールの「セキュリティ使用」プロパティが「はい」の場合は、SET_MENU_ITEM_PROPERTYを使用してメニュー項目のプロパティを設定できるかどうかは、フォームのオペレータにその項目へのアクセス権限が付与されているかによって決まります。
- メニュー項目が画面に表示されておらず、かつ、オペレータにメニュー項目へのセキュリティ

のアクセス権限が付与されていない場合、Runformはその項目を表示しません。項目が現在隠されている場合は、SET_MENU_ITEM_PROPERTYを使用してメニュー項目のプロパティを設定することはできません。

- メニュー項目が表示されるが、使用禁止であり、このメニュー項目の「権限なしで表示」プロパティがForm Builderで設定されていれば、Runformはその項目を使用禁止状態で表示します。この場合は、プログラミングによりメニュー項目プロパティを設定できます。

SET_MENU_ITEM_PROPERTYの例

GET_MENU_ITEM_PROPERTYの例を参照してください。

SET_OLEビルトイン

説明

OLEプロパティの値を変更します。

このプロシージャには、NUMBER、VARCHARおよびOLEVARという新しい値の型用として、3種類のバージョンが用意されています。

構文

```
PROCEDURE SET_OLE
  (obj OLEOBJ, memberid PLS_INTEGER
   newval NUMBER, vtype VT_TYPE);

PROCEDURE SET_OLE
  (obj OLEOBJ, memberid PLS_INTEGER
   newval VARCHAR2, vtype VT_TYPE);

PROCEDURE SET_OLE
  (obj OLEOBJ, memberid PLS_INTEGER
   newval OLEVAR, vtype VT_TYPE);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

obj	OLEオブジェクトへのポインタ。
memberid	OLEプロパティのメンバーID。
newval	OLEプロパティに置き換る、指定された型の新しい値。

Vtype	元のバリアントのVT_TYPE。 これはオプションのパラメータです。指定されなかった場合、プロシージャのNUMBERバージョンのデフォルト値はVT_R8です。VARCHAR2バージョンに対しては、デフォルト値はVT_BSTRです。OLEVARバージョンの場合、デフォルトはVT_VARIANTです。つまり、バリアント自体によって実際に指定される型です。

使用上の注意

INIT_OLEARGSコールとADD_OLEARGコールがこのSET_OLEコールよりも先に発生し、その間にGET_OLE、SET_OLEまたはCALL_OLEの各コールが発生しなかった場合、このコールでは、INIT_OLEARGSコールとADD_OLEARGコールで指定された引数を使用してプロパティにアクセスします。

SET_PARAMETER_ATTRビルトイン

説明

指定されたパラメータ・リストの指定されたパラメータのタイプと値を設定します。

構文

```
SET_PARAMETER_ATTR
  (list        PARAMLIST,
   key         VARCHAR2,
   paramtype   NUMBER,
   value       VARCHAR2);
SET_PARAMETER_ATTR
  (name        VARCHAR2,
   key         VARCHAR2,
   paramtype   NUMBER,
   value       VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

*list*または*name*	パラメータ・リストを指定します。実際のパラメータには、PARAMLIST型のパラメータ・リストIDまたはパラメータ・リストのVARCHAR2名のどちらかを指定できます。
key	パラメータの名前をVARCHAR2で指定します。

paramtype	渡すパラメータのタイプを指定します。 DATA_PARAMETER パラメータの値がレコード・グループの名前であることを示します。 TEXT_PARAMETER パラメータの値が実際のデータ値であることを示します。
value	指定したパラメータの値をデータ型VARCHAR2型で指定します。

SET_RADIO_BUTTON_PROPERTYビルトイン

説明

ラジオ・ボタンは*item_name*または*item_id*で指定されたもので、与えられたラジオ・グループの一部であり、与えられたプロパティを設定します。

構文

```
SET_RADIO_BUTTON_PROPERTY
  (item_id      VARCHAR2,
   button_name  VARCHAR2,
   property     NUMBER,
   value        NUMBER);
SET_RADIO_BUTTON_PROPERTY
  (item_id      VARCHAR2,
   button_name  VARCHAR2,
   property     NUMBER,
   x            NUMBER,
   y            NUMBER);
SET_RADIO_BUTTON_PROPERTY
  (item_name    VARCHAR2,
   button_name  VARCHAR2,
   property     NUMBER,
   x            NUMBER,
   y            NUMBER);
SET_RADIO_BUTTON_PROPERTY
  (item_name    VARCHAR2,
   button_name  VARCHAR2,
   property     NUMBER,
   value        NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_id	ラジオ・グループ項目のIDを指定します。オブジェクト作成時に、Form Builderによってそのオブジェクトに一意のIDが割り当てられます。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。
item_name	ラジオ・グループの名前を指定します。ラジオ・グループは、従属するラジオ・ボタンの所有者または親です。その名前のデータ型はVARCHAR2です。
button_name	プロパティを設定する場合は、ラジオ・ボタンの名前を指定します。その名前のデータ型はVARCHAR2です。
property	設定するプロパティを指定します。設定できるプロパティの定数は次のとおりです。 BACKGROUND_COLOR　オブジェクトのバックグラウンド領域のカラー。 ENABLED　ラジオ・ボタンを変更可能にする場合は、定数PROPERTY_TRUEを指定します。オペレータに対してラジオ・ボタンを使用禁止にする場合は、定数PROPERTY_FALSEを指定します。 FILL_PATTERN　オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME　オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE　ポイント数で指定されるフォントのサイズ。 FONT_SPACING　フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE　フォントのスタイル。 FONT_WEIGHT　フォントの太さ。 FOREGROUND_COLOR　オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 HEIGHT　ラジオ・ボタンの高さを指定します。NUMBERで指定します。 ITEM_SIZE　ラジオ・ボタンの幅と高さを設定します。幅と高さはX、Y座標の組み合わせで、NUMBERで指定します。 LABEL　そのラジオ・ボタンの実際の文字列ラベルを指定します。 POSITION　ラジオ・ボタンの位置を指定します。幅と高さはX、Y座標の組み合わせで、NUMBERで指定します。

property（続き）	PROMPT　オブジェクトに表示されるテキスト。 PROMPT_BACKGROUND_COLOR　オブジェクトのバックグラウンド領域のカラー。 PROMPT_FILL_PATTERN　オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 PROMPT_FONT_NAME　オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 PROMPT_FONT_SIZE　ポイント数で指定されるフォントのサイズ。 PROMPT_FONT_SPACING　フォントの幅つまり文字間のスペース（カーニング）。 PROMPT_FONT_STYLE　フォントのスタイル。 PROMPT_FONT_WEIGHT　フォントの太さ。 PROMPT_FOREGROUND_COLOR　オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 PROMPT_WHITE_ON_BLACK　白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 VISIBLE　ラジオ・ボタンを表示する場合は、定数PROPERTY_TRUEを指定します。ラジオ・ボタンを隠す場合は、定数PROPERTY_FALSEを指定します。 VISUAL_ATTRIBUTE　現在のフォームに存在する有効な名前付き可視属性、またはForm Builderによってラジオ・ボタンに適用するランタイム・リソース・ファイルにある論理属性定義の名前を指定します。 WHITE_ON_BLACK　白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 WIDTH　ラジオ・ボタンの幅を指定します。NUMBERで指定します。 X_POS　ラジオ・ボタンのX座標を指定します。NUMBERで指定します。 Y_POS　ラジオ・ボタンのY座標を指定します。NUMBERで指定します。
value	値をNUMBERまたはVARCHAR2で指定します。指定したプロパティのデータ型により入力する値のデータ型が決まります。VARCHAR2の値を入力する場合は、テキスト項目や変数を参照する場合を除き、値を引用符で囲む必要があります。 PROPERTY_TRUE　プロパティをTRUE状態に設定することを指定します。 PROPERTY_FALSE　プロパティをFALSE状態に設定することを指定します。

x	「項目サイズ」プロパティと「位置」プロパティの最初の数値を指定します。
y	「項目サイズ」プロパティと「位置」プロパティの2番目の数値を指定します。

SET_RADIO_BUTTON_PROPERTYの例

```
/*
** Built-in:SET_RADIO_BUTTON_PROPERTY
** Example:Set a particular radio button to disabled.
*/
BEGIN
  Set_Radio_Button_Property('MYBLOCK.FLIGHT_STATUS',
                      'GROUNDED',ENABLED,PROPERTY_FALSE);
END;
```

SET_RECORD_PROPERTYビルトイン

説明

指定されたレコード・プロパティを指定された値に設定します。

構文

```
SET_RECORD_PROPERTY
  (record_number NUMBER,
   block_name    VARCHAR2,
   property      NUMBER,
   value         NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

record_number	ステータスを設定するレコードの番号を指定します。レコード番号はブロックにおけるレコードの位置です。行番号はNUMBER型で指定します。
block_name	ターゲット・レコードを含むブロックの名前を指定します。その名前のデータ型はVARCHAR2です。
property	次のプロパティを指定します。 STATUS　レコードのステータスを変更することを指定します。STATUSは定数です。

value	次の値をどれか1つ指定します。 CHANGED_STATUS　レコードに更新マークを付け、次にコミット動作が発生したときにそのレコードが更新として処理されるように指定します。 INSERT_STATUS　レコードにINSERTマークを付け、次にコミット動作が発生した時にそのレコードが適切な表に挿入されるように指定します。 NEW_STATUS　レコードを新規レコードとして処理する、つまり、挿入、更新、問合せのマークが付いていないレコードであることを指定します。変更されたが消去またはコミットされていないレコードには、NEW_STATUSを割り当てることはできません。 QUERY_STATUS　レコードが実際に問合せレコードであるかどうかに関係なく、レコードを問合せレコードとして処理することを指定します。CREATE_QUERIED_RECORDビルトインも参照してください。

SET_RECORD_PROPERTYの制限事項

レコードの有効な移行状態は、次表のとおりです。

現行のステータス	ターゲット状態			
	NEW	QUERY	INSERT	CHANGED
NEW	可	可[1]	可[2]	不可
QUERY	可[4]	可	不可	可
INSERT	可[4]	可[3]	可	不可
CHANGED	可[4]	不可	不可	可

1　脚注2、3の規則に従います。

2　問合せモードではQUERYおよびINSERTは無効なため、この移行は問合せモードでは実行できません。

3　ランフォームが「一意キー」モードで実行されていて、かつトランザクション制御トリガーが1つも存在しないときに、この移行を実行する場合は、ROWIDフィールドに適切な値を入力する必要があります。さらに、ROWIDをサポートしていないOracle以外のデータ・ソースに接続しているが、一意キーも使用しているという場合は、3つのトランザクション・トリガーOn-Lock、On-Update、On-Deleteの1つで挿入から問合せに移行するレコードのキーを与える必要があります。これらの条件が満たされていなければ、Form Builderによってエラーが返されます。

4　変更されたがまだコミットまたは消去されていないレコードには、NEW_STATUSを割り当てることはできません。

SET_RECORD_PROPERTYの例

```
/*
** Built-in:SET_RECORD_PROPERTY
** Example:Mark the third record in the EMP block as if it
**          were a queried record.
*/
BEGIN
  Set_Record_Property( 3, 'EMP', STATUS, QUERY_STATUS);
END;
```

SET_RELATION_PROPERTYビルトイン

説明

マスター/ディテール・リレーションに指定されたリレーション・プロパティを設定します。

構文

```
SET_RELATION_PROPERTY
  (relation_id   Relation,
   property      NUMBER,
   value         NUMBER);
SET_RELATION_PROPERTY
  (relation_name VARCHAR2,
   property      NUMBER,
   value         NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

relation_id	リレーション・オブジェクト作成時にForm Builderによってそのリレーションに割り当てられた一意のIDを指定します。これはマスター/ディテール・リレーションを定義するとき自動的に生成されます。あるいはリレーションを明示的に作成することもできます。このIDのデータ型はRELATION型です。
relation_name	ユーザーがリレーション・オブジェクトを定義したときにそのオブジェクトに付けた名前、または FormBuilderによって割り当てられた名前を指定します。その名前のデータ型はVARCHAR2です。

property	次のリレーション・プロパティをどれか1つ指定します。プロパティは定数としてビルトインに渡されます。 AUTOQUERY　このリレーションのディテール・ブロックは、ブロックのインスタンシエーションに基づいて自動的に調整されることを指定します。このようにすると、潜在的に負荷の高い処理をリレーションに組み込まれているブロックが実際に現れるまで延期することができます。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 DEFERRED_COORDINATION　調整が必要なブロックはマークされますが、ディテール・ブロックがインスタンシエートされるまで調整されないことを指定します。遅延調整では、調整の挿入フェーズのみが参照されます。遅延ディテール・ブロックが調整の削除フェースで削除されても、フォームは視覚的に一貫した状態で表示されます。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 MASTER_DELETES　ディテール・ブロックのディテール・レコードの削除について、マスター・ブロックに対応するマスター・レコードが存在しない場合の、デフォルトのリレーション動作を指定します。有効な値は、NON-ISOLATED、ISOLATEDまたはCASCADINGです。プログラミングによりこのプロパティを設定する機能は、独自のマスター/ディテール調整をコーディングする設計者のために用意されている機能です。この機能で、設計時に作成されたデフォルトのリレーションを変更することはできません。 PREVENT_MASTERLESS_OPERATION ディテール・ブロックでの操作は、対応するマスター・レコードが存在しない場合は許可されないことを指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。
value	前述のプロパティに設定できる定数は次のとおりです。 CASCADING　オペレータがマスター・レコードを削除するときに、マスター・レコードがロックされたと同時に対応するディテール・レコードもロックされるように、「レコード削除時の動作」プロパティを設定することを指定します。 ISOLATED　ディテール・ブロックが存在するマスター・レコードをオペレータが削除できるように、「レコード削除時の動作」プロパティを設定することを指定します。この後、ディテール・レコードのロックや削除は行われませんが、この場合もForm Builderによってディテール・ブロックの調整が起動されます。 NON_ISOLATED　オペレータがディテール・レコードが存在するマスター・レコードを削除しようするとForm Builderによってエラー・メッセージが出力され、削除が禁止されるように「レコード削除時の動作」プロパティを設定することを指定します。 PROPERTY_TRUE　プロパティをTRUE状態に設定することを指定します。 PROPERTY_FALSE　プロパティをFALSE状態に設定することを指定します。

SET_RELATION_PROPERTYの制限事項

このビルトインのコールを設定できるプロパティは1つのみです。

SET_RELATION_PROPERTYの例

```
/*
** Built-in:SET_RELATION_PROPERTY
** Example:Set the coordination behavior of a relation to
**          be deferred, and auto-query.
*/
PROCEDURE Make_Relation_Deferred( rl_name VARCHAR2 ) IS
  rl_id Relation;
BEGIN
  /*
  ** Look for the relation's ID
  */
  rl_id := Find_Relation( rl_name );
  /*
  ** Set the two required properties
  */
  Set_Relation_Property(rl_id,AUTOQUERY,PROPERTY_TRUE);
END;
```

SET_REPORT_OBJECT_PROPERTYビルトイン

説明

レポート・プロパティの値をプログラムによって設定します。

構文

```
PROCEDURE SET_REPORT_OBJECT_PROPERTY
(report_id REPORT_OBJECT,
   property NUMBER,
   value VARCHAR2
);
PROCEDURE SET_REPORT_OBJECT_PROPERTY
  (report_name VARCHAR2,
   property NUMBER,
   value VARCHAR2
);
PROCEDURE SET_REPORT_OBJECT_PROPERTY
(report_id REPORT_OBJECT,
```

```
   property NUMBER,
   value NUMBER
);
PROCEDURE SET_REPORT_OBJECT_PROPERTY
  (report_name VARCHAR2,
   property NUMBER,
   value NUMBER
);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

report_id	レポートの一意のIDを指定します。FIND_REPORT_OBJECTを使用すると、特定のレポートのレポートIDを取得できます。
report_name	レポートの一意の名前を指定します。
property	次のいずれかの定数です。 REPORT_EXECUTION_MODE　レポート実行モード、BATCHまたはRUNTIMEです。 REPORT_COMM_MODE　レポート通信モード、SYNCHRONOUSまたはASYNCHRONOUSです。 REPORT_DESTYPE　レポート宛先タイプ、PREVIEW、FILE、PRINTER、MAIL、CACHEまたはSCREENです。 次のいずれかの文字列です。 REPORT_FILENAME　レポートのファイル名です。 REPORT_SOURCE_BLOCK　レポート・ソース・ブロック名です。 REPORT_QUERY_NAME　レポート問合せ名です。 REPORT_DESNAME　レポート宛先名です。 REPORT_DESFORMAT　レポート宛先の書式です。 REPORT_SERVER　レポートのサーバ名です。 REPORT_OTHER　他のユーザーが指定したレポート・プロパティです。
value	次のいずれかの定数です。 REPORT_EXECUTION_MODE　値はBATCHまたはRUNTIMEのいずれかです。 REPORT_COMM_MODE　値はSYNCHRONOUSまたはASYNCHRONOUSのいずれかです。 REPORT_DESTYPE　値はPREVIEW、FILE、PRINTER、MAIL、CACHEまたはSCREENです。

value（続き）	次のいずれかの文字列です。 REPORT_FILENAME　値はVARCHAR2型です。 REPORT_SOURCE_BLOCK　値はVARCHAR2型です。 REPORT_QUERY_NAME　値はVARCHAR2型です。 REPORT_DEST_NAME　値はVARCHAR2型です。 REPORT_DEST_FORMAT　値はVARCHAR2型です。 REPORT_SERVER　値はVARCHAR2型です。 REPORT_OTHER　値はVARCHAR2型です。

使用上の注意

- SET_REPORT_OBJECT_PROPERTYでは、定数値または文字列値を使用してプロパティを設定します。値の型は前述に示すとおり、設定されている特定のプロパティによって異なります。反対に、GET_REPORT_OBJECT_PROPERTYでは、すべてのプロパティについて文字列値を返します。

SET_REPORT_OBJECT_PROPERTYの例

```
DECLARE
   repid REPORT_OBJECT;
   report_prop VARCHAR2(20);
BEGIN
   repid := find_report_object('report4');
   SET_REPORT_OBJECT_PROPERTY(repid, REPORT_EXECUTION_MODE, BATCH);
   SET_REPORT_OBJECT_PROPERTY(repid, REPORT_COMM_MODE, SYNCHRONOUS);
   SET_REPORT_OBJECT_PROPERTY(repid, REPORT_DESTYPE, FILE);
END;
```

SET_TAB_PAGE_PROPERTYビルトイン

説明

指定されたタブ・キャンバスのタブ・ページ・プロパティを設定します。

構文

```
SET_TAB_PAGE_PROPERTY
  (tab_page_id     TAB_PAGE,
   property        NUMBER,
   value           NUMBER);
SET_TAB_PAGE_PROPERTY
  (tab_page_name   VARCHAR2,
   property        NUMBER,
   value           NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

tab_page_id	タブ・ページ作成時にForm Builderによってそのページに割り当てられた一意のID。このIDのデータ型はTAB_PAGEです。
tab_page_name	ユーザーがタブ・ページを定義したときにそのページに付けた名前。データ型はVARCHAR2です。
property	指定したタブ・ページに設定するプロパティ。指定できる値は次のとおりです。 BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 ENABLED タブ・ページを変更可能にする場合文字列TRUEを指定し、使用不可能にする場合（グレーになり使用できない）FALSEを指定します。 FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE ポイント数で指定されるフォントのサイズ。 FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE フォントのスタイル。 FONT_WEIGHT フォントの太さ。 FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 LABEL タブ・ページのラベルの文字列。 VISIBLE タブ・ページを可視にする場合はVARCHAR2値にTRUEを、不可視にする場合はFALSEを指定します。タブ・ページが現在画面にマップされていれば、そのページが他のタブ・ページの後ろに隠れていても、そのページは参照可能とみなされます。 VISUAL_ATTRIBUTE 現在使用されている可視属性の名前を指定します。 WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。

value	次の定数を引数として前述のプロパティの値に渡すことができます。 PROPERTY_TRUE　（プロパティをTRUE状態に設定します）。 PROPERTY_FALSE　（プロパティをFALSE状態に設定します）。

SET_TAB_PAGE_PROPERTYの例

```
/* Example 1:Use SET_TAB_PAGE_PROPERTY to set the
** ENABLED property to TRUE for a tab page (if it currently
** is set to FALSE:
*/

DECLARE
  tb_pg_id  TAB_PAGE;

BEGIN
  tb_pg_id := FIND_TAB_PAGE('tab_page_1');
  IF GET_TAB_PAGE_PROPERTY(tb_pg_id, enabled) = 'FALSE' THEN
     SET_TAB_PAGE_PROPERTY(tb_pg_id, enabled, property_true);
  END IF;
END;
```

SET_TIMERビルトイン

説明

既存のタイマーの設定値を変更します。間隔または繰り返しパラメータ、あるいはその両方を変更できます。

構文

```
SET_TIMER
  (timer_id     Timer,
   milliseconds NUMBER,
   iterate      NUMBER);
SET_TIMER
  (timer_name   VARCHAR2,
   milliseconds NUMBER,
   iterate      NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

timer_id	Form Builderにより、タイマー作成の際に各タイマーに一意に割り当てられるID。タイマーは、CREATE_TIMERビルトインをコールして作成されます。このIDに合ったデータ型の変数にIDを戻す場合は、FIND_TIMERビルトインを実行してください。このIDのデータ型はTIMERです。
timer_name	タイマーを定義する際につけたタイマーの名前。その名前のデータ型はVARCHAR2です。
milliseconds	タイマーの持続時間をミリ秒単位で指定します。このパラメータに指定できる値の範囲は1～2147483648ミリ秒です。2147483648より大きい値は、2147483648に切り捨てられます。指定できるのは正の整数のみであることに注意してください。このパラメータのデータ型はNUMBER型です。詳細は、制限事項を参照してください。 NO_CHANGE　「ミリ秒」プロパティを現在の設定値に保つことを指定します。
iterate	タイマーの反復を指定します。 REPEAT　時間切れ時にタイマーを繰り返すことを示します。これがデフォルトです。 NO_REPEAT　再び明示的にコールされるまで、時間切れ時にタイマーを繰り返さず、1回のみ使用することを示します。 NO_CHANGE　「反復」プロパティを現在の設定値に保つことを指定します。

SET_TIMERの制限事項

- 2147483648を超える値は、2147483648に統一されます。
- 1より小さい値を指定すると、ランタイム・エラーが発生します。
- 前述の上限を超える値を指定すると、整数オーバーフローが発生します。
- ミリ秒を負数で表すことはできません。
- 大文字か小文字かに関係なく、2つのタイマーが同じフォーム・インスタンスで同じ名前を共有することはできません。
- タイマーの実行時に定義されたWhen-Timer-Expiredトリガーがない場合は、エラーが戻されます。
- タイマーの実行時に定義されたWhen-Timer-Expiredトリガーがなく、タイマーが繰返しタイマーの場合、その後の繰返しは取り消されますが、タイマーは保持されます。

SET_TIMERの例

FIND_TIMERの例を参照してください。

SET_TREE_NODE_PROPERTYビルトイン

説明

ブランチ・ノードの状態を設定します。

構文

```
PROCEDURE SET_TREE_NODE_PROPERTY
  (item_name VARCHAR2,
   node FTREE.NODE,
   property NUMBER,
   value NUMBER);
PROCEDURE SET_TREE_NODE_PROPERTY
  (item_name VARCHAR2,
   node FTREE.NODE,
   property NUMBER,
   value VARCHAR2);
PROCEDURE SET_TREE_NODE_PROPERTY
  (item_id ITEM,
   node FTREE.NODE,
   property NUMBER,
   value NUMBER);
PROCEDURE SET_TREE_NODE_PROPERTY
  (item_id ITEM,
   node FTREE.NODE,
   property NUMBER,
   value VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
node	有効なノードを指定します。
property	次のいずれかのプロパティを指定します。 NODE_STATE　有効な値は、EXPANDED_NODE、COLLAPSED_NODEおよびLEAF_NODEです。 NODE_LABEL　ノードのラベルを設定します。 NODE_ICON　ノードのアイコンを設定します。 NODE_VALUE　ノードの値を設定します。
value	実際に渡す値です。

SET_TREE_NODE_PROPERTYの例

```
/*
** Built-in:SET_TREE_NODE_PROPERTY
*/

-- This code could be used in a WHEN-TREE-NODE-SELECTED
-- trigger to change the icon of the node clicked on.

DECLARE
htree        ITEM;
current_node  FTREE.NODE;
find_node     FTREE.NODE;
BEGIN
-- Find the tree itself.
htree := Find_Item('tree_block.htree3');

-- Change it icon of the clicked node.
-- The icon file will be located using the
-- UI60_ICON environment variable in client/server
-- or in the virtual directory for web deployment.
Ftree.Set_Tree_Node_Property(htree, :SYSTEM.TRIGGER_NODE,
Ftree.NODE_ICON, 'Open');
END;
```

SET_TREE_PROPERTYビルトイン

説明

指定された階層ツリーのプロパティを設定します。

構文

```
PROCEDURE SET_TREE_PROPERTY
  (item_name VARCHAR2,
   property NUMBER,
   value NUMBER);
PROCEDURE SET_TREE_PROPERTY
  (item_name VARCHAR2,
   property NUMBER,
   value VARCHAR2);
PROCEDURE SET_TREE_PROPERTY
  (item_name VARCHAR2,
   property NUMBER,
   value RECORDGROUP);
PROCEDURE SET_TREE_PROPERTY
  (item_id ITEM,
   property NUMBER,
   value NUMBER);
PROCEDURE SET_TREE_PROPERTY
  (item_id ITEM,
   property NUMBER,
   value VARCHAR2);
PROCEDURE SET_TREE_PROPERTY
  (item_id ITEM,
   property NUMBER,
   value RECORDGROUP);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_name	**設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。**

Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
property	次のいずれかのプロパティを指定します。 RECORD_GROUP 階層ツリーのデータ・セットをレコード・グループに置き換えて、表示します。 QUERY_TEXT 階層ツリーのデータ・セットをSQL問合せに置き換えて、表示します。 ALLOW_EMPTY_BRANCHES 有効な値はPROPERTY_TRUEとPROPERTY_FALSEです。
value	設定するプロパティに合った値を指定します。 PROPERTY_TRUE プロパティがTRUE状態に設定されます。 PROPERTY_FALSE プロパティがFALSE状態に設定されます。

SET_TREE_PROPERTYの例

```
/*
** Built-in:SET_TREE_PROPERTY
*/

-- This code could be used in a WHEN-NEW-FORM-INSTANCE
-- trigger to initially populate the hierarchical tree
-- with data.

DECLARE
   htree        ITEM;
   v_ignore     NUMBER;
   rg_emps      RECORDGROUP;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Check for the existence of the record group.
   rg_emps := Find_Group('emps');
   IF NOT Id_Null(rg_emps) THEN
   DELETE_GROUP(rg_emps);
   END IF;

   -- Create the record group.
   rg_emps := Create_Group_From_Query('rg_emps',
   'select 1, level, ename, NULL, to_char(empno) ' ||
   'from emp '                                     ||
   'connect by prior empno = mgr '                    ||
```

```
    'start with job = ''PRESIDENT''');

    -- Populate the record group with data.
    v_ignore := Populate_Group(rg_emps);

    -- Transfer the data from the record group to the hierarchical
    -- tree and cause it to display.
    Ftree.Set_Tree_Property(htree, Ftree.RECORD_GROUP, rg_emps);
END;
```

SET_TREE_SELECTIONビルトイン

説明

単一ノードの選択を指定します。

構文

```
PROCEDURE SET_TREE_SELECTION
  (item_name VARCHAR2,
   node NODE,
   selection_type NUMBER);
PROCEDURE SET_TREE_SELECTION
  (item_id ITEM,
   node NODE,
   selection_type NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

不可

パラメータ

item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
Item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。IDのデータ型はITEMです。
node	有効なノードを指定します。

selection_type	選択のタイプを指定します。 SELECT_ON　ノードを選択します。 SELECT_OFF　ノードを選択解除します。 SELECT_TOGGLE　ノードの選択状態を切り替えます。

SET_TREE_SELECTIONの例

```
/*
** Built-in:SET_TREE_SELECTION
*/

-- This code could be used in a WHEN-TREE-NODE-EXPANDED
-- trigger and will mark the clicked node as selected.

DECLARE
   htree        ITEM;
BEGIN
   -- Find the tree itself.
   htree := Find_Item('tree_block.htree3');

   -- Mark the clicked node as selected.
   Ftree.Set_Tree_Selection(htree, :SYSTEM.TRIGGER_NODE,
   Ftree.SELECT_ON);
END;
```

SET_VA_PROPERTYビルトイン

説明

指定されたプロパティについて、可視属性プロパティの値を変更します。

構文

```
SET_VA_PROPERTY
  (va_id VISUALATTRIBUTE
   property NUMBER
   value VARCHAR2);
SET_VA_PROPERTY
  (va_name VARCHAR2
   property NUMBER
   value VARCHAR2);
SET_VA_PROPERTY
```

```
  (va_id VISUALATTRIBUTE
   property NUMBER
   value NUMBER);
SET_VA_PROPERTY
  (va_name VARCHAR2
   property NUMBER
   value NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

問合せ入力モード

可

パラメータ

va_id	可視属性の作成時にForm Builderによって割り当てられる一意のID。データ型はVISUALATTRIBUTEです。
va_name	可視属性の作成時に付けた名前。データ型はVARCHAR2です。
Property	次のいずれかのプロパティを指定します。 BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE ポイント数に100を掛けた値で指定されるフォントのサイズ。 FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE フォントのスタイル。 FONT_WEIGHT フォントの太さ。 FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。

value	WHITE_ON_BLACK	白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。所定のプロパティに適用される値を指定します。プロパティのデータ型によって、入力する値のデータ型が決まります。たとえば、WHITE_ON_BLACKプロパティをTRUEに設定する場合、その値について定数PROPERTY_TRUEを指定します。項目についてFONT_NAMEを変更する場合は、値を指定します。つまり、ラベルをVARCHAR2文字列として指定します。
	PROPERTY_TRUE	プロパティをTRUE状態に設定することを指定します。
	PROPERTY_FALSE	プロパティをFALSE状態に設定することを指定します。プロパティの値を設計時に確定した元の値にリセットする場合は、2つの一重引用符（''）を間にスペースを入れずに入力します。たとえば、SET_ITEM_PROPERTY('DEPTNO',FONT_SIZE, '');と入力すると、書式サイズは設計時の値にリセットされます。

SET_VARビルトイン

説明

新規作成されたOLEバリアントを初期値に設定します。または、既存のOLEバリアントを新しい値にリセットします。

プロシージャには4つのバージョンがあり、それぞれがCHAR、NUMBER、OLEVARおよびtableといった新しい値の型について用意されています。

構文

```
PROCEDURE SET_VAR
  (var OLEVAR, newval CHAR
   vtype VT_TYPE, arrspec VARCHAR2);


PROCEDURE SET_VAR
  (var OLEVAR, newval NUMBER
   vtype VT_TYPE, arrspec VARCHAR2);


PROCEDURE SET_VAR
  (var OLEVAR, newval OLEVAR
   vtype VT_TYPE, arrspec VARCHAR2);


PROCEDURE SET_VAR
  (var OLEVAR, source_table,
```

```
  vtype VT_TYPE, arrspec VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

var	設定するバリアント。
newval	バリアントに付与される値。
vtype	バリアントに付与されるOLE VT_TYPE。 これはオプションのパラメータです。指定されなかった場合、プロシージャのNUMBERバージョンのデフォルト値はVT_R8です。VARCHAR2バージョンに対しては、デフォルト値はVT_BSTRです。OLEVARバージョンの場合、デフォルトはVT_VARIANTです。つまり、バリアント自体によって実際に指定される型です。
source_table	ディメンションおよび要素の値がバリアントに付与されるPL/SQL表。
arrspec	新規バリアントの作成で、ソース表で選択されたどの要素が使用されるかを示します。詳細は、「OLE配列の指定子」」を参照してください。 これはオプションのパラメータです。指定されなかった場合、ソース表全体が使用されます。

使用上の注意

SET_VARプロシージャのターゲット・バリアントは、CREATE_VARファンクションを使用して最初に作成する必要があります。

SET_VIEW_PROPERTYビルトイン

説明

指定されたキャンバスのプロパティを設定します。各ビルトインのに設定できるプロパティは１つのみです。つまり、x座標をX_POSに適用し、y座標をHEIGHTに適用するように、引数を分割することはできません。

構文

```
SET_VIEW_PROPERTY
  (view_id   ViewPort,
   property  NUMBER,
   value     NUMBER);
SET_VIEW_PROPERTY
  (view_id   ViewPort,
   property  NUMBER,
   x         NUMBER,
   y         NUMBER);
```

```
SET_VIEW_PROPERTY
  (view_name  VARCHAR2,
   property   NUMBER,
   value      NUMBER);
SET_VIEW_PROPERTY
  (view_name  ViewPort,
   property   NUMBER,
   x          NUMBER,
   y          NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

view_id	キャンバス/ビューの作成時にForm Builderによってそのキャンバス/ビューに割り当てられる一意のID。FIND_VIEWのビルトインを使用して、そのIDを適切なタイプの変数に戻します。このIDのデータ型はVIEWPORTです。
view_name	キャンバス・オブジェクトの定義時に付けた名前。データ型はVARCHAR2です。
property	次のプロパティのいずれかを指定します。 DIRECTION 双方向オブジェクトのレイアウト方向。有効な値は、DIRECTION_DEFAULT、RIGHT_TO_LEFTおよびLEFT_TO_RIGHTです。 DISPLAY_POSITION スタック・ビューについて、ウィンドウのコンテント・ビューに対して相対的なビューの左上角の位置をX、Yの組合せで表します。プロパティによって、ウィンドウ内のどこにビューが表示されるかが決まります。 HEIGHT スタック・キャンバスについて、ビューの高さを表します。キャンバス自体のサイズを変更するには、SET_CANVAS_PROPERTYを使用します。 POSITION_ON_CANVAS キャンバスに対しビューの左上角の位置を相対的に表すX、Yの組合せ。 VIEWPORT_X_POS スタック・ビューについて、ウィンドウのコンテント・ビューに対して相対的なビューの左上角のX座標。 VIEWPORT_Y_POS スタック・ビューについて、ウィンドウのコンテント・ビューに対して相対的なビューの左上角のY座標。

property（続き）	VIEWPORT_X_POS_ON_CANVAS キャンバスに対して相対的なビューの左上角のX座標。 VIEWPORT_Y_POS_ON_CANVAS キャンバスに対して相対的なビューの左上角のY座標。 VIEW_SIZE　スタック・キャンバスについて、幅と高さの組合せによるビューのサイズ。キャンバス自体のサイズを変更するには、SET_CANVAS_PROPERTYを使用します。 VISIBLE　ビューを表示するかどうかを示すプロパティ。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 WIDTH　スタック・キャンバスについて、ビューの幅を表します。キャンバス自体のサイズを変更するには、SET_CANVAS_PROPERTYを使用します。
value	設定するプロパティに合った値を指定します。 PROPERTY_TRUE　プロパティがTRUE状態に設定されます。 PROPERTY_FALSE　プロパティがFALSE状態に設定されます。
x	指定したプロパティの種類に応じて、x座標つまり幅を表すNUMBER値。引数はフォームの座標システムの単位で指定します。
y	指定したプロパティの種類に応じて、y座標つまり高さを表すNUMBER値。引数はフォームの座標システムの単位で指定します。

SET_WINDOW_PROPERTYビルトイン

説明

指定されたウィンドウのプロパティを設定します。

構文

```
SET_WINDOW_PROPERTY
  (window_id    Window,
   property     NUMBER,
   value        VARCHAR2);
SET_WINDOW_PROPERTY
  (window_id    Window,
   property     NUMBER,
   x            NUMBER);
SET_WINDOW_PROPERTY
  (window_id    Window,
   property     NUMBER,
   x            NUMBER,
   y            NUMBER);
SET_WINDOW_PROPERTY
  (window_name  VARCHAR2,
```

```
   property     NUMBER,
   value        VARCHAR2);
SET_WINDOW_PROPERTY
  (window_name  VARCHAR2,
   property     NUMBER,
   x            NUMBER);
SET_WINDOW_PROPERTY
  (window_name  VARCHAR2,
   property     NUMBER,
   x            NUMBER,
   y            NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

window_id	Form Builderがウィンドウ作成時にそのウィンドウに割り当てられる一意のIDを指定します。FIND_WINDOWのビルトインを使用して、そのIDを適切なタイプの変数に戻します。このIDのデータ型はWINDOW型です。
window_name	ウィンドウの作成時に付けた名前を指定します。その名前のデータ型はVARCHAR2です。
property	次のいずれかのウィンドウ・プロパティを指定します。 BACKGROUND_COLOR オブジェクトのバックグラウンド領域のカラー。 DIRECTION 双方向オブジェクトのレイアウト方向を指定します。有効な値は、DIRECTION_DEFAULT、RIGHT_TO_LEFTおよびLEFT_TO_RIGHTです。 FILL_PATTERN オブジェクトの塗り領域で使用されるパターン。パターンは、「バックグラウンド・カラー」と「フォアグラウンド・カラー」に指定された2つのカラーでレンダリング処理されます。 FONT_NAME オブジェクト内のテキストに使用されるフォントのファミリつまりタイプフェース。どのようなフォントが使用できるかはシステムによって異なります。 FONT_SIZE ポイント数で指定されるフォントのサイズ。 FONT_SPACING フォントの幅つまり文字間のスペース（カーニング）。 FONT_STYLE フォントのスタイル。 FONT_WEIGHT フォントの太さ。

property（続き）	FOREGROUND_COLOR オブジェクトのフォアグラウンド領域のカラー。項目では、「フォアグラウンド・カラー」属性は項目内に表示されるテキストのカラーを定義します。 HEIGHT ウィンドウの高さを指定します。 HIDE_ON_EXIT オペレータが別のウィンドウの項目へナビゲートしたとき、Form Builderによって自動的に現在のウィンドウを非表示にするかどうかを指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 ICON_NAME ウインドウが最小化されるときにウインドウ項目に関連するアイコン・リソースのファイル名を指定します。 POSITION ウィンドウの画面位置を示すx、yの組合せを指定します。 TITLE ウィンドウのタイトルを設定します。 VISIBLE ウィンドウを表示するかどうかを指定します。有効な値は、PROPERTY_TRUEとPROPERTY_FALSEです。 WHITE_ON_BLACK 白黒ビットマップ表示デバイスで、オブジェクトを黒いバックグラウンド上に白いテキストとして表示するように指定します。 WINDOW_SIZE 画面上のウィンドウ・サイズを示す幅、高さの組合せを指定します。 WINDOW_STATE ウィンドウの現行の表示状態を指定します。有効な値は、NORMAL、MAXIMIZEまたはMINIMIZEです。 WIDTH ウィンドウの幅を指定します。 X_POS 画面上のウィンドウの左上角のx座標を設定します。 Y_POS 画面上のウィンドウの左上角のy座標を設定します。
value	次の定数を前述のプロパティの値に引数として渡すことができます。 PROPERTY_TRUE プロパティをTRUE状態に設定することを指定します。この値は、VISIBLEプロパティについてのみ適用されます。 PROPERTY_FALSE プロパティをFALSE状態に設定することを指定します。この値は、VISIBLEプロパティについてのみ適用されます。 次の定数は、WINDOW_STATEプロパティで使用するための引数として渡すことできます。 NORMAL 「幅」、「高さ」、「X位置」および「Y位置」の各プロパティの現行の設定値に従って、ウィンドウを通常通りに表示することを指定します。 MAXIMIZE ウィンドウ・マネージャの表示形式に従って、画面全体にウィンドウを拡大して表示することを指定します。 MINIMIZE ウィンドウを最小化、つまりアイコン化することを指定します。
x	指定したプロパティの種類に応じて、x座標つまり幅を表すNUMBER値を指定します。引数はフォームの座標システムの単位で指定します。
y	指定したプロパティの種類に応じて、y座標つまり高さを表すNUMBER値を指定します。引数はフォームの座標システムの単位で指定します。

使用上の注意

Microsoft Windowsでは、フォームはMDIアプリケーション・ウィンドウ内で実行されます。SET_WINDOW_PROPERTYビルトインを使用して、MDIアプリケーション・ウィンドウの次のプロパティを設定できます。

- TITLE
- POSITION
- WIDTH, HEIGHT
- WINDOW_SIZE
- WINDOW_STATE
- X_POS, Y_POS

SET_WINDOW_PROPERTYコールでMDIアプリケーション・ウィンドウを参照するには、次のように定数FORMS_MDI_WINDOWを使用します。

```
Set_Window_Property(FORMS_MDI_WINDOW, POSITION, 5,10)
Set_Window_Property(FORMS_MDI_WINDOW, WINDOW_STATE, MINIMIZE)
```

SET_WINDOW_PROPERTYの制限事項

- ウィンドウのサイズまたは位置を変更すると、変更は、フォームを実行中、またはウィンドウのサイズまたは位置を明示的に再度変更するまで有効です。ウィンドウを一度閉じ、もう一度開いても、ウィンドウは設計時のデフォルト値にリセットされません。ウィンドウを設計時のデフォルト値に戻す場合は、設計時のデフォルト値を変数に割り当てる必要があります。

SET_WINDOW_PROPERTYの例

FIND_WINDOWの例を参照してください。

SHOW_ALERTビルトイン

説明

指定された警告を表示し、オプレータが3つの警告ボタンをどれか1つ選択すると数値を返します。

構文

```
SHOW_ALERT
  (alert_id  Alert);
SHOW_ALERT
  (alert_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

オペレータが警告発行時に選択するボタンに対応する数値定数。ボタンの対応付けは警告設計時に指定します。

オペレータが選択するボタン	Form Builderによって返される定数
ボタン1	ALERT_BUTTON1
ボタン2	ALERT_BUTTON2
ボタン3	ALERT_BUTTON3

問合せ入力モード

可

パラメータ

alert_id	警告の作成時にForm Builderによって割り当てられる一意のID。IDを適切な型の変数に戻すには、FIND_ALERTビルトインを使用します。IDのデータ型はAlertです。
alert_name	警告の定義時に付けた名前。その名前のデータ型はVARCHAR2です。

SHOW_ALERTの例

FIND_ALERTとSET_ALERT_PROPERTYの例を参照してください。

SHOW_EDITORビルトイン

説明

指定された位置に指定されたエディタを表示して、そのエディタに文字列を渡したり、そのエディタから既存の文字列を取り出したりします。位置が指定されていない場合は、設計時にそのエディタに指定されたデフォルトの位置に表示されます。

構文

```
SHOW_EDITOR
  (editor_id    Editor,
   message_in   VARCHAR2,
   message_out  VARCHAR2,
   result       BOOLKAN);
SHOW_EDITOR
  (editor_id    Editor,
   message_in   VARCHAR2,
   x            NUMBER,
   y            NUMBER,
   message_out  VARCHAR2,
   result       BOOLEAN);
SHOW_EDITOR
  (editor_name  VARCHAR2,
   message_in   VARCHAR2,
   message_out  VARCHAR2,
   result       BOOLEAN);
SHOW_EDITOR
  (editor_name  VARCHAR2,
   message_in   VARCHAR2,
   x            NUMBER,
   y            NUMBER,
   message_out  VARCHAR2,
   result       BOOLEAN);
```

ビルトイン・タイプ

OUTパラメータ（*result*および*message_out*）を2つ返す制限なしプロシージャ

問合せ入力モード

可

パラメータ

editor_id	エディタ作成時にForm Builderによってそのエディタに割り当てられた一意のIDを指定します。FIND_EDITORビルトインを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型はEditor型です。
editor_name	ユーザーがエディタを定義した時にそのエディタに付けた名前を指定します。その名前のデータ型はVARCHAR2です。
message_in	VARCHAR2型の必須INパラメータを指定します。このパラメータにはNULL値を渡すことができます。また、テキスト項目や変数も参照できます。
x	エディタのX座標を指定します。この引数はINTEGERで指定します。
y	エディタのY座標を指定します。この引数はINTEGERで指定します。

message_out	VARCHAR2型の必須OUTパラメータを指定します。また、テキスト項目や変数も参照できます。オペレータがエディタを取り消すと、*result*はFALSEに、*message_out*はNULLになります。
result	BOOLEAN型の必須OUTパラメータを指定します。オペレータがエディタを受け入れると、*result*はTRUEになります。オペレータがエディタを取り消すと、*result*はFALSEに、*message_out*はNULLになります。

SHOW_EDITORの制限事項

- *message_out*に指定された変数またはテキスト項目の長さによってエディタが受け入れられる最大文字数が決まるため、*message_out*の長さは*message_in*の長さ以上にしてください。
- message_inパラメータの値は、エディタに渡されるときにForm Builderによって必ずVARCHAR2に変換されます。しかし、ユーザーが*message_out*をVARCHAR2以外のオブジェクトに渡す場合は、まずその値をある変数に渡して変換し、次にPL/SQL関数TO_DATEまたはTO_NUMBERを使用してその変数上で型変換を実行する必要があります。
- 横幅には、少なくともエディタ・ウィンドウ下部のボタンが表示できる長さを指定してください。

SHOW_EDITORの例

```
/*
** Built-in:SHOW_EDITOR
** Example:Accept input from the operator in a user-defined
**          editor.Use the system editor if the user has
**          checked the "System_Editor" menu item under the
**          "Preferences" menu in our custom menu module.
*/
DECLARE
  ed_id   Editor;
  mi_id   MenuItem;
  ed_name VARCHAR2(40);
  val     VARCHAR2(32000);
  ed_ok   BOOLEAN;
BEGIN
  mi_id := Find_Menu_Item('PREFERENCES.SYSTEM_EDITOR');
  IF Get_Menu_Item_Property(mi_id,CHECKED) = 'TRUE' THEN
    ed_name := 'system_editor';
  ELSE
    ed_name := 'my_editor1';
  END IF;

  ed_id := Find_Editor( ed_name );
  /*
```

```
  ** Show the appropriate editor at position (10,14) on the
  ** screen.Pass the contents of the :emp.comments item
  ** into the editor and reassign the edited contents if
  ** 'ed_ok' returns boolean TRUE.
  */
  val := :emp.comments;
  Show_Editor( ed_id, val, 10,14, val, ed_ok);
  IF ed_ok THEN
    :emp.comments := val;
  END IF;
END;
```

SHOW_KEYSビルトイン

説明

「キー一覧」画面を表示します。オペレータがファンクション・キーを押すと、Form Builderによって SHOW_KEYSビルトインが起動される前のフォームが再表示されます。

構文

```
SHOW_KEYS;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

SHOW_KEYSの例

```
/*
** Built-in:SHOW_KEYS
** Example:BEGIN
 Show_Keys;
END;
```

SHOW_LOVビルトイン

説明

指定された位置に値リスト（LOV）ウィンドウを表示し、オペレータがリストから値を選択した場合はTRUEを、オペレータがリストを取り消して画面から消すとFALSEを返します。

構文

```
SHOW_LOV
  (lov_id     LOV);
SHOW_LOV
  (lov_id     LOV,
              NUMBER,
   y          NUMBER);
SHOW_LOV
  (lov_name   VARCHAR2);
SHOW_LOV
  (lov_name   VARCHAR2,
   x          NUMBER,
   y          NUMBER);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

BOOLEAN

問合せ入力モード

可

パラメータ

lov_id	値リスト作成時にForm Builderによってその値リストに割り当てられた一意のIDを指定します。FIND_LOVビルトインを使用すると、このIDを適切な型の変数に返すことができます。このIDのデータ型は値リスト型です。
lov_name	ユーザーが値リストを定義したときにその値リストに付けた名前。その名前のデータ型はVARCHAR2です。
x	値リストのX座標を指定します。
y	値リストのY座標を指定します。

使用上の注意

値リストを表示するためにSHOW_LOVを使用すると、Form Builderは値リストの「自動スキップ」プロパティを無視します。

カーソルを次のナビゲーション項目に移動する場合は、LIST_VALUESビルトインを使用します。

SHOW_LOVの制限事項

lov_name引数を指定せず、かつ現項目に関連付けられた値リストが存在しない場合は、Form Builderによってエラーが出力されます。

値リストの基礎を形成するレコード・グループにレコードが1つも含まれていない場合は、SHOW_LOVのBOOLEAN戻り値はFALSEになります。

SHOW_LOVの例

```
/*
** Built-in:SHOW_LOV
** Example:Display a named List of Values (LOV)
*/
DECLARE
  a_value_chosen BOOLEAN;
BEGIN
  a_value_chosen := Show_Lov('my_employee_status_lov');
  IF NOT a_value_chosen THEN
    Message('You have not selected a value.');
    Bell;
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

SHOW_MENUビルトイン

説明

現行のメニューが現在表示されていなければ表示します。ただし、表示されたメニューはアクティブになりません。

SHOW_MENUはメニューをアクティブにしないため、メニューで現行のキャンバスの一部を隠すことはできません。したがって、現行のキャンバスがメニューの上に被さっている場合は、メニュー全体またはメニューの一部が画面に表示されません。

構文

```
SHOW_MENU;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

SHOW_MENUの制限事項

このビルトインを使用できるのは、文字モード環境のみです。

SHOW_MENUの例

```
/*
** Built-in:SHOW_MENU
** Example:Display the menu if no canvas overlays it.
*/
BEGIN
  Show_Menu;
END;
```

SHOW_VIEWビルトイン

説明

所定のキャンバスを、キャンバスの「キャンバス上のビューポートのX位置」および「キャンバス上のビューポートのY位置」プロパティの設定によって指定された座標に表示します。そのビューがすでに表示されている場合は、SHOW_VIEWによって同じウィンドウ内の他のビューの一番手前にそのビューが移動します。

構文

```
SHOW_VIEW
  (view_id  ViewPort);
SHOW_VIEW
  (view_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

view_id	Form Builderによりオブジェクト作成時にそのビューに割り当てられる一意のIDを指定します。FIND_VIEWのビルトインを使用して、そのIDを適切なタイプの変数に戻します。このIDのデータ型はViewPort型です。
view_name	ユーザーがビューを定義したときにそのビューに付けた名前を指定します。その名前のデータ型はVARCHAR2です。

SHOW_VIEWの例

```
/*
** Built-in:SHOW_VIEW
** Example:Programmatically display a view in the window to
**          which it was assigned at design time.
*/
BEGIN
  Show_View('My_Stacked_Overlay');
END;
```

SHOW_WINDOWビルトイン

説明

所定のウィンドウを、任意で取り込まれたXY座標か、ウィンドウの現在のXY座標に表示します。指定されたウィンドウがモーダル・ウィンドウの場合は、SHOW_WINDOWがモーダル・ウィンドウの最初のナビゲート可能項目へのGO_ITEMコールとして実行されます。

構文

```
SHOW_WINDOW
  (window_id  Window);
SHOW_WINDOW
  (window_id  Window,
   x         NUMBER,
   y         NUMBER);
SHOW_WINDOW
  (window_name  VARCHAR2);
SHOW_WINDOW
```

```
(window_name  VARCHAR2,
 x            NUMBER,
 y            NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

window_id	Form Builderがウィンドウ作成時にそのウィンドウに割り当てられる一意のIDを指定します。FIND_WINDOWのビルトインを使用して、そのIDを適切なタイプの変数に戻します。このIDのデータ型はWINDOW型です。
window_name	ユーザーがウィンドウを定義したときにそのウィンドウに付けた名前を指定します。その名前のデータ型はVARCHAR2です。
x	ウィンドウのX座標を指定します。この引数はINTEGERで指定します。
y	ウィンドウのY座標を指定します。この引数はNUMBERで指定します。

SHOW_WINDOWの例

```
/*
** Built-in:SHOW_WINDOW
** Example:Override the default (x,y) coordinates for a
**           windows location while showing it.
*/
BEGIN
  Show_Window('online_help',20,5);
END;
```

SYNCHRONIZEビルトイン

説明

端末の画面をフォームの内部状態と同期化します。つまり、SYNCHRONIZEは、画面の内部表現としてForm Builderが保持している情報を反映するように画面上の表示を更新します。

構文

```
SYNCHRONIZE;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

SYNCHRONIZEの制限事項

次の条件が両方ともTrueの場合にのみ、SYNCHRONIZEは画面上の表示を更新します。

- Form Builderがフォーム階層の項目レベルにある（すなわち、SYSTEM.CURRENT_ITEMがNULLではない）場合。

SYNCHRONIZEの例

```
/*
** Built-in:  SYNCHRONIZE
** Example:   Achieve an odometer effect by updating the
**           screen as an items value changes quickly.
**           Without synchronize, the screen is typically
**           only updated when Form Builder completes all trigger
**           execution and comes back for user input.
*/
BEGIN
  FOR j IN 1..1000 LOOP
    :control.units_processed := j;
    SYNCHRONIZE;
    Process_Element(j);
  END LOOP;
END;
```

TERMINATEビルトイン

説明

TERMINATEによって、フォームまたはダイアログ・ボックスへの入力が終了されます。この機能は、オペレータが[アクセプト]を押すのと同じです。

構文

```
PROCEDURE TERMINATE;
```

ビルトイン・タイプ

制限付きプロシージャ

パラメータ

なし

TERMINATEの制限事項

Terminateは「パラメータ値入力」ダイアログにのみ適用されます。

TO_VARIANTビルトイン

説明

OLEバリアントを作成し、それに値を割り当てます。ファンクションのバージョンは4つ存在します。

構文

```
FUNCTION TO_VARIANT
  (newval NUMBER,
   vtype VT_TYPE
   persistence BOOLEAN)
RETURN newvar OLEVAR;


FUNCTION TO_VARIANT
  (newval VARCHAR2,
   vtype VT_TYPE
   persistence BOOLEAN)
RETURN newvar OLEVAR;
```

```
FUNCTION TO_VARIANT
  (source_table,
   vtype VT_TYPE
   arrspec VARCHAR2, persistence BOOLEAN)
RETURN newvar OLEVAR;


FUNCTION TO_VARIANT
  (var OLEVAR,
   vtype VT_TYPE
   arrspec VARCHAR2, persistence BOOLEAN)
RETURN newvar OLEVAR;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

新しく作成されたOLEバリアント

パラメータ

newval	新しく作成されるOLEバリアントに与えられる値です。
vtype	新しく作成されるバリアントに対して与えられるOLE VT_TYPEです。 これはオプションのパラメータです。指定されない場合、ファンクションのNUMBERバージョンのデフォルト値はVT_R8です。VARCHAR2バージョンに対しては、デフォルト値はVT_BSTRです。表バージョンに対しては、表のPL/SQL型によってデフォルト値が決まります。OLEVARバージョンに対しては、デフォルト値はソース・バリアントの型です。
persistence	作成後のバリアントの永続性を制御します。ブール値がTRUEの場合、永続としてバリアントが作成されます。値がFALSEの場合、非永続してバリアントが作成されます。 これはオプションのパラメータです。指定されない場合のデフォルト値は非永続です。
source_table	新しく作成されるバリアント表の範囲と値を確定するために使用される既存のPL/SQL表です。ソース表のタイプは何であってもかまいません。
arrspec	新しいバリアントの作成に使用される、選択済みの要素またはソース表の要素の種類を示します。下限は常に1から始まります。詳細は、OLE配列の指定子を参照してください。 これはオプションのパラメータです。指定されない場合、ソース表全体またはソース・バリアント全体が使用されます。
var	その値が新しいバリアントに与えられる既存のOLEバリアント。このソース・バリアントは表であってもかまいません。

使用上の注意

- このファンクションは、まず空のバリアントを作成し、次に値を与えます。このファンクションの働きはCREATE_VAR操作とSET_VAR操作とを組み合わせたものです。
- このTO_VARIANTファンクションは、VAR_TO_*ファンクションの逆バージョンと考えることもできます。
- このファンクションのOLEVARバージョンは、PL/SQLの等価の値ではなく、既存のOLEバリアントをソースとして使用する点で、NUMBER、VARCHAR2および表の各バージョンとは異なることに注意してください。

UNSET_GROUP_SELECTIONビルトイン

構文

```
PROCEDURE UNSET_GROUP_SELECTION
  (recordgroup_id  RecordGroup,
   row_number         NUMBER);
PROCEDURE UNSET_GROUP_SELECTION
  (recordgroup_name  VARCHAR2,
   row_number         NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

説明

指示されたレコード・グループ内の指定行のマークを解除します。この手順により、前のSET_GROUP_SELECTIONのコールによりプログラム的に選択されている行のマークを解除します。

行番号が1から順に付けられます。たとえば、行3、8および12を選択すると、Form Builderではこれらの行が選択1、2および3とみなされます。RESET_GROUP_SELECTIONビルトインをコールすることによりグループ全体の行選択を取り消せます。

パラメータ

recordgroup_id	Form Builderによりレコード・グループ作成時にそのコード・グループに割り当てられる一意のIDを指定します。FIND_GROUPビルトインを使用して、そのIDを変数に戻します。そのIDのデータ型はRecordGroupです。

recordgroup_name	ユーザーがレコード・グループ作成時に与えたレコード・グループ名を指定します。その名前のデータ型はVARCHAR2です。
row_number	ユーザーが選択するレコード・グループ行の番号を指定します。ユーザーが指定する値はNUMBERです。

UNSET_GROUP_SELECTIONの例

```
/*
** Built-in:UNSET_GROUP_SELECTION
** Example:Clear all of the even rows as selected in the
**          record group whose id is passed-in as a
**          parameter.
*/
PROCEDURE Clear_Even_Rows ( rg_id RecordGroup ) IS
BEGIN
  FOR j IN 1..Get_Group_Row_Count(rg_id) LOOP
    IF MOD(j,2)=0 THEN
      Unset_Group_Selection( rg_id, j );
    END IF;
  END LOOP;
END;
```

UPビルトイン

説明

順序番号が次に小さいレコード内にある現項目のインスタンスにナビゲートします。

構文

```
PROCEDURE UP;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

UPDATE_CHARTビルトイン

説明

データ・ブロックは、問合せを受けたり、変更を加えられたりするたびに更新されます。デフォルトでは、ブロックが更新されると、そのデータ・ブロックを基にしたすべてのグラフが自動的に更新されます。UPDATE_CHARTビルトインを使用すると、グラフの基になっているデータ・ブロックが更新されていない場合でも、グラフ項目を明示的に更新できます。たとえば、データ・ブロック内でまだ発生していない変更を反映するようにグラフを更新できます。

構文

```
PROCEDURE UPDATE_CHART
  (chart_name VARCHAR2,
   param_list_id TOOLS.PARAMLIST
);
PROCEDURE UPDATE_CHART
  (chart_name VARCHAR2,
   param_list_name VARCHAR2
);
PROCEDURE UPDATE_CHART
  (chart_id FORMS4C.ITEM,
   param_list_id TOOLS.PARAMLIST
);
PROCEDURE UPDATE_CHART
  (chart_id FORMS4C.ITEM,
   param_list_name VARCHAR2
);
PROCEDURE UPDATE_CHART
  (chart_id FORMS4C.ITEM
);
PROCEDURE UPDATE_CHART
  (chart_name VARCHAR2
);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

chart_id	グラフの一意のIDを指定します。

chart_name	グラフの一意の名前を指定します。
param_list_id	グラフ・パラメータ・リストの一意のIDを指定します。
param_list_name	グラフ・パラメータ・リストの一意の名前を指定します。

UPDATE_RECORDビルトイン

説明

On-Updateトリガーからコールされると、ポストおよびコミット・トランザクション処理中にデータベース内のレコードを更新するデフォルトのForm Builderの処理が開始されます。

このビルトインは、主に非Oracleデータ・ソースに対して実行されるアプリケーション用に組み込まれます。

構文

```
PROCEDURE UPDATE RECORD;
```

ビルトイン・タイプ

制限付きプロシージャ

問合せ入力モード

不可

パラメータ

なし

UPDATE_RECORDの制限事項

On-Updateトリガーにおいてのみ有効です。

USER_EXITビルトイン

説明

user_exit_stringで命名したユーザー・イグジットをコールします。

構文

```
PROCEDURE USER_EXIT
  (user_exit_string  VARCHAR2);

PROCEDURE USER_EXIT
  (user_exit_string  VARCHAR2,
   error_string      VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

user_exit_string	パラメータなど、ユーザーがコールするユーザー・イグジット名を指定します。
error_string	そのユーザー・イグジットが失敗したときにForm Builderに表示されるユーザー定義エラー・メッセージを指定します。

USER_EXITの例

```
/*
** Built-in:USER_EXIT
** Example:Invoke a 3GL program by name which has been
**          properly linked into your current Form Builder
**          executable.The user exit subprogram must parse
**          the string argument it is passed to decide what
**          functionality to perform.
*/
PROCEDURE Command_Robotic_Arm( cmd_string VARCHAR2 ) IS
BEGIN
  /*
  ** Call a C function 'RobotLnk' to initialize the
  ** communication card before sending the robot a message.
  */
  User_Exit('RobotLnk INITIALIZE Unit=6,CmdToFollow=1');
  IF NOT Form_Success THEN
    Message('Failed to initialize Robot 6');
    RAISE Form_Trigger_Failure;
  END IF;
  /*
  ** Pass the string argument as a command to the robot
  */
```

```
  User_Exit('RobotLnk SEND Unit=6,Msg='||cmd_string );
  IF NOT Form_Success THEN
    Message('Command not understood by Robot 6');
    RAISE Form_Trigger_Failure;
  END IF;
  /*
  ** Close the robot's communication channel
  */
  User_Exit('RobotLnk DEACTIVATE Unit=6');
  IF NOT Form_Success THEN
    Message('Failed to Deactivate Robot');
    RAISE Form_Trigger_Failure;
  END IF;

  /*
  ** The user exit will deposit a timing code into the item
  ** called 'CONTROL.ROBOT_STATUS'.
  */
  Message('Command Successfully Completed by Robot 6'||
      ' in '||TO_VARCHAR2(:control.robot_timing)||
      ' seconds.');
END;
```

VALIDATEビルトイン

説明

VALIDATEを使用すると、Form Builderでは、指示された妥当性チェックの有効範囲について即時、妥当性チェック処理が強制的に実行されます。

構文

```
PROCEDURE VALIDATE
  (validation_scope  NUMBER);
```

ビルトイン・タイプ:

制限なしプロシージャ

問合せ入力モード

可

パラメータ

validation scope	次のいずれかの有効範囲を指定します。	
	DEFAULT_SCOPE	実行時のプラットフォームが判別したデフォルト有効範囲について標準妥当性チェックを実行します。 注意: ユーザーがSET_FORM_PROPERTY(VALIDATION UNIT)によって有効範囲を変更し、次にVALIDATE(DEFAULT_SCOPE)をコールすると、フォーム・モジュールで定義されたデフォルト有効範囲を上書きすることになります。この場合、Form Builderではデフォルト有効範囲では妥当性チェックが行われず、SET_FORM_PROPERTYによって定義された有効範囲で妥当性チェックが行われます。
	FORM_SCOPE	現フォームについて標準の妥当性チェックを実行します。
	BLOCK_SCOPE	現ブロックについて標準の妥当性チェックを実行します。
	RECORD_SCOPE	現レコードについて標準の妥当性チェックを実行します。
	ITEM_SCOPE	現項目について標準の妥当性チェックを実行します。

実行時の動作に関する注意

妥当性チェックの実行時に無効なフィールドが検出されると、カーソルはそのフィールドに移動せずに、元の位置に留まります。

VALIDATEの例

```
/*
** Built-in:VALIDATE
** Example:Deposits the primary key value, which the user
**         has typed, into a global variable, and then
**         validates the current block.
** Trigger:When-New-Item-Instance
*/
BEGIN
  IF :Emp.Empno IS NOT NULL THEN
   :Global.Employee_Id := :Emp.Empno;
    Validate(block_scope);
      IF NOT Form_Success THEN
        RAISE Form_Trigger_Failure;
      END IF;
    Execute_Query;
  END IF;
END;
```

VARPTR_TO_VARビルトイン

説明

バリアント・ポインタを通常のバリアントに変更します。

構文

```
FUNCTION VARPTR_TO_VAR
  (variant OLEVAR, vtype VT_TYPE)
RETURN changed OLEVAR;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

変換されたバリアント

パラメータ

variant	バリアントに変換されるOLEバリアント・ポインタ。
vtype	変換されるバリアントに与えられるOLE VT_TYPE。 これはオプションのパラメータです。指定されない場合のデフォルト値はVT_VARIANTです。

使用上の注意

- このファンクションは、バリアントからVT_BYREFの型指定を削除します。
- バリアントの型がVT_BYREFでなかった場合、バリアントはvtypeで指定された型へのアドレスを保持すると見なされ、参照解除されます。
- ポインタがNULLであったか、バリアントがVT_NULL型であった場合、VT_EMPTYが戻されます。

VAR_TO_TABLEビルトイン

説明

OLE配列バリアントを読み取り、PL/SQL表をそこから移入します。

構文

```
PROCEDURE VAR_TO_TABLE
  (var OLEVAR,
   target_table,
   arrspec VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

パラメータ

var	ソース配列であるOLEバリアント。
target_table	移入されるPL/SQL表。
arrspec	ソース配列のどの行、列または要素が使用されるかを示します。詳細は、OLE配列の指定子を参照してください。 これはオプションのパラメータです。指定されない場合、ソース配列内の全要素が使用されます。

使用上の注意

その他のデータ型上での同様の操作については、VAR_TO_<type>ファンクションを使用します。

VAR_TO_<type>ビルトイン

説明

OLEバリアントを読み取り、その値を等価のPL/SQL型に変換します。

CHAR型、NUMBER型およびOBJ型のそれぞれに対して、3つのバージョンのファンクションが存在します（型における値によって示されます）。

構文

```
FUNCTION VAR_TO_CHAR
  (var OLEVAR, arrspec VARCHAR2)
RETURN retval VARCHAR2;


FUNCTION VAR_TO_NUMBER
  (var OLEVAR, arrspec VARCHAR2)
RETURN retval NUMBER;
```

```
FUNCTION VAR_TO_OBJ
  (var OLEVAR, arrspec VARCHAR2)
RETURN retval OLEOBJ;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

その値が等価のPL/SQLタイプに変換されたバリアント。戻り値の型、どのバージョンのファンクションを選択したかによって異なることに注意してください。

パラメータ

var	読み取られるOLEバリアント。
arrspec	このパラメータは、OLEバリアントが配列の場合にのみ使用されます。このパラメータは、配列のどの要素を読み込んで、返すかを示します。 詳細は、OLE配列の指定子を参照してください。

使用上の注意

- 表を戻すには、VAR_TO_TABLEプロシージャを使用します。
- このファンクションのVAR_TO_OBJバージョンでは、ローカル（非永続）のオブジェクトが戻されます。

VAR_TO_VARPTRビルトイン

説明

既存のバリアントを指すOLEバリアントを作成します。

構文

```
FUNCTION VAR_TO_VARPTR
  (variant OLEVAR, vtype VT_TYPE)
RETURN newpointer OLEVAR;
```

ビルトイン・タイプ

制限なしファンクション

戻り値

作成されたバリアント

パラメータ

variant	指示対象となる既存のOLEバリアント。
vtype	作成されるOLEバリアントに割り当てられる型。 割り当て可能な型はVT_BYREF、VT_PTRおよびVT_NULLです。 これはオプションのパラメータです。指定されない場合のデフォルト値はVT_BYREFです。

使用上の注意

- 指示先のバリアントがVT_EMPTY型である場合、ファンクション内でのvtypeの指定に関係なく、常にVT_NULL型のポインタが作成されます。
- vtypeがVT_BYREFとして指定されるか、あるいはvtypeのデフォルトがVT_BYREFである場合、作成されるポインタはVT_BYREF型にソース・バリアントの型を加えたものになります。
- vtypeがその中にVT_BYREFを持たない場合、作成されるポインタはVT_PTR型のものになり、バリアント内部の内容を指します。

VBX.FIRE_EVENTビルトイン

説明

VBXコントロール用のイベントを呼び出します。

構文

```
PROCEDURE VBX.FIRE_EVENT
  (item_id         ITEM,
   event_name      VARCHAR2,
   paramlist_id    PARAMLIST);
PROCEDURE VBX.FIRE_EVENT
  (item_id         ITEM,
   event_name      VARCHAR2,
   paramlist_name  VARCHAR2);
PROCEDURE VBX.FIRE_EVENT
  (item_name       VARCHAR2,
   event_nam       VARCHAR2,
   paramlist_id  PARAMLIST);
PROCEDURE VBX.FIRE_EVENT
  (item_name       VARCHAR2,
```

```
  event_name      VARCHAR2,
  paramlist_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
event_name	VBXコントロールに送信されるVBXイベントの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
paramlist_id	パラメータ・リスト作成時に、Form Builderにより割り当てられる一意のIDを指定します。そのIDのデータ型は、PARAMLISTです。
paramlist_name	パラメータ・リスト・オブジェクトの定義時に与える名前。その名前のデータ型は、VARCHAR2文字列です。

VBX.FIRE_EVENTの制限事項

Microsoft Windowsにおいてのみ有効です。

VBX.FIRE_EVENTの例

```
/*
** Built-in:VBX.FIRE_EVENT
** Example:The VBX.FIRE_EVENT built-in triggers a SpinDown
**          event for the SpinButton VBX control.
** Trigger:When-Button-Pressed
*/
DECLARE
 ItemName VARCHAR2(40) := 'SPINBUTTON';
 PL_ID  PARAMLIST;
 PL_NAME  VARCHAR2(20) := 'EName';
BEGIN
 PL_ID := Get_Parameter_List(PL_NAME);
 IF id_null(PL_ID) THEN
  PL_ID := Create_Parameter_List(PL_NAME);
 END IF;
```

```
 VBX.FIRE_EVENT(ItemName,'SpinDown',PL_ID);
END;
```

VBX.GET_PROPERTYビルトイン

説明

VBXコントロールからプロパティの値を取得します。

構文

```
FUNCTION VBX.GET_PROPERTY
  (item_id     ITEM,
   property    VARCHAR2);
FUNCTION VBX.GET_PROPERTY
  (item_name   VARCHAR2,
   property    VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
property	VBXコントロール用のプロパティまたはプロパティ配列の要素を指定します。どのVBXコントロールについても、一連のVBXプロパティが存在します。VBXプロパティの例としては、幅および高さ、フォントサイズがあります。プロパティのデータ型は、VARCHAR2文字列です。

VBX.GET_PROPERTYの制限事項

Microsoft Windowsにおいてのみ有効です。

VBX.GET_PROPERTYの例

```
/*
** Built-in:VBX.GET_PROPERTY
** Example:Uses the VBX.GET_PROPERTY built-in to obtain the
**          CURRTAB property of the VBX item named TABCONTROL.
**          The property value of CURRTAB is returned to the
**          TabNumber variable and is used as input in the
**          user-defined Goto_Tab_Page subprogram.
** Trigger:When-Custom-Item-Event
*/
DECLARE
 TabEvent varchar2(80);
 TabNumber char;
BEGIN
 TabEvent := :system.custom_item_event;
 IF (UPPER(TabEvent) = 'CLICK') then
  TabNumber := VBX.Get_Property('TABCONTROL','CurrTab');
  Goto_Tab_Page(TO_NUMBER(TabNumber));
 END IF;
END;
```

VBX.GET_VALUE_PROPERTYビルトイン

説明

VBXコントロールのVBXコントロール値プロパティを取得します。

構文

```
FUNCTION VBX.GET_VALUE_PROPERTY
  (item_id  ITEM);
FUNCTION VBX.GET_VALUE_PROPERTY
  (item_name  VARCHAR2);
```

ビルトイン・タイプ

制限なしファンクション

戻り値

VARCHAR2

問合せ入力モード

可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。

VBX.GET_VALUE_PROPERTYの制限事項

Microsoft Windowsにおいてのみ有効です。

VBX.GET_VALUE_PROPERTYの例

```
/*
** Built-in:VBX.GET_VALUE_PROPERTY
** Example:Passes the VBX Control Value to the user-defined
**          Verify_Item_Value subprogram.Verify_Item_Value
**          ensures the display value is the expected value.
*/
DECLARE
 ItemName VARCHAR2(40) := 'SPINBUTTON';
 VBX_VAL_PROP VARCHAR2(40);
BEGIN
 VBX_VAL_PROP := VBX.Get_Value_Property(ItemName);
 Verify_Item_Value(VBX_VAL_PROP);
END;
```

VBX.INVOKE_METHODビルトイン

構文

```
VBX.INVOKE_METHOD(item_id, method_name, w, x, y, z);
PROCEDURE VBX.INVOKE_METHOD(item_name, method_name, w, x, y, z);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

説明

その項目に対して指定されたメソッドを起動します。そのメソッドに引数が必要な場合は、指定する必要があります。引数の提供順序は、そのVBXコントロールが求める順序で行います。各VBXコントロールに有効な方法、および必要な引数のリストは、そのVBXコントロールに付属するマニュアルで説明されています。

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
method_name	起動対象の方法名を指定します。その名前のデータ型は、VARCHAR2文字列です。
w, x, y, z	VBXコントロールに必要となる可能性のある任意指定の引数を指定します。これらの引数のデータ型は、VARCHAR2文字列です。

VBX.INVOKE_METHODの制限事項

Microsoft Windowsにおいてのみ有効です。

VBX.INVOKE_METHODの例

```
/*
** Built-in:VBX.INVOKE_METHOD_PROPERTY
** Example:Adds an entry to a combobox.The entry to
**          add to the combobox is the first optional argument.
**          The position where the entry appears is the second
**          optional argument.
*/
DECLARE
 ItemName VARCHAR2(40) := 'COMBOBOX';
BEGIN
 VBX.Invoke_Method(ItemName,'ADDITEM','blue','2');
END;
```

VBX.SET_PROPERTYビルトイン

説明

VBXコントロール用の指定プロパティを設定します。

構文

```
PROCEDURE VBX.SET_PROPERTY
  (item_id   ITEM,
   property  VARCHAR2,
   value     VARCHAR2);
PROCEDURE VBX.SET_PROPERTY
  (item_name  VARCHAR2,
   property   VARCHAR2,
   value      VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
property	VBXコントロール用のプロパティまたはプロパティ配列の要素を指定します。どのVBXコントロールについても、一連のVBXプロパティが存在します。VBXプロパティの例としては、幅および高さ、フォントサイズがあります。プロパティのデータ型は、VARCHAR2文字列です。
value	そのVBXプロパティに適用される値を指定します。値のデータ型は、VARCHAR2文字列です。

VBX.SET_PROPERTYの制限事項

Microsoft Windowsにおいてのみ有効です。

VBX.SET_PROPERTYの例

```
/*
** Built-in:VBX.SET_PROPERTY
** Example:Uses the VBX.SET_PROPERTY built-in to set the Index
**          property of the SpinButton VBX control.
** Trigger:When-Button-Pressed
*/
DECLARE
 ItemName VARCHAR2(40) := 'SPINBUTTON';
 VBX_VAL_PROP VARCHAR2(40);
```

```
 VBX_VAL  VARCHAR2(40);
BEGIN
 IF :System.Custom_Item_Event = 'SpinDown' THEN
  VBX_VAL_PROP := 'Index';
  VBX_VAL := '5';
  VBX.Set_Property(ItemName,VBX_VAL_PROP,VBX_VAL);
 END IF;
END;
```

VBX.SET_VALUE_PROPERTYビルトイン

説明

VBXコントロールのVBXコントロール値プロパティを設定します。

構文

```
PROCEDURE VBX.SET_VALUE_PROPERTY
  (item_id     ITEM,
   property    VARCHAR2);
PROCEDURE VBX.SET_VALUE_PROPERTY
  (item_name   VARCHAR2,
   property    VARCHAR2);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

item_id	Form Builderにより項目作成時に割り当てられた一意のIDを指定します。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。このIDのデータ型はITEMです。
item_name	設計時に作成されたオブジェクトの名前を指定します。その名前のデータ型は、VARCHAR2文字列です。
property	Form Builder VBXコントロール値プロパティのプロパティを指定します。どのVBXコントロールについても、一連のVBXプロパティが存在します。VBXプロパティの例としては、幅および高さ、フォントサイズがあります。プロパティのデータ型は、VARCHAR2文字列です。

VBX.SET_VALUE_PROPERTYの制限事項

Microsoft Windowsにおいてのみ有効です。

VBX.SET_VALUE_PROPERTYの例

```
/*
** Built-in:VBX.SET_VALUE_PROPERTY
** Example:Uses the VBX.SET_VALUE_PROPERTY built-in to set the
**          VBX Control Value property.
*/
DECLARE
 ItemName VARCHAR2(40) := 'SPINBUTTON';
 VBX_VAL_PROP VARCHAR2(40);
BEGIN
 IF :System.Custom_Item_Event = 'SpinDown' THEN
  VBX_VAL_PROP := 'Index';
  VBX.Set_Value_Property(ItemName,VBX_VAL_PROP);
 END IF;
END;
```

WEB.SHOW_DOCUMENTビルトイン

構文

```
SHOW_DOCUMENT(url, target);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モードに入る

可

説明

WebアプリケーションのURLおよびターゲット・ウィンドウを指定します。

パラメータ

url	**データ型はVARCHAR2です。ロードする文書のURL（Uniform Resource Locator）を指定します。**

target	データ型はVARCHAR2です。次のターゲットを1つ指定します。 _SELF 文書を、ソース文書と同じフレームかウィンドウにロードします。 _PARENT ターゲット文書を、ハイパーテキスト参照を含む親ウィンドウかフレームセットにロードします。参照がウィンドウかトップ・レベルのフレームにあると、target _selfと同じになります。 _TOP 文書を、ハイパーテキスト・リンクを含むウィンドウにロードし、現在そのウィンドウに表示されているフレームをすべて置換します。 _BLANK 文書を、名前の付いていないトップ・レベルの新規ウィンドウにロードします。

制限事項

Webから実行されたフォームの範囲内でのみ使用できます。

例

```
/*
** Built-in:WEB.SHOW_DOCUMENT
** Example:Display the specified URL in the target window.
*/
BEGIN
  Web.Show_Document('http://www.abc.com', '_self');
END;
```

WHERE_DISPLAYビルトイン

説明

Whereメニューのナビゲーション・オプションをトグル・オン、オフにします。全画面メニューでは、このWhereオプションは、メニュー階層におけるオペレータの現在位置についての情報が表示されます。

構文

```
PROCEDURE WHERE_DISPLAY;
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

なし

WHERE_DISPLAYの制限事項

WHERE_DISPLAYは、全画面メニューで使用する場合のみ有効です。

WRITE_IMAGE_FILEビルトイン

説明

Form Builderのイメージ項目のイメージを指定ファイルに書き込みます。

構文

```
WRITE_IMAGE_FILE
 (file_name            VARCHAR2,
  file_type            VARCHAR2,
  item_id              ITEM,
  compression_quality  NUMBER,
  image_depth          NUMBER);
PROCEDURE WRITE_IMAGE_FILE
  (file_name VARCHAR2,
   file_type VARCHAR2,
   item_name VARCHAR2,
   compression_quality NUMBER,
   image_depth NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

file_name	そのイメージを格納するファイル名。そのファイル名は、ご使用のオペレーティング・システムの要件に従っている必要があります。
file_type	イメージのファイル・タイプは、BMP、CALS、GIF、JFIF、JPEG、PICT、RAS、TIFFまたはTPICです。このパラメータはVARCHAR2引数をとります。
item_id	ユーザーによるイメージ項目作成時に、Form Builderによりそのイメージ項目に割り当てられた一意のIDです。FIND_ITEMビルトインを使用して、適切な型の変数にそのIDを戻します。データ型はITEMです。

item_name	ユーザーがイメージ項目定義時にそのイメージ項目に与えた名前です。データ型はVARCHAR2です。
compression_quality	Form Builderによりイメージがそのファイル（任意指定）に格納される際、そのイメージに適用する圧縮程度です。データ型はVARCHAR2です。有効な値はNO_COMPRESSION、MINIMIZE_COMPRESSION、LOW_COMPRESSION、MEDIUM_COMPRESSION、HIGH_COMPRESSIONまたはMAXIMIZE_COMPRESSIONです。
image_depth	Form Builderによりイメージがそのファイル（任意指定）に格納される際、そのイメージに適用する深度です。データ型はVARCHAR2です。有効な値は、ORIGINAL_DEPTH、MONOCHROME、GRAYSCALE、LUT（ルックアップ・テーブル）またはRGB（赤、緑、青）です。

WRITE_IMAGE_FILEの制限事項

- 指示されたファイル・タイプは、そのイメージの実際のファイル・タイプとの互換性が必要です。
- どのファイルについても当てはまることですが、そのイメージを既存ファイルに書き込むと、そのファイルの内容はそのイメージ項目の内容で上書きされます。
- ファイル・システムまたはデータベースからPCDファイルおよびPCXファイルを読み込むことはできますが、（WRITE_IMAGE_FILEを使用して）イメージ・ファイルをPCD形式やPCX形式でファイル・システムに書き込むことはできません。（ファイル・システムへのイメージの書込み時に制限付きファイル・タイプが使用された場合、Form Builderではイメージ・ファイルをデフォルトの TIFF形式に設定します）。
- Form Builderイメージ項目からJPEGファイルに書き込むと、本来の解像度が失われます。

WRITE_IMAGE_FILEの例

```
/* Built-in:WRITE_IMAGE_FILE
**
** Save the contents of an image item out to a file
** on the filesystem in a supported image format.
*/
BEGIN
  WRITE_IMAGE_FILE('output.tif',
                   'TIFF',
                   'emp.photo_image_data',
                   maximize_compression,
                   original_depth);
END;
```

WRITE_SOUND_FILEビルトイン

説明

指定したサウンド項目のサウンド・データを指定ファイルに書き込みます。

構文

```
PROCEDURE WRITE_SOUND_FILE(file_name     VARCHAR2,
                           file_type     VARCHAR2,
                           item_id       ITEM,
                           compression   NUMBER,
                           sound_quality NUMBER,
                           channels      NUMBER);
PROCEDURE WRITE_SOUND_FILE(file_name     VARCHAR2,
                           file_type     VARCHAR2,
                           item_name     VARCHAR2,
                           compression   NUMBER,
                           sound_quality NUMBER,
                           channels      NUMBER);
```

ビルトイン・タイプ

制限なしプロシージャ

問合せ入力モード

可

パラメータ

file_name	サウンド・データを書き込みたいフル･パス指定のファイル名。
file_type	そのサウンド･データ･ファイルのファイル･タイプ。有効な値は、AU、AIFF、AIFF-C、およびWAVEです。 **注意:** ファイル・タイプは任意指定ですが、すでにわかっていれば、パフォーマンス向上のために指定してください。フィルタ・タイプを省略すると、Form Builderはデフォルトのファイル・タイプとして、WAVE（Microsoft Windows）、AIFF-C（Macintosh）、またはAU（それ以外のもの）を適用します。
item_id	ユーザーがサウンド項目作成時に、Form Builderにより与えられた一意のID。
item_name	ユーザーがサウンド項目作成時に与えた名前。
compression	Form Builderによりデータがファイルに書き込まれる前にそのサウンド・データを圧縮するかどうかを指定します。有効値は、COMPRESSION_ONおよびCOMPRESSION_OFF、ORIGINAL_SETTING（そのデータのデフォルト圧縮設定を保持）です。圧縮は任意指定です。省略した場合のデフォルト値は、ORIGINAL_SETTINGです。

sound_quality	そのサウンド・データのサンプリング・レートおよび深度の質。指定できる値は、HIGHEST_SOUND_QUALITY、HIGH_SOUND_QUALITY、MEDIUM_SOUND_QUALITY、LOW_SOUND_QUALITY、LOWEST_SOUND_QUALITY、およびORIGINAL_QUALITYです。サウンドの質は任意指定です。省略した場合のデフォルト値はORIGINAL_QUALITYです。
channels	Form Builderによりサウンド・データがファイルに書き込まれる際、モノラルとして書き込むのか、ステレオとして書き込むのかを指定します。有効値は、MONOPHONICおよびSTEREOPHONIC、ORIGINAL_SETTING（そのデータのデフォルト・チャネル設定を保持）です。チャネルは任意指定です。省略した場合のデフォルト値は、ORIGINAL_SETTINGです。

WRITE_SOUND_FILEの制限事項

- PLAY_SOUNDおよびREAD_SOUND_FILE、WRITE_SOUND_FILEビルトインを使用してサウンド・データを再生またはファイルに保存、あるいはその両方を行うには、そのビルトインのコールを実行する前に、サウンド項目を作成し、その項目にフォーカスを配置してください。そのサウンド項目は設計時にサウンド項目制御によって示されますが、その制御は実行時、エンド・ユーザーに対して機能しません。したがって、各ユーザーが実行時にその制御を見ないように、サウンド項目を、キャンバス上の他のオブジェクトの背後に"隠す必要があります"。（項目にフォーカスを入れる場合、それをNULLキャンバスに割り当てたり、その表示プロパティを「いいえ」に設定することはできません）。たとえば、When-Button-Pressedトリガーから READ_SOUND_FILEビルトインおよびPLAY_SOUNDビルトインをコールするには、READ_SOUND_FILEおよびPLAY_SOUNDをコールする前にビルトイン・プロシージャGO_ITEMへのコールをトリガー・コードに組み込むことで、フォーカスを"非表示"サウンド項目に入れます。

トリガー

Delete-Procedureトリガー

説明

削除データ・ソースがストアド・プロシージャのときに、Form Builderによって自動的に作成されます。このトリガーは、削除操作が必要なときにコールされます。これは、デフォルトの削除操作を実行する代わりにシステムによってコールされるOnDeleteトリガーであると考えてください。このトリガーは変更しないでください。

問合せ入力モード

適用不可

失敗時

影響なし

ファンクション・キー・トリガー

説明

ファンクション・キー・トリガーは、個々のランタイム・ファンクション・キーに対応付けられています。ファンクション・キー・トリガーは、対応するファンクション・キーをオペレータが押したときにのみ起動されます。ファンクション・キー・トリガーで定義された動作に、ファンクション・キーが通常実行するデフォルトの処理が置き換わります。

次の表に、すべてのファンクション・キーとそれに対応するランタイム・ファンクション・キーを示します。

キー・トリガー	対応付けられたファンクション・キー
Key-CLRBLK	[ブロック消去]
Key-CLRFRM	[フォーム消去]
Key-CLRREC	[レコード消去]
Key-COMMIT	[アクセプト]
Key-CQUERY	[問合せ件数]
Key-CREREC	[レコード挿入]
Key-DELREC	[レコード削除]
Key-DOWN	[下へ]
Key-DUP-ITEM	[項目の複製]
Key-DUPREC	[レコードの複製]

Key-EDIT	[編集]
Key-ENTQRY	[問合せ入力]
Key-EXEQRY	[問合せ実行]
Key-EXIT	[終了]
Key-HELP	[ヘルプ]
Key-LISTVAL	[値リスト]
Key-MENU	[ブロック・メニュー]
Key-NXTBLK	[次のブロックへ]
Key-NXT-ITEM	[次の項目へ]
Key-NXTKEY	[次の主キーへ]
Key-NXTREC	[次のレコードへ]
Key-NXTSET	[次のレコード・セットへ]
Key-PRINT	[印刷]
Key-PRVBLK	[前のブロックへ]
Key-PRV-ITEM	[前の項目へ]
Key-PRVREC	[前のレコードへ]
Key-SCRDOWN	[下方へスクロール]
Key-SCRUP	[上方へスクロール]
Key-UP	[上へ]
Key-UPDREC	[レコード・ブロック]

ファンクション・キー・トリガーを持つすべてのランタイム・ファンクション・キーを再定義できるわけではないことに注意してください。特に、次の静的ファンクション・キーは、Form Builderによってではなく端末またはユーザー・インタフェース管理でしばしば実行されるため、決して再定義できません。

[項目消去]	[先頭行]	[左のスクロール]
[コピー]	[行挿入]	[右のスクロール]
[カット]	[最後行]	[検索]
[文字削除]	[左へ]	[選択]
[行削除]	[ペースト]	[キー表示]
[エラー表示]	[リフレッシュ]	[挿入/置換トグル]
[行の終わり]	[右へ]	[転送]

定義レベル

フォーム、ブロックまたは項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

失敗時

影響なし

問合せ入力モード

可

使用上の注意

次のキーで実行されるデフォルトの機能は、問合せ入力モードでは使用できません。

[ブロック消去]	**[下へ]**	**[次のレコードへ]**
[フォーム消去]	**[項目の複製]**	**[次のレコード・セットへ]**
[レコード消去]	**[レコードの複製]**	**[前のブロックへ]**
[アクセプト]	**[ブロック・メニュー]**	**[前のレコードへ]**
[レコード挿入]	**[次のブ_ロックへ]**	**[上へ]**
[レコード削除]	**[次の主キーへ]**	**[レコード・ロック]**

一般的な用途

ファンクション・キー・トリガーは、次のタスクを実行するときに使用します。

- ファンクション・キーを動的に使用不可にします。
- ファンクション・キーのデフォルト動作を置き換えます。
- ファンクション・キーを動的に再設定します。
- 1つのキーまたはキー・シーケンスで、複雑な機能または複数の機能を実行します。

ファンクション・キー・トリガーの制限事項

- Form Builderでは、オペレータがダイアログ・ボックスで[コミット/アクセプト]を押したときは、Key-Commitトリガーは無視されます。

Insert-Procedureトリガー

説明

挿入データ・ソースがストアド・プロシージャであるときに、Form Builderによって自動的に作成されます。このトリガーは、挿入操作が必要なときにコールされます。これは、デフォルトの挿入操作を実行する代わりにシステムによってコールされるOnInsertトリガーであると考えてください。

定義レベル

このトリガーは変更しないでください。

問合せ入力モード

適用不可

失敗時

影響なし

Key-Fnトリガー

説明

Key-Fnトリガーは、対応付けられたキーをオペレータが押したときに起動されます。

通常は、Key-FnトリガーをForm Builder操作を実行しない10個のキーまたはキー・シーケンスに付加できます。これらのキーはKey-F0～Key-F9と呼ばれます。これらのキーをキー・トリガーに付加する前に、ユーザーまたはデータベース管理者はOracle Terminalを使用して適切な機能にキーをマップする必要があります。

定義レベル

フォーム、ブロックまたは項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

- Key-Fnトリガーで、カスタム機能の追加ファンクション・キーを作成します。
- デフォルト・メニューの**「ヘルプ」→「キー」**メニューに示されているキー説明は常に、そのキーに対して定義されているフォーム・レベルのトリガーに対してのものです。同じくそのキーに対して定義されている下位のトリガー（ブロック・レベルのトリガーなど）が存在する場合、下位のほうのトリガーの説明は、フォーカスが下位レベル（ブロックなど）にあり、かつ[キー表示]が押されているときに表示されますが、デフォルト・メニューの**「ヘルプ」→「キー」**メニューには表示されません。
- 特定のオペレーティング・システム上では、すべてのキーを再マッピングできるとは限りません。たとえば、Microsoft Windowsオペレーティング・システムでは、[F1]が押されると常にWindowsのヘルプ・システムが表示され、[Alt]+[F4]が押されるとアプリケーション・ウィンドウが閉じられようとします。

Key-Fnトリガーの制限事項

Form Builderでは、「編集」モードのKey-Fnトリガーは無視されます。

Key-Othersトリガー

説明

Key-Othersトリガーは、対応付けられたキーをオペレータが押すと起動されます。

Key-Othersトリガーは、キーに対応付けられたキー・トリガーがあり、ファンクション・キー・トリガーが（どのレベルにも）現在定義されていないすべてのキーに対応付けられます。

Key-Othersトリガーは、ランタイムのファンクション・キーのデフォルト動作を上書きします（ただし、制限が適用されない場合）。しかしデフォルト動作を上書きしても、Form Builderでは、ファンクション・キーのデフォルト操作がキー一覧表示画面にまだ表示されています。

トリガーのタイプ

キー

定義レベル

フォーム、ブロックまたは項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

Key-Othersトリガーで、オペレータの操作を制限します。特に、Key-Othersトリガーは、次のようなタスクの実行に使用されます。

- 特定の状況で、関係のないキーをすべて使用不可にします
- オペレータが任意のキーを押したとき、ある特定の動作を実行します

次のことにも注意してください。

- デフォルト・メニューの**「ヘルプ」→「キー」**メニューに示されているキー説明は常に、そのキーに対して定義されているフォーム・レベルのトリガーに対してのものです。同じくそのキーに対して定義されている下位のトリガー（ブロック・レベルのトリガーなど）が存在する場合、下位のほうのトリガーの説明は、フォーカスが下位レベル（ブロックなど）にあり、かつ[キー表示]が押されているときに表示されますが、デフォルト・メニューの**「ヘルプ」→「キー」**メニューには表示されません。

Key-Othersトリガーの制限事項

Form Builderでは、次の条件の場合に、Key-Othersトリガーは無視されます。

- フォームが「問合せ入力」モードであり「問合せ入力モードで起動」プロパティが「いいえ」になっている場合
- 値リストまたは、キー一覧表示画面、ヘルプ画面、エラー画面が表示されている場合
- オペレータがランタイムのプロンプトに応答している場合
- オペレータが静的ファンクション・キーを押している場合

Lock-Procedureトリガー

説明

ロック・データ・ソースがストアド・プロシージャであるときに、Form Builderによって自動的に作成されます。このトリガーは、ロック操作が必要なときにコールされます。これは、デフォルトのロック操作を実行する代わりにシステムによってコールされるOnLockトリガーであると考えてください。このトリガーは変更しないでください。

問合せ入力モード

適用不可

失敗時

影響なし

On-Check-Delete-Masterトリガー

説明

Form Builderでは、ユーザーがマスター/ディテール・リレーションを定義して「レコード削除時の動作」プロパティを「非孤立」に設定したときに、このトリガーが自動的に作成されます。また、このトリガーは、マスター/ディテール・リレーションのマスター・ブロック内のレコードを削除しようとすると起動します。

定義レベル

フォームまたはブロック

有効コマンド

制限付きビルトインを含むすべてのコマンド

問合せ入力モード

不可

失敗時

現行のマスター・レコードの削除を防ぎます。

起動条件

マスター/ディテール調整

On-Check-Delete-Masterトリガーの例

次の例は、マスター/ディテール・リレーション用にデフォルトで生成される On-Check-Delete-Masterトリガーです。このトリガーは、配送合計が購入注文合計と等しくない場合に失敗します。

```
DECLARE
  the_sum NUMBER;
BEGIN
  SELECT SUM(dollar_amt)
    INTO the_sum
    FROM po_distribution
   WHERE po_number = :purchase_order.number;

/* Check for errors */
  IF the_sum <> :purchase_order.total THEN
    Message('PO Distributions do not reconcile.');
    RAISE Form_Trigger_Failure;
     ELSIF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  end if;
END;
```

On-Check-Uniqueトリガー

説明

コミット操作中、データベースにレコードを挿入または更新する前に、Form Builderで主キーの値が一意であることがチェックされると、On-Check-Uniqueトリガーが起動されます。このトリガーは、レコードが挿入または更新されるごとに1回起動されます。

レコードの一意性をチェックするデフォルト処理を置換します。ブロックのDeleteProcedureトリガープロパティが「はい」に設定されているとき、Form Builderは、デフォルトにより適切なSQL文を組み立て実行して、レコードの一意性をチェックし、現行のレコードの主キー値に一致する行を選択します。重複する行が見つかると、メッセージ「FRM-40600: レコードは既に挿入されています。」が表示されます。

挿入がマークされているレコードの場合、Form Builderでは常に主キー値の一意性がチェックされます。更新の場合には、「主キー」プロパティが「はい」に設定された項目が変更されると、主キー値の一意性がチェックされます。

定義レベル

フォーム、ブロック

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

このトリガーからデフォルト処理を実行するには、CHECK_RECORD_UNIQUENESSビルトインをコールします。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「レコードの一意性チェック」および「ポストおよびコミット・トランザクション」を参照してください。

On-Check-Uniqueトリガーの例

次の例では、問合せ中の現行のレコードがDEPT表にもう存在しないことを検証しています。

```
DECLARE
  CURSOR chk_unique IS SELECT 'x'
                       FROM dept
                      WHERE deptno = :dept.deptno;
  tmp VARCHAR2(1);
BEGIN
  OPEN chk_unique;
  FETCH chk_unique INTO tmp;
  CLOSE chk_unique;
  IF tmp IS NOT NULL THEN
    Message('This department already exists.');
    RAISE Form_Trigger_Failure;
     ELSIF form_fatal OR form_failure THEN
```

```
    raise form_trigger_failure;
  END IF;
END;
```

On-Clear-Detailsトリガー

説明

マスター/ディテール・リレーションのマスター・ブロックであるブロックに調整の原因となるイベントが発生したときに起動されます。調整の原因となるイベントとは、別のレコードをマスター・ブロック内の現行のレコードにするイベントです。

定義レベル

フォーム、ブロック

有効コマンド

制限付きビルトインを含むすべてのコマンド

問合せ入力モード

不可

使用上の注意

マスター/ディテール・ブロック・リレーションが定義されると、Form Builderで自動的にOn-Clear-Detailsトリガーが作成されます。

失敗時

調整の原因となる操作トリガーおよび予定の調整トリガーが異常終了します。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「マスター/ディテールのデータ整合」を参照してください。

On-Closeトリガー

説明

オペレータまたはアプリケーションが問合せをクローズさせたときに起動されます。デフォルトでは、問合せ基準で識別されたレコードがすべてフェッチされたとき、またはオペレータかアプリケーションが問合せを取り消したときに、Form Builderで問合せがクローズされます。

On-Closeトリガーにより、標準のForm Builderの問合せの「クローズ・カーソル」フェーズが補強されます。

定義レベル

フォーム

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- 特に、On-CloseトリガーをOn-SelectトリガーまたはOn-Fetchトリガーの使用後に使用すると、ファイルおよびカーソルがクローズされてメモリーが解放にされます。
- On-Closeトリガーは、ABORT_QUERYビルトインがOn-Selectトリガーからコールされたときに、自動的に起動されます。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ABORT_QUERY」および「問合せのクローズ」を参照してください。

On-Closeトリガーの例

次の例では、トランザクション・トリガーを使用したユーザー定義のデータ・アクセス・メソッドで使用されているメモリーが解放されます。

```
BEGIN
  IF NOT my_data source_open('DX110_DEPT') THEN
    my_datasource_close('DX110_DEPT');
     ELSIF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
END;
```

On-Column-Securityトリガー

説明

このトリガーは、「列セキュリティの強化」ブロック・プロパティを持つ各ブロックの列レベルのセキュリティ設定が「はい」にされているときに起動されます。

デフォルトでは、データベースに問合せて列のセキュリティが施行され、現行のフォーム・オペレータが更新権限を持つデータベースの列が判別されます。オペレータに更新権限がない列の場合、「更新可」項目プロパティが動的に「いいえ」に設定され、フォーム内の対応するデータベース項目が更新不可能になります。Form Builderでは、フォームの起動時に各ブロックを順番に処理しながらこの操作が実行されます。

定義レベル

フォーム、ブロック

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

このトリガーからデフォルト処理を実行するには、ENFORCE_COLUMN_SECURITYビルトインをコールします。

失敗時

影響なし

On-Column-Securityトリガーの例

次の例では、SUPERUSERロールが使用可能でない限り、現行のブロック内の給料および歩合テキスト項目を、使用不可および更新不可能に設定します。ユーザー定義のSUPERUSERロールを持つユーザーのみがこれらの番号フィールドを変更できます。

```
DECLARE
  itm_id Item;
  on_or_off NUMBER;
BEGIN
  IF NOT role_is_set('SUPERUSER') THEN
     on_or_off := PROPERTY_OFF;
  ELSE
    on_or_off := PROPERTY_ON;
  END IF;
  itm_id := Find_Item('Emp.Sal');
  Set_Item_Property(itm_id,ENABLED,on_or_off);
  Set_Item_Property(itm_id,UPDATEABLE,on_or_off);
  itm_id := Find_Item('Emp.Comm');
  Set_Item_Property(itm_id,ENABLED,on_or_off);
  Set_Item_Property(itm_id,UPDATEABLE,on_or_off);

IF form_fatal OR form_failure THEN
    raise form_trigger_failure;
END IF;
END;
```

On-Commitトリガー

説明

Form Builderで、トランザクションを終了するデータベース・コミット文が通常に発行されるときに起動されます。デフォルトでは、この操作は、更新および、挿入、削除のマークが付いているすべてのレコードがデータベースにポストされた後に発生します。

このトリガーは、「トランザクションのポストおよびコミット」プロセス中に、トランザクションを終了させるためにForm Builderで発行されたデフォルトのCOMMIT文を置き換えます。

定義レベル

フォーム

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- On-Commitトリガーは、通常のForm Builderのコミット処理の条件を変更し、Oracle以外のデータベースの特定のコミット要件に適合させるために使用されます。
- このトリガーからデフォルト処理を実行するには、COMMIT_FORMビルトインをコールします。

失敗時

ポストおよびコミット処理が中止されます。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

On-Commitトリガーの例

この例では、トランザクション制御がサポートされていないデータ・ソースに対して処理が実行される場合に、コミット操作を使用不可能にしています。アプリケーションがOracleに対して実行される場合は、コミット操作は正常に動作します。

```
BEGIN
  IF Get_Application_Property(DATA_SOURCE) = 'ORACLE' THEN
    Commit_Form;
     ELSIF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
/*
  ** Otherwise, no action is performed
  */
END;
```

On-Countトリガー

説明

Form Builderでデフォルトの問合せ件数のカウント処理が正常に実行されて、現行の問合せ基準に一致するデータベース内の行数が判断されたとき起動されます。On-Countトリガーの実行が終了すると、標準的な問合せヒット・メッセージ「FRM-40355:問合せによって<レコード件数>件のレコードが取り出されます。」が表示されます。

定義レベル

フォーム、ブロック

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

- On-Countトリガーは、Oracle以外のデータ・ソースで実行されるアプリケーション内の「問合せ件数のカウント」のデフォルト処理を置き換えます。
- このトリガーからデフォルト処理を実行するには、COUNT_QUERYビルトインをコールします。
- デフォルト処理を置き換えていると、「Query_Hits」ブロック・プロパティの値を設定して、問合せ基準に一致するOracle以外のデータ・ソース内のレコード数を指定できます。
- Form Builderでは、On-Countトリガーが「Query_Hits」ブロック・プロパティの値の設定に失敗した場合でも、問合せヒット・メッセージ（FRM-40355)が表示されます。その場合、メッセージには指定レコード数0が表示されます。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「COUNT_QUERYCOUNTQUE」を参照してください。

On-Countトリガーの例

この例では、ユーザー命名サブプログラムがコールされて現行の問合せ基準で検索されたレコード数がカウントされ、「Query_Hits」プロパティが適切に設定されます。

```
DECLARE
  j NUMBER;
BEGIN
  j := Recs_Returned('DEPT',Name_In('DEPT.DNAME'));
  Set_Block_Property('DEPT',QUERY_HITS,j);
END;
```

On-Deleteトリガー

説明

「トランザクションのポストおよびコミット」プロセスの間に起動し、トランザクションのポスト中に削除されたレコードを取り扱うForm Builderのデフォルト処理を置き換えます。特に、このトリガーはPre-Deleteトリガーの起動後およびPost-Deleteトリガーの起動前に起動し、指定された行の実際のデータベース削除を置き換えます。データベースからの削除のマークが付けられた行ごとに、このトリガーが1回起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- On-Deleteトリガーにより、トランザクションのポスト中に削除されたレコードを取り扱うForm Builderのデフォルト処理を置き換えます。
- このトリガーからデフォルトのForm Builderのデフォルト処理を実行するには、すなわちフォームまたはデータベースからレコードを削除するには、DELETE_RECORDビルトインのコールを組み込みます。

失敗時

最後のセーブポイントまでロールバックします。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

On-Deleteトリガーの例

この例では、実際にデータベースから従業員を削除せずに、Termination_Dateを設定して従業員表を更新します。

```
BEGIN
  UPDATE emp
     SET termination_date = SYSDATE
   WHERE empno = :Emp.Empno;
IF form_fatal OR form_failure THEN
   raise form_trigger_failure;
END IF;
END;
```

On-Dispatch-Eventトリガー

説明

このトリガーは、ActiveXコントロール・イベントの発生時にコールされます。このトリガーの内部からDISPATCH_EVENTビルトインをコールして、ディスパッチ・モードを制限付きまたは制限なしのどちらかに指定できます。ActiveXコントロール・イベントを使用した作業の詳細は、オンライン・ヘルプの「ActiveXコントロール・イベントへの応答」を参照してください。

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

失敗時

影響なし

On-Dispatch-Eventトリガーの例

```
/*
ON-DISPATCH-EVENT trigger
*/
BEGIN
```

```
    IF SYSTEM.CUSTOM_ITEM_EVENT = 4294966696 THEN
       /*when event occurs, allow it to apply to different items */.
       FORMS4W.DISPATCH_EVENT(RESTRICTED_ALLOWED);
    ELSE
       /*run the default, that does not allow applying any other item */
       FORMS4W.DISPATCH_EVENT(RESTRICTED_UNALLOWED);
    ENDIF;
IF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;

END;
```

On-Errorトリガー

説明

On-Errorトリガーは、Form Builderで正常にエラー・メッセージが表示されると起動されます。

置換

メッセージ行へのエラー・メッセージの書込み。

定義レベル

フォーム、ブロックまたは項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

- On-Errorトリガーは、次の目的に使用します。
- エラーを検出および修復します
- 標準エラー・メッセージをカスタム・メッセージと置き換えます。

On-Errorトリガーの ERROR_CODE、ERROR_TEXT、ERROR_TYPE、DBMS_ERROR_TEXT またはDBMS_ERROR_CODEのいずれかのビルトイン・ファンクションを使用して、特定のエラー条件を識別します。

- ほとんどの場合、On-Errorトリガーは、ブロックまたは項目ではなく、フォームに付加してください。コミット処理中のようなForm Builderの内部ナビゲーション実行中にエラーが発生すると、ブロックまたは項目レベルで特定のエラーの検出が困難なことがあります。

失敗時

影響なし

On-Errorトリガーの例

次の例では、特定のエラー・メッセージ・コードをチェックし、適切に対応します。

```
DECLARE
   lv_errcod  NUMBER      := ERROR_CODE;
   lv_errtyp  VARCHAR2(3)  := ERROR_TYPE;
   lv_errtxt  VARCHAR2(80) := ERROR_TEXT;
BEGIN
   IF (lv_errcod = 40nnn) THEN
   /*
   **  Perform some tasks here
   */
   ELSIF (lv_errcod = 40mmm) THEN
   /*
   **  More tasks here
   */
   ...
...
ELSIF (lv_errcod = 40zzz) THEN
   /*
   **  More tasks here
   */
ELSE
   Message(lv_errtyp||'-'||to_char(lv_errcod)||':'||lv_errtxt);
   RAISE Form_Trigger_Failure;
END IF;
END;
```

On-Fetchトリガー

説明

問合せが最初にオープンされると、最初のレコードがブロックにフェッチされたときにOn-Selectトリガーが起動され、そのすぐ後にこのOn-Fetchトリガーが起動されます。問合せがオープンしている間、行のセットをブロックにフェッチする必要があるごとに再度起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

On-Fetchトリガーを書いてデフォルトのフェッチ処理と置き換えたとき、トリガーは次のことを行う必要があります。

- 「Records_To_Fetch」プロパティの設定で指定されているように、Oracle以外のデータ・ソースから適切なレコード数を検索します。
- 現行のブロック内に問合せされたレコード件数分のレコードを作成します。
- ブロック内のレコードに検索データを移入します。
- CREATE_QUERIED_RECORDビルトイン・サブプログラムをコールすることによって、On-Fetchトリガーから問合せレコードを作成します。
- 問合せがオープンしている間、ブロック内に必要なレコードがある限りOn-Fetchトリガーを起動し続ける。この動作は、次の時点まで続きます。
- 問合せレコードがトリガーの1回の実行で作成されなくなるまで。レコードの作成に失敗すると、フェッチ終了を指示するシグナルがForm Builderに送られて、検索するレコードがもう存在しないことを示します。
- オペレータまたはプログラムによりABORT_QUERYビルトインがコールされて問合せがクローズされるまで。
- トリガー内でビルトイン例外状況FORM_TRIGGER_FAILUREが発生するまで。

このトリガーからForm Builderのデフォルト処理を実行するには、トリガーにFETCH_RECORDSビルトインのコールを組み込みます。

ABORT_QUERYビルトインをOn-Fetchトリガー内で使用しないでください。ABORT_QUERYは、On-Fetchトリガー内では無効であり、一貫性に欠けた結果が生じます。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「レコードのフェッチ」を参照してください。

On-Fetchトリガーの例

この例では、クライアント側のパッケージ・ファンクションがコールされて、パッケージ・カーソルから適切な行数が検索されます。

```
DECLARE
  j NUMBER := Get_Block_Property(blk_name, RECORDS_TO_FETCH);
  emprow emp%ROWTYPE;

BEGIN
  FOR ctr IN 1..j LOOP
    /*
    ** Try to get the next row.
    */
    EXIT WHEN NOT MyPackage.Get_Next_Row(emprow);
    Create_Queried_Record;
    :Emp.rowid := emprow.ROWID;
    :Emp.empno := emprow.EMPNO;
    :Emp.ename := emprow.ENAME;
       :
       :
  END LOOP;
    IF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
END;
```

On-Insertトリガー

説明

「トランザクションのポストおよびコミット」プロセス中、レコードの挿入時に起動されます。このトリガーは特に、Form Builderでデータベースに正常にレコードが挿入されるときに、Pre-Insertトリガーの起動後およびPost-Insertトリガーの起動前に起動されます。データベースへの挿入のマークが付けられている行ごとに、このトリガーが1回起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- On-Insertトリガーでは、トランザクションのポスト中にレコード挿入を処理するForm Builderのデフォルト処理が置き換わります。
- このトリガーからForm Builderのデフォルト処理を実行するには、トリガーにINSERT_RECORDビルトインのコールを組み込みます。

失敗時

On-Insertトリガーが失敗すると、Form Builderでは次のステップが実行されます。

- エラー位置の設定
- 最後のセーブポイントまでロールバック

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

On-Lockトリガー

説明

オペレータが項目のデータを変更するためにキーを押し、Form Builderで正常に行のロックが試みられたときに起動します。このトリガーは、キーの押下と変更データの表示の間に起動します。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- On-Lockトリガーは、行をロックするForm Builderのデフォルト処理を置き換えます。たとえば、単一ユーザー・システムで使用するアプリケーションに対してOn-Lockトリガーを使用すると、すべてのロック処理が迂回され、処理が速くなります。また、Open Gatewayによってではなく、Oracle以外のデータ・ソースに直接アクセスしているときにも、On-Lockトリガーを使用します。

- オペレータがデータ変更を試みた結果On-Lockトリガーが起動されるときは、オペレータがレコード内の項目を最初に変更しようとしたときのみです。同じレコードの項目に対するその後の変更には起動されません。つまり、このトリガーは、ロックされるすべての行ごとに1回のみ起動されます。

- このトリガーからForm Builderのデフォルト処理を実行するには、トリガーにLOCK_RECORDビルトインのコールを組み込みます。

- このトリガーで、更新不可能なビューのひな型となる表をロックします。

注意：特殊な状況では、On-LockトリガーにLOCK TABLE DML文を使用できます。しかし、これは他のユーザーを表から締め出してしまう可能性があるので、使用の際は注意が必要です。LOCK TABLEを使用する前に、『Oracle8 Serverアプリケーション開発者ガイド』を参照してください。

失敗時

On-Lockトリガーが失敗すると、ターゲット・レコードはロックされず、現行の項目に入力フォーカスが置かれます。何らかの理由で現行の項目に入力できない場合、Form Builderでは前のナビゲート可能な項目に入力フォーカスが移ります。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「行のロック」を参照してください。

On-Logonトリガー

説明

Form Builderで正常にログオン・シーケンスが開始されると、ログオンごとに1回起動されます。

定義レベル

フォーム

有効コマンド

制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- On-Logonトリガーでは、Oracle以外のデータ・ソースへのログオン・プロシージャのために使用されます。
- Pre-LogonトリガーおよびPost-Logonトリガーは、ログオン・プロシージャの一部として起動されます。
- データ・ソースを必要としないアプリケーションを作成するには、このトリガーにNULLコマンドを指定してデータ・ソースへの接続を迂回します。
- このトリガーからForm Builderのデフォルト処理を実行するには、トリガーにLOGONビルトインのコールを組み込みます。

失敗時

フォームの終了が試みられますが、Post-Logonトリガーは起動されません。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「LOGON」を参照してください。

On-Logoutトリガー

説明

Form BuilderとRDBMSで正常にログアウト・プロシージャが開始されたときに起動されます。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- On-Logoutトリガーは、OracleまたはOracle以外のデータ・ソースからのデフォルト・ログアウト処理を置き換えるときに使用します。
- このトリガーからForm Builderのデフォルト処理を実行するには、トリガーにLOGOUTビルトインのコールを組み込みます。
- ログアウト・トリガーの1つの中から特定のビルトインをコールしても、結果は未定義です。たとえば、Pre-Logoutは「フォームを出る」イベントの後で起動されるため、Pre-Logoutトリガーから COPYビルトインはコールできません。フォームはもうアクセスできないため、COPYビルトイン操作はできません。

失敗時

On-Logoutトリガーに例外状況が生じ、現行のランタイム・セッションが終了されると、Form Builderで、Post-Logoutのような他のトリガーは起動されません。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「LOGOUT」を参照してください。

On-Messageトリガー

説明

Form Builderで正常にメッセージが表示されたときに起動され、そのメッセージを差し替えます。

定義レベル

フォーム、ブロックまたは項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

On-Messageトリガーは次の目的に使用します

- 情報メッセージを検出し応答する場合。
- 標準情報メッセージをカスタム・メッセージと置き換える場合。
- 不適切なメッセージを除外する場合
- On-Messageトリガーの中のMESSAGE_CODE、MESSAGE_TEXTまたはMESSAGE_TYPEの各ビルトインを使用して、特定のメッセージ条件の発生を識別します。
- On-Messageトリガーを使用してメッセージを検出しても、メッセージ行にそれが表示されない場合は、GET_MESSAGEビルトインは値を戻しません。このトリガーから現行のメッセージを表示するには、メッセージを検出してそれを表示デバイスに明示的に書く必要があります。
- ほとんどの場合、On-Messageトリガーはブロックや項目ではなく、フォームに付加します。コミット処理中のようなForm Builderの内部ナビゲーション実行中にエラーが発生すると、ブロックまたは項目レベルで特定のエラーの検出が困難なことがあります。

失敗時

影響なし

On-Messageトリガーの例

次の例では、ユーザーにメッセージが表示され、継続または停止の選択警告が表示されて、エラー・メッセージに対応します。

```
DECLARE
  alert_button NUMBER;
  lv_errtype VARCHAR2(3) := MESSAGE_TYPE;
  lv_errcod NUMBER := MESSAGE_CODE;
  lv_errtxt VARCHAR2(80) := MESSAGE_TEXT;
BEGIN
  IF lv_errcod = 40350 THEN
    alert_button := Show_Alert('continue_alert');
    IF alert_button = ALERT_BUTTON1 THEN
      ...
    ELSE
      ...
    END IF;
  ELSE
    Message(lv_errtyp||'-'||to_char(lv_errcod)||':'||lv_errtxt);
    RAISE Form_Trigger_Failure;
  END IF;
    IF form_fatal OR form_failure THEN
     raise form_trigger_failure;
  END IF;
END;
```

On-Populate-Detailsトリガー

説明

マスター/ディテール・リレーションが定義されると、Form Builderで自動的にこのトリガーが作成されます。このトリガーは、Form Builderで通常どおりにマスター/ディテール・リレーションのディテール・ブロックにデータを入れる必要がある場合に起動します。

定義レベル

フォーム、ブロック

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

On-Populate-Detailsトリガーは、マスター/ディテール・リレーションを確立した後で問合せのデフォルトのデータ入力フェーズを置き換えるときに使用します。

On-Populate-Detailsトリガーは、On-Clear-Detailsトリガーが存在しない限り起動されません。デフォルトのマスター/ディテール機能を使用している場合、Form Builderで必要なトリガーが自動的に作成されます。しかし、ユーザー独自のマスター/ディテール論理を書いた場合には、NULL文のみを含む場合でもOn-Clear-Detailsトリガーが必要です。

「延期」プロパティが「いいえ」に設定されていると、マスターのディテール項目へレコードの挿入が即座に行われます。「いいえ」はデフォルト設定です。

「延期」プロパティが「はい」に設定されているときにこのトリガーが起動すると、ブロックに調整の必要があるというマークが付けられます。

ブロック調整をユーザーが管理する場合には、SET_BLOCK_PROPERTY（COORDINATION_STATUS）ビルトインをコールできます。

失敗時

フォームに矛盾が生じる場合があります。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「マスター/ディテールのデータ整合」を参照してください。

On-Rollbackトリガー

説明

Form Builderで正常にROLLBACK文を発行されたときに起動され、発行された最後のセーブポイントまでトランザクションがロールバックされます。

定義レベル

フォーム

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

On-Rollbackトリガーは、標準Form Builderロールバック処理を置き換えるときに使用します。

このトリガーからデフォルトのForm Builder処理を実行するには、トリガーにISSUE_ROLLBACKビルトインのコールを組み込む。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「CLEAR_FORM」、「ポストおよびコミット・トランザクション」および「フォームのロールバック」を参照してください。

On-Savepointトリガー

説明

Form Builderで正常にSavepoint文が発行されたときに起動されます。デフォルトでは、フォームの起動時および「ポストおよびコミット・トランザクション」処理の起動時にセーブポイントが発行されます。

定義レベル

フォーム

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

このトリガーからForm Builderのデフォルトを実行するには、トリガーにISSUE_SAVEPOINTビルトインのコールを組み込みます。

On-Savepointトリガーでは、「Savepoint_Name」アプリケーション・プロパティにより、On-Savepointトリガーが存在しない場合にデフォルトで発行される次のセーブポイントの名前が返されます。On-Rollbackトリガーでは、Savepoint_NameからForm Builderがロールバックするセーブポイントの名前が返されます。

「Savepoint Mode」フォーム・モジュール・プロパティを「PROPERTY_FALSE」に設定して、セーブポイントのデフォルト処理を抑止できます。「SAVEPOINT_MODE」が「PROPERTY_OFF」のときは、Form Builderではセーブポイントが発行されず、On-Savepointトリガーは起動されません。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「CALL_FORM」、「ポストおよびコミット・トランザクション」および「セーブポイント」を参照してください。

On-Selectトリガー

説明

Form Builderでカーソルのオープン、解析および問合せの実行フェーズが正常に実行されたときに起動され、現行の問合せ基準に一致するデータベースのレコードを識別します。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、PL/SQL、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

On-Selectトリガーは、データベース・カーソルをオープンして実行します。特に、Oracle以外のデータ・ソースからデータを検索するときには、このトリガーが使用されます。On-Selectトリガーは、On-Fetchトリガーと共に使用され、EXECUTE_QUERYビルトイン・サブプログラムに通常発生する処理を置き換えます。

このトリガーからForm Builderのデフォルト処理を実行するには、SELECT_RECORDSビルトインのコールを組み込みます。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「EXECUTE_QUERY」および「問合せのオープン」を参照してください。

On-Selectトリガーの例

次の例では、On-Selectトリガーを使用してユーザー・イグジット、「Query」ビルトイン・サブプログラムSELECT_RECORDSがコールされ、データベースに対する問合せが実行されます。

```
IF Get_Application_Property(DATASOURCE) = 'DB2' THEN
  User_Exit ( 'Query' );
  IF Form_Failure OR Form_Fatal THEN
    ABORT_QUERY;
  END IF;
ELSE
  /*
  **  Perform the default Form Builder task of opening the query.
  */
  Select_Records;
END IF;
```

On-Sequence-Numberトリガー

説明

Form Builderで通常どおりにデフォルトの項目値の順序番号を生成するデフォルト処理を実行するときに起動されます。このトリガーは、Form Builderとデータベースとの対話時に、データベースに定義されたSEQUENCEオブジェクトから次の値を得たときに発生する一連のデフォルトのイベントを置き換えます。

定義レベル

フォーム、ブロックまたは項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

SEQUENCEがデフォルト項目値として使用されるとき「レコード作成」イベントが発生すると、Form Builderではデータベースへの問合せが行われ、SEQUENCEから次の値が得られます。On-Sequence-Numberトリガーでこの機能を抑止または上書きします。

このトリガーからForm Builderのデフォルト処理を実行するには、トリガーにGENERATE_SEQUENCE_NUMBERビルトインのコールを組み込みます。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「順序番号の生成」を参照してください。

On-Updateトリガー

説明

「トランザクションのポストおよびコミット」プロセス中、レコードの更新時に起動されます。特に、Form Builderで正常にデータベース内のレコードが更新されたときに、Pre-Updateトリガーの起動後およびPost-Updateトリガーの起動前に起動されます。フォームの中で更新のマークが付いている行ごとに1回起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

On-Updateトリガーは、トランザクションのポスト中にレコードを更新するForm Builderのデフォルト処理を置き換えます。

このトリガーからForm Builderのデフォルト処理を実行するには、トリガーにUPDATE_RECORDビルトインのコールを組み込みます。

失敗時

On-Updateトリガーが失敗すると、Form Builderでは次のステップが実行されます。

■ エラー位置の設定

■ 最後に発行されたセーブポイントまでロールバック

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Post-Blockトリガー

説明

「ブロックを出る」プロセスの間、フォーカスが現在のブロックを離れたときに起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Blockトリガーは、ブロックの現行のレコード、すなわち「ブロックを出る」イベントが発生したときに入力フォーカスのあったレコードを検証します。

ユーザーがこの条件に基づくブロックから出るのを防ぐには、このトリガーを使用して条件をテストします。

失敗時

フォームにナビゲーション単位を作成させるときにこのトリガーが失敗すると、Form Builderで特定のブロックまたはレコード、項目へのターゲットが設定されます。これに失敗すると、Form Builderによりターゲットの位置にカーソルが置かれます。しかし、ターゲットが現行の単位の外にある場合またはオペレータがプロセスの終了を指示した場合には、フォームは中止されます。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ブロックを出る」を参照してください。

Post-Blockトリガーの制限

Post-Blockトリガーは、「有効単位」フォーム・モジュール・プロパティが「フォーム」に設定されているときは起動されません。

Post-Changeトリガー

説明

次の条件のいずれかで起動されます。

- 「項目の妥当性チェック」プロセスで、項目が「変更」とマークされNULLではないと判断されたとき。
- オペレータが値リストから選択して項目に値を返し、その項目がNULLではないとき。
- Form BuilderでNULLでない値が項目にフェッチされたとき。この場合、When-Validate-Itemトリガーは起動されません。この状況を回避し、Post-Changeトリガーを効率的に除去するには、When-Validate-ItemトリガーにPost-Queryトリガーを含める必要があります。後述の「使用上の注意」を参照してください。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Changeトリガーは、Form Builderの以前のバージョンと互換性のある場合にのみ含まれます。新しいアプリケーションでの使用はお薦めしません。

Post-Queryトリガーには、Post-Changeトリガーの制限はありません。Post-Queryを使用して、フェッチしたデータベースの値を変更できます。このような変更を指定した場合、Form Builderで、対応する項目およびレコードが変更とマークされます。

失敗時

ナビゲーションにより開始された妥当性チェックの一部として起動された場合、ナビゲーションは失敗し、フォーカスは元の項目のままです。

フォーム内にV2-スタイル・トリガーがあり、フェッチされた値をForm Builderでレコードに挿入する場合、次の制限が適用されます。

- Form Builderでは、レコード内のデータベース項目の値を変更する試みはすべて無視されます。

- Form Builderでは、レコード内のデータベース項目に影響するのみのSELECT文は実行されません。
- Form Builderでは、INTO句を持たないSELECT文は実行されません。
- Form Builderにより、V2-スタイル・トリガー・ステップのSELECT文が実行されない場合、「逆戻りコード」トリガー・ステップ・プロパティが設定されているときでも、トリガー・ステップは正常終了したステップであるかのように扱われます。

フェッチ処理中には、PL/SQLトリガーとして定義されたPost-Changeトリガーでは、これらの制限は実行されません。トリガー・スタイルにかかわらず、レコードのフェッチ中は、Form Builderで妥当性チェックは実行されませんが、各項目ごとにPost-Changeトリガーを起動した後で、レコードおよび項目が有効とマークされます。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「項目の妥当性チェック」および「レコードのフェッチ」を参照してください。

Post-Changeトリガーの制限

Form Builderで検証されている項目の値を変更するPost-Changeトリガーを書き込むことができます。妥当性チェックが正常終了すると、Form Builderは変更された項目に有効のマークを付けるので、変更した項目をもう一度妥当性チェックすることはありません。この動作は妥当性チェックのループを避けるために必要ですが、ユーザーはデータベースに無効な値もコミットできてしまいます。

Post-Database-Commitトリガー

説明

このトリガーは、「トランザクションのポストおよびコミット」プロセスの間、データベースのコミットが発生した後に起動されます。Post-Forms-Commitトリガーは、データベースに挿入、更新、削除がポストされた後で起動されますが、トランザクションがコミットを発行して終了する前であることに注意してください。Post-Database-Commitトリガーは、Form Builderでコミットが発行されてトランザクションが終了した後に起動されます。

定義レベル

フォーム

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Database-Commitトリガーは、データベース・コミットが発生したときにある動作を実行するために使用します。

失敗時

このトリガーが失敗した時点では、Form Builderではすでに正常終了のロールバック操作を失敗の応答として開始できる場所を過ぎてしまっているため、ロールバックしません。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザション」を参照してください。

Post-Database-Commitトリガーの例

```
/*
** FUNCTION recs_posted_and_not_committed
**  RETURN BOOLEAN IS
** BEGIN
**  Default_Value('TRUE','Global.Did_DB_Commit');
**  RETURN (:System.Form_Status = 'QUERY'
**        AND :Global.Did_DB_Commit = 'FALSE');
** END;
*/
BEGIN
  :Global.Did_DB_Commit := 'FALSE';
END;
```

Post-Deleteトリガー

説明

「トランザクションのポストおよびコミット」プロセスの間、行が削除された後に起動されます。コミット・プロセス中に行がデータベースから削除されるごとに1回起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Deleteトリガーは、トランザクションを監査するために使用します。

失敗時

Post-Deleteトリガーが失敗すると、Form Builderでは次のステップが実行されます。

■ エラー位置の設定

■ 最後のセーブポイントまでロールバック

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Post-Formトリガー

説明

「フォームを出る」プロセス中、フォームが終了されると起動されます。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Formトリガーは、次のタスクに使用します。

■ 終了前にフォームをクリーンアップします。たとえば、Post-Formトリガーを使用して、フォームで今後必要としないグローバル変数を消去します。

■ フォームの終了時にオペレータにメッセージを表示します。

このトリガーは、フォームが異常終了したとき、たとえばフォームの妥当性チェックに失敗したときは起動されません。

失敗時

処理が停止します。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「フォームを出る」を参照してください。

Post-Forms-Commitトリガー

説明

「トランザクションのポストおよびコミット」プロセス中に起動されます。挿入または更新、削除とマークされたレコードがフォーム内にある場合、Post-Forms-Commitトリガーは、これらの変更がデータベースに書き込まれた後、Form Builderでデータベース・コミットが発行されてトランザクションが終了する前に起動されます。

フォームに挿入または更新、削除とマークされたレコードがないときに、オペレータが、またはアプリケーションでコミットを開始した場合、Form BuilderではPost-Forms-Commitトリガーがすぐに起動され、データベースには変更がポストされません。

定義レベル

フォーム

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Forms-Commitトリガーは、データベース・コミットが発生しそうなときに、監査証跡のような処理を実行するために使用します。

失敗時

ポストおよびコミット・プロセスが中止されます。Form Builderによりロールバックが発行され、内部のセーブポイント・カウンタが減分されます。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Post-Forms-Commitトリガーの例

この例では、Post-Database-Commitトリガーとともに使用し、ポストされてもコミットされていないレコードがないか検出されます。

```
/*
** FUNCTION recs_posted_and_not_committed
** RETURN BOOLEAN IS
** BEGIN
**    Default_Value('TRUE','Global.Did_DB_Commit');
**    RETURN (:System.Form_Status = 'QUERY'
**       AND :Global.Did_DB_Commit = 'FALSE');
** END;
*/
BEGIN
  :Global.Did_DB_Commit := 'FALSE';
END;
```

Post-Insertトリガー

説明

「トランザクションのポストおよびコミット」プロセス中、列が挿入されたすぐ後に起動されます。コミット・プロセス中、データベースにレコードが挿入されるごとに1回起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Insertトリガーは、トランザクションを監査するために使用します。

失敗時

Post-Insertトリガーが失敗すると、Form Builderでは次のステップが実行されます。

■ エラー位置の設定

■ 最後のセーブポイントまでロールバック

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Post-Logonトリガー

説明

次のいずれかのイベントの後に起動されます。

■ Form Builderのデフォルトのログオン処理が正常終了したとき

■ On-Logonトリガーの実行が正常終了したとき

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Logonトリガーは、フォームごとではなくアプリケーション・レベルで開始する特殊な変数のカスタム環境を設定するようなタスクに、イベント・ポイントを用意するために使用します。このトリガーから特殊なグローバル変数を開始すると実行できます。

失敗時

条件により異なります。

- トリガーが最初のログオン・プロセス中に失敗した場合、Form Builderによりフォームが終了され、オペレーティング・システムに戻ります。
- ログオンを正常終了した後にトリガーの起動が失敗した場合、Form Builderでビルトイン例外状況FORM_TRIGGER_FAILUREが発生します。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「LOGON」を参照してください。

Post-Logonトリガーの例

この例では、ユーザー・イグジットがコールされ、セキュリティー上の理由からデータベースの外部にあるファイル・システムの暗号化監査証跡ファイルに、ユーザー名と時間が記録されます。

```
BEGIN
  User_Exit('LogCrypt '||
           USER||' '  ||
           TO_CHAR(SYSDATE,'YYYYMMDDHH24MISS'));
END;
```

Post-Logoutトリガー

説明

次のいずれかのイベントの後に起動されます。

- Form BuilderがORACLEからのログアウトを正常に終了したとき
- On-Logoutトリガーの実行が正常終了したとき

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Logoutトリガーは、RDBMSまたはその他のデータ・ソースを必要としない、または影響しないForm Builderのアプリケーション上のタスクを、監査または実行するために使用します。

ログアウト・トリガーの1つの中から特定のビルトインをコールしても、結果は未定義です。たとえば、Pre-Logoutトリガーは「フォームを出る」イベントの後に起動されるため、COPYビルトインはPre-Logoutトリガーからコールできません。フォームはもうアクセスできないため、COPYビルトイン操作はできません。

失敗時

フォームを出るときにこのトリガーが失敗しても、影響はありません。

このトリガーが失敗し、ユーザーがトリガーからLOGOUTビルトインのコールを開始した場合、FORM_FAILUREビルトインの戻り値は「TRUE」に設定されます。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「LOGOUT」を参照してください。

Post-Queryトリガー

説明

ブロック内で問合せがオープンしているとき、Post-QueryトリガーはForm Builderでブロックにレコードがフェッチされるごとに起動されます。このトリガーは、レコードがレコードのブロック・リストに置かれるごとに起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Queryトリガーは、次のタスクに使用します。

- 他のブロックに制御項目を挿入します。
- 問合せで検索されたレコードに関する統計を計算します。
- 実行合計を計算します。
- Post-Queryトリガーを使用して制御項目に実表以外の値を返すとき、各レコードは変更とマークされ、デフォルトでWhen-Validate-Itemトリガーが起動されます。Post-Queryトリガー内で、各レコードの「Status」プロパティを明示的に「QUERY」に設定することにより、When-Validate-Itemトリガーの実行を回避できます。プログラムによってレコード状況を設定するには、SET_RECORD_PROPERTYビルトインを使用します。

失敗時

Form Builderではブロックからレコードがフラッシュされて、データベースから次のレコードをフェッチしようとします。データベースに他のレコードが存在しなくなると、Form Builderでは問合せがクローズされ、オペレータの次の操作を待ちます。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「レコードのフェッチ」を参照してください。

Post-Queryトリガーの例

この例では、現行のブロック内のデータベース以外の項目に表示するため、コード・フィールドの説明を検索します。

```
DECLARE

  CURSOR lookup_payplan IS SELECT Payplan_Desc
                           FROM Payplan
                          WHERE Payplan_Id =
                                :Employee.Payplan_Id;

  CURSOR lookup_area    IS SELECT Area_Name
                           FROM Zip_Code
                           WHERE Zip = :Employee.Zip;
```

```
BEGIN
  /*
  ** Lookup the Payment Plan Description given the
  ** Payplan_Id in the Employee Record just fetched.
  ** Use Explicit Cursor for highest efficiency.
  */
  OPEN lookup_payplan;
  FETCH lookup_payplan INTO :Employee.Payplan_Desc_Nondb;
  CLOSE lookup_payplan;

  /*
  ** Lookup Area Descript given the Zipcode in
  ** the Employee Record just fetched.Use Explicit
  ** Cursor for highest efficiency.
  */
  OPEN lookup_area;
  FETCH lookup_area INTO :Employee.Area_Desc_Nondb;
  CLOSE lookup_area;
END;
```

Post-Recordトリガー

説明

「レコードを出る」プロセス中に起動されます。特に、Post-Recordトリガーは、オペレータが、またはアプリケーションで入力フォーカスをあるレコードから別のレコードに移すときに起動されます。「レコードを出る」プロセスは、INSERT_RECORD、DELETE_RECORD、NEXT_RECORD、NEXT_BLOCK、CREATE_RECORD、PREVIOUS_BLOCKなどを含む多数の操作の結果、発生します。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

オペレータまたはアプリケーションが入力フォーカスをレコード間で移動するたびにある動作を実行するには、Post-Recordトリガーを使用します。たとえば、オペレータがレコード・セットをスクロールしたときに項目の可視属性を設定するには、このトリガーの内部にコードを配置します。

失敗時

入力フォーカスは現行のレコードのままです。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「レコードを出る」を参照してください。

Post-Recordトリガーの制限

Post-Recordトリガーは、「有効単位」フォーム・モジュール・プロパティで指定されるように、フォームが項目またはレコード単位の妥当性チェックを実行するときのみ、起動されます。

Post-Selectトリガー

説明

Post-Selectトリガーは、問合せ処理のデフォルト選択フェーズ後、またはOn-Selectトリガーの正常終了後に起動されます。また、フェッチ処理によってレコードが実際に検索される前に起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Selectトリガーは、問合せ基準に一致するレコード数に基づく動作など、問合せ処理の選択フェーズの発生に基づいた動作を実行するために使用します。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「問合せの実行」および「問合せのオープン」を参照してください。

Post-Text-Itemトリガー

説明

テキスト項目の「項目を出る」プロセス中に起動されます。特に、あるテキストから別のテキストに入力フォーカスが移ったときに、このトリガーが起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Post-Text-Itemトリガーは、項目値の計算または変更に使用します。

失敗時

ナビゲーションが失敗し、フォーカスは現テキスト項目のままです。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「項目を出る」を参照してください。

Post-Text-Itemトリガーの制限

Post-Text-Itemトリガーは、入力フォーカスがテキスト項目にあり、オペレータがマウスを使用して「マウス・ナビゲート」「マウス・ナビゲート」プロパティが「いいえ」になっているボタンまたは、チェックボックス、リスト項目、ラジオ・グループ項目をクリックするときには起動さ

れません。これらの項目で「マウス・ナビゲート」が「いいえ」のとき、マウスでこれらをクリックするのはナビゲーション以外のイベントであり、入力フォーカスは現行の項目のままです（この例ではテキスト項目）。

Post-Updateトリガー

説明

「トランザクションのポストおよびコミット」プロセス中､行が更新された後に起動されます。コミット・プロセス中に行がデータベース内で更新されるごとに1回起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文（DELETE、INSERT、UPDATE）、制限なしビルトイン

問合せ入力モード

不可

使用上の注意

Post-Updateトリガーは、トランザクションを監査するために使用します。

失敗時

Post-Updateトリガーが失敗すると、Form Builderでは次のステップが実行されます。

■ エラー位置の設定

■ 最後のセーブポイントまでロールバック

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Pre-Blockトリガー

説明

あるブロックから別のブロックへナビゲーションする際、「ブロックに入る」プロセス中に起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Blockトリガーは、次の目的に使用します。

■ ブロックへのアクセスを許可または禁止します。

■ 変数の値を設定します。

失敗時

ナビゲーションが失敗し、フォーカスは現項目のままです。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ブロックに入る」を参照してください。

Pre-Blockトリガーの制限

Pre-Blockトリガーは、「有効単位」フォーム・モジュール・プロパティで指定されるように、フォームが項目またはレコード、ブロック単位の妥当性チェックを実行するときのみ起動されます。

Pre-Commitトリガー

説明

「トランザクションのポストおよびコミット」プロセス中、Form Builderでレコードの変更が処理される前に起動されます。特に、フォームにポストまたはコミットを要する挿入、更新または削除があるとForm Builderで判断された後、変更のコミット前に起動されます。このトリガーは、コミットの試みがあってもフォーム内に変更されたレコードがないと判断された場合には起動されません。

定義レベル

フォーム

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Commitトリガーは、データベース・コミットが発生しそうな任意の時点で、特殊なロック要件の設定などの動作を常に実行するときに使用します。

失敗時

「ポストおよびコミット」プロセスが失敗します。データベースにレコードは書き込まれず、フォーカスは現行の項目のままです。

注意: Pre-CommitトリガーでDMLを実行してトリガーが失敗した場合、Form Builderでは自動ロールバックが実行されないため、手動ロールバックが必要になります。手動ロールバックをするために、GET_APPLICATION_PROPERTY（Savepoint_Name）を使用して、On-Savepointトリガーにセーブポイント名を保存しておきます。それから、ISSUE_ROLLBACK（Savepoint_Name）を使用してロールバックします。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Pre-Delete トリガー

説明

「トランザクションのポストおよびコミット」プロセス中、行が削除される前に起動されます。削除がマークされたレコードごとに1回ずつ起動されます。

注意: Form Builderでは、「レコード削除時の動作」プロパティが「カスケード」に設定されているマスター/ディテール・リレーションの場合には、Pre-Deleteトリガーが自動作成されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文（DELETE、INSERT、UPDATEなど）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Deleteトリガーは、マスター・レコードのディテール・レコードを削除するために使用します。

Pre-Deleteを使用すると、まだ存在するディテール・レコードのマスター・レコードであるレコードの削除が回避できます。

失敗時

Post-Deleteトリガーが失敗すると、Form Builderでは次のステップが実行されます。

- エラー位置の設定
- 最後のセーブポイントまでロールバック

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Pre-Formトリガー

説明

フォームの起動時、「フォームに入る」イベント中に起動されます。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Formトリガーは、次のタスクに使用します。

■ シーケンスから一意の主キーを割り当てます。

■ フォームへのアクセスを制限します。

■ グローバル変数を初期化します。

失敗時

Form Builderで現行のフォームが終了され、他のトリガーは起動されません。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「フォームに入る」を参照してください。

Pre-Insertトリガー

説明

「トランザクションのポストおよびコミット」プロセス中に行が挿入される前に起動されます。挿入がマークされたレコードごとに1回ずつ起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Insertトリガーは、次のタスクに使用します。

■ 項目の値を変更します。

■ レコードの作成日を記録し、コミット前にレコードにそれを保存します。

失敗時

Pre-Insertトリガーが失敗すると、Form Builderでは次のステップが実行されます。

■ エラー位置の設定

■ 最後のセーブポイントまでロールバック

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Pre-Insertトリガーの例

この例では、シーケンス番号を基にした主キー・フィールドが割り当てられ、新しいシーケンス作成のフラッグが付いた監査表に行が書き込まれます。

```
DECLARE
  CURSOR next_ord IS SELECT orderid_seq.NEXTVAL FROM dual;
BEGIN

  /*
  ** Fetch the next sequence number from the
  ** explicit cursor directly into the item in
  ** the Order record.Could use SELECT...INTO,
  ** but explicit cursor is more efficient.
  */
```

```
  OPEN next_ord;
  FETCH next_ord INTO :Order.OrderId;
  CLOSE next_ord;

  /*
  ** Make sure we populated a new order id ok...
  */
  IF :Order.OrderId IS NULL THEN
    Message('Error Generating Next Order Id');
    RAISE Form_Trigger_Failure;
  END IF;

  /*
  ** Insert a row into the audit table
  */
  INSERT INTO ord_audit( orderid, operation, username, timestamp )
  VALUES ( :Order.OrderId,
          'New Order',
          USER,
          SYSDATE );

END;
```

Pre-Logonトリガー

説明

Form Builderでデータ・ソースへのログオンが開始される直前に起動されます。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Logonトリガーは、フォームの、特にOracle以外のデータ・ソースのログオン・プロシージャを準備するために使用します。

失敗時

失敗時の結果は、次のいずれの条件が適用されるかによって異なります。

- Form Builderがフォームに初めて入ってトリガーが失敗した場合は、フォームは正常に終了され、他のトリガーは起動されません。
- Form Builderでトリガー内からLOGONビルトインの実行が試みられたときにトリガーが失敗した場合、Form BuilderではFORM_TRIGGER_FAILURE例外状況が発生します。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「LOGON」を参照してください。

Pre-Logoutトリガー

説明

Form Builderでログアウト・プロシージャが開始される前に起動されます。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Logoutトリガーは、フォームの、特にOracle以外のデータ・ソースからログアウトを準備するために使用します。

ログアウト・トリガーの1つの中から特定のビルトインをコールしても、結果は未定義です。たとえば、Pre-Logoutは「フォームを出る」イベントの後で起動されるため、Pre-LogoutトリガーからCOPYビルトインをコールできません。フォームにはこの時点ですでにアクセスできないため、COPYビルトイン操作はできません。

失敗時

失敗時の結果は、次のいずれの条件が適用されるかによって異なります。

- Form Builderでフォームが終了されてトリガーが失敗した場合、フォームは正常に終了され、他のトリガーは起動されません。
- Form Builderによりトリガー内からLOGOUTビルトインの実行が試みられたときにトリガーが失敗した場合は、Form BuilderではFORM_TRIGGER_FAILURE例外状況が発生します。

Pre-Logoutトリガーに例外が生じた場合、Form BuilderではOn-LogoutおよびPost-Logoutのような他のトリガーは起動されません。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「LOGOUT」を参照してください。

Pre-Popup-Menuトリガー

説明

このトリガーは、ユーザーがポップアップ・メニューを表示させたときにコールされます（Microsoft Windows環境では、これはユーザーがマウスの右ボタンを押したときに発生します）。このトリガーに対して定義された動作は、ポップアップ・メニューが表示される前に実行されます。

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

ポップアップ・メニューが表示される前に、メニュー上のメニュー項目の有効/無効を切り換えるには、このトリガーを使用します。

失敗時

影響なし

Pre-Queryトリガー

説明

「問合せ実行」または「問合せ件数のカウント」の処理中、Form BuilderでSELECT文が作成および発行され、問合せ基準に一致する行が識別される直前に起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Queryトリガーは、どの行が問合せで識別されるか判断する条件を変更するために使用します。

失敗時

問合せは取り消されます。オペレータが、またはアプリケーションでフォームを「問合せ入力」モードにした場合、フォームは「問合せ入力」モードのままです。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「COUNT_QUERY」、「EXECUTE_QUERY」、「問合せのオープン」および「問合せの準備」を参照してください。

Pre-Queryトリガーの例

この例では、データベース・ブロック問合せの問合せ基準が検査または変更されます。

```
BEGIN
  /*
  ** Set the ORDER BY clause for the current block
  ** being queried, based on a radio group
  ** called 'Sort_Column' in a control block named
```

```
  ** 'Switches'.The Radio Group has three buttons
  ** with character values giving the names of
  ** three different columns in the table this
  ** block is based on:
  **
  **     SAL
  **     MGR,ENAME
  **     ENAME
  */
  Set_Block_Property('EMP',ORDER_BY, :Switches.Sort_Column);
  /*
  ** Make sure the user has given one of the two
  ** Columns which we have indexed in their search
  ** criteria, otherwise fail the query with a helpful
  ** message
  */
  IF :Employee.Ename IS NULL AND :Employee.Mgr IS NULL THEN
    Message('Supply Employee Name and/or Manager Id '||
           'for Query.');
    RAISE Form_Trigger_Failure;
  END IF;

  /*
  ** Change the default where clause to either show "Current
  ** Employees Only" or "Terminated Employees" based on the
  ** setting of a check box named 'Show_Term' in a control
  ** block named 'Switches'.
  */
  IF Check box_Checked('Switches.Show_Term') THEN
    Set_Block_Property('EMP',DEFAULT_WHERE,'TERM_DATE IS NOT NULL');
  ELSE
    Set_Block_Property('EMP',DEFAULT_WHERE,'TERM_DATE IS NULL');
  END IF;
END;
```

Pre-Recordトリガー

説明

「レコードに入る」プロセス中、別のレコードへのナビゲーションの際に起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Recordトリガーは、実行合計を記録するために使用します。

失敗時

ナビゲーションが失敗し、フォーカスは現行の項目のままです。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「レコードに入る」を参照してください。

Pre-Recordトリガーの制限

Pre-Recordトリガーは、「有効単位」フォーム・モジュール・プロパティで指定されるように、フォームが項目またはレコードの検証単位で実行される場合にのみ起動されます。

Pre-Recordトリガーの例

次のトリガーは、いくつかの動的条件とSYSTEM.RECORD_STATUSの状態が「NEW」と指定されている新規レコードに、ユーザーが入力することを防ぐときに使用します。

```
IF (( dynamic-condition)
   AND :System.Record_Status = 'NEW') THEN
   RAISE Form_Trigger_Failure;
END IF;
```

Pre-Selectトリガー

説明

発行されるSELECT文がForm Builderで作成された後で、まだ文が実際に実行される前、「問合せ実行」および「問合せ件数のカウント」プロセス中に起動されます。システム変数SYSTEM.LAST_QUERYの値を読むことで、Pre-Selectトリガー内でSELEST文を検査できます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Selectトリガーは、Oracle以外のデータ・ソースの問合せを実行する前の準備に使用します。

失敗時

影響はありません。現問合せでその表からフェッチされたレコードはありません。表が空であるか、問合せの検索基準に合うレコードがありません。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「EXECUTE_QUERY」、「問合せのオープン」および「問合せの準備」を参照してください。

Pre-Text-Itemトリガー

説明

「項目に入る」プロセス中、ある項目から別の項目へナビゲーションする際に起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Text-Itemトリガーは、次のようなタスクに使用します。

- 同じレコードに以前入力されていた他の項目に基づいて、複雑なデフォルト値を引き出します。
- 後で参照できるようにテキスト項目の現行値を記録し、その値をグローバル変数またはフォーム・パラメータ内に格納します。

失敗時

ナビゲーションが失敗し、フォーカスは現行の項目のままです。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「項目に入る」を参照してください。

Pre-Text-Itemトリガーの制限

Pre-Text-Itemトリガーは、「有効単位」フォーム・モジュール・プロパティで指定されるようにフォームが項目の検証単位で実行されるときのみ起動されます.

Pre-Updateトリガー

説明

「トランザクションのポストおよびコミット」プロセス中に行が更新される前に起動されます。更新がマークされたレコードごとに1回ずつ起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、DML文(DELETE、INSERT、UPDATE）、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

Pre-Updateトリガーは、トランザクションの監査に使用します。

失敗時

Pre-Updateトリガーが失敗すると、Form Builderでは次のステップが実行されます。

■ エラー位置の設定

■ 最後のセーブポイントまでロールバック

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「ポストおよびコミット・トランザクション」を参照してください。

Pre-Updateトリガーの例

次の例では、変更を行うタイムスタンプとユーザー名も含めた指定顧客の旧割引および新割引を示す監査表に行が書き込まれます。

```
DECLARE
  old_discount NUMBER;
  new_discount NUMBER := :Customer.Discount_Pct;
  oper_desc    VARCHAR2(80);
  CURSOR old_value IS SELECT discount_pct FROM customer
                   WHERE CustId = :Customer.CustId;
BEGIN
  /*
  ** Fetch the old value of discount percentage from the
  ** database by CustomerId.We need to do this since the
  ** value of :Customer.Discount_Pct will be the *new* value
  ** we're getting ready to commit and we want to record for
  ** posterity the old and new values.We could use
  ** SELECT...INTO but choose an explicit cursor for
  ** efficiency.
  */
  OPEN old_value;
  FETCH old_value INTO old_discount;
  CLOSE old_value;
```

```
  /*
  ** If the old and current values are different, then
  ** we need to write out an audit record
  */
  IF old_discount <> new_discount THEN
    /*
    ** Construct a string that shows the operation of
    ** Changing the old value to the new value. e.g.
    **
    **   'Changed Discount from 13.5% to 20%'
    */
    oper_desc := 'Changed Discount from '||
                 TO_CHAR(old_discount)||'% to '||
                 TO_CHAR(new_discount)||'%';

    /*
    ** Insert the audit record with timestamp and user
    */
    INSERT INTO cust_audit( custid, operation, username,
      timestamp )
    VALUES ( :Customer.CustId,
             oper_desc,
             USER,
             SYSDATE );
  END IF;
END;
```

Query-Procedureトリガー

説明

問合せデータ・ソースがストアド・プロシージャのときに、Form Builderによって自動的に作成されます。このトリガーは、問合せ操作が必要なときにコールされます。これは、デフォルトの問合せ操作を実行するかわりにシステムによってコールされるOn-Queryトリガーであると考えてください。このトリガーは変更しないでください。

問合せ入力モード

「使用上の注意」を参照

使用上の注意

問合せの構成時には任意の項目を使用できますが、それらの項目を問合せストアド・プロシージャに渡せるように、「問合せデータ・ソースの列」プロパティが設定されている必要があります。また、問合せストアド・プロシージャは、それらの値を使用してデータをフィルタリングする必要があります。これは、ユーザーが指定しない限り、問合せ入力モードに自動的に入らないことを意味します。

失敗時

影響なし

Update-Procedureトリガー

説明

更新データ・ソースがストアド・プロシージャのときに、Form Builderによって自動的に作成されます。このトリガーは、更新操作が必要なときにコールされます。これは、デフォルトの更新操作を実行するかわりにシステムによってコールされるOn-Updateトリガーであると考えてください。このトリガーは変更しないでください。

問合せ入力モード

適用不可

失敗時

影響なし

User-Namedトリガー

説明

User-Namedトリガーは、開発者によってフォーム内で定義されるトリガーです。User-Namedトリガーは、Form Builderのイベントに応答して自動的に起動するトリガーではなく、他のトリガーまたはユーザー命名のサブプログラムから明示的にコールする必要があります。同じ定義レベルで定義されるUser-Namedトリガーは、他と重複しない名前にする必要があります。

User-Namedトリガーを実行するには、次に示すようにEXECUTE_TRIGGERビルトイン・プロシージャをコールする必要があります。

```
Execute_Trigger('my_user_named_trigger');
```

定義レベル

フォームまたは、ブロック、項目

有効コマンド

User-Namedトリガーがコールされる親トリガーの中で有効なすべてのコマンド。

問合せ入力モード

不可

使用上の注意

User-Namedトリガーを使用して実行するタスクの大半は、ユーザー命名のPL/SQLサブプログラムを記述することによっても実行できます。

すべてのトリガーと同様に、User-Namedトリガーの有効範囲は、定義レベルおよびそれ以下です。1つ以上のUser-Namedトリガーが同じ名前を持っている場合、最も低いレベルで定義されているトリガーが優先されます。

User-Namedトリガーは、フォーム・レベルに定義するのが最も実用的です。

メニューのPL/SQLコマンドおよびユーザー命名サブプログラムからフォーム・モジュールに定義されているユーザー命名サブプログラムを実行するときに、User-Namedトリガーが作成されます（フォームに定義されているユーザー命名サブプログラムは、別のモジュールに定義されているメニューPL/SQLからは直接コールできません）。メニューPL/SQL内でEXECUTE_TRIGGERビルトインをコールして、User-Namedトリガーを実行します。このトリガーで今度は現行のフォームで定義されているユーザー命名サブプログラムがコールされます。

失敗時

FORM_FAILUREビルトインの戻り値を「TRUE」に設定します。User-Namedトリガーは常にEXECUTE_TRIGGERビルトインでコールされるため、ビルトイン・サブプログラムの結果テストと同じ方法、すなわちビルトイン・ファンクションFORM_FAILURE、FORM_SUCCESS、FORM_FATALなどでエラーをテストして、User-Namedトリガーの結果をテストします。

When-Button-Pressedトリガー

説明

マウス・ボタンをクリックしたりキーボードを使用したりして、オペレータがボタンを選択したときに起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Button-Pressedトリガーは、ナビゲーションの実行、テキスト項目値の計算、他の項目、ブロック、フォーム・レベル機能の実行に使用します。

失敗時

影響なし

When-Button-Pressedトリガーの例

この例では、フォームに変更があった場合にCOMMIT_FORMビルトインが実行されます。

```
BEGIN
  IF :System.Form_Status = 'CHANGED' THEN
    Commit_Form;
    /*
    ** If the Form_Status is not back to 'QUERY'
    ** following a commit, then the commit was
    ** not successful.
    */
    IF :System.Form_Status <> 'QUERY' THEN
      Message('Unable to commit order to database...');
      RAISE Form_Trigger_Failure;
    END IF;
  END IF;
END;
```

When-Checkbox-Changedトリガー

説明

マウス・ボタンをクリックしたりキーボードを使用したりして、オペレータがチェックボックスの状態を変更したときに起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Checkbox-Changedトリガーは、チェックボックスの状態によってタスクを開始するために使用されます。

オペレータがチェックボックスをクリックしたとき、その項目の内部値は、ナビゲーションが正常に終了するまで変化されません。したがって、When-Checkbox-Changedトリガーはチェックボックス項目の変更値を登録する最初のトリガーになります。When-Checkbox-Changedトリガーの前に起動されるすべてのナビゲーション・トリガーの場合、チェックボックス項目の値は、オペレータがナビゲートする前の値のまま保持されます。

失敗時

影響なし

When-Clear-Blockトリガー

説明

Form Builderで現行のブロックからデータが消去される直前に起動されます。

When-Clear-Blockトリガーは、CLEAR_FORMイベントの際に現行のブロックがForm Builderで消去されていると起動できないため注意してください。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

- When-Clear-Blockトリガーは、Form Builderで現行のブロックが消去されるごとに動作を実行します。たとえば、この条件が発生したらいつでも自動的にコミットさせる、という場合に使用します。
- When-Clear-Blockトリガーでは、現行のレコードが存在しないためSYSTEM.RECORD_STATUSの値は信頼できません。かわりにGET_RECORD_PROPERTYビルトインを使用して、レコードの状態を獲得します。GET_RECORD_PROPERTYビルトインでは特定レコードの参照が必要とされるため、その値は常に正確です。

失敗時

ブロックの消去には影響しません。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「CLEAR_BLOCK」、「COUNT_QUERY」、「ENTER_QUERY」および「問合せのオープン」を参照してください。

When-Create-Recordトリガー

説明

Form Builderで新規レコードが作成されたときに起動します。たとえば、オペレータがレコード作成キーを押したとき、あるいはスクロール中にセット内のレコードをナビゲートしたとき、このトリガーが起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

When-Create-Recordトリガーは、Form Builderで新規レコードの作成が試みられるごとに、動作を実行します。またこのトリガーは、設計時よりも実行時に指定する必要のある複合デフォルト値、計算済みのデフォルト値、データ駆動のデフォルト値などの設定にも役立ちます。

失敗時

新規レコードの作成が妨げられます。可能ならば、前の位置に戻ります。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「CREATE_RECORD」を参照してください。

When-Create-Recordトリガーの例

この例では、レコードに変更のマークを付けずにデータ駆動または計算済みのデフォルト値が割り当てられます。

```
DECLARE
  CURSOR ship_dflt IS SELECT val
                       FROM cust_pref
                      WHERE Custid = :Customer.Custid
                        AND pref   = 'SHIP';
BEGIN
  /*
  ** Default Invoice Due Date based on Customer's
  ** Net Days Allowed value from the Customer block.
  */
  :Invoice.Due_Date := SYSDATE + :Customer.Net_Days_Allowed;
  /*
  ** Default the shipping method based on this customers
  ** preference, stored in a preference table.We could
  ** use SELECT...INTO, but explicit cursor is more
  ** efficient.
  */
  OPEN ship_dflt;
```

```
  FETCH ship_dflt INTO :Invoice.Ship_Method;
  CLOSE ship_dflt;
END;
```

When-Custom-Item-Eventトリガー

説明

フォーム内でJavaBean、ActiveXカスタム・コンポーネント（32ビットWindowsの場合）またはVBXカスタム・コンポーネント（16ビットのMicrosoft Windows 3.xの場合）でイベントが発生される場合に常時起動されます。

定義レベル

フォーム、ブロック、項目

有効コマンド

制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Custom-Item-Eventトリガーを使用して、カスタム・コンポーネントの値の選択または変更に応答します。システム変数SYSTEM.CUSTOM_ITEM_EVENTには、発生したイベント名が大文字小文字を区別して格納されます。また、システム変数SYSTEM.CUSTOM_ITEM_EVENT_PARAMETERSには、カスタム・コントロールによって起動されたイベントの補足引数を含むパラメータ名が格納されます。

コントロール・イベント名は状況に応じて変化します。

失敗時

影響なし

When-Custom-Item-Eventトリガーの例

JavaBeansの例

これは、2つのJavaBeansを使用するフォームのWhen-Custom-Item-Eventトリガーにコールされるプロシージャの例です（コンテキストについては、完全例を参照してください）。

トリガーは、値の変更に呼応してJavaBeansの一方のカスタム・イベントのディスパッチから起動されます。

```
CustomEvent ce = new CustomEvent(mHandler, VALUECHANGED);
dispatchCustomEvent(ce);
```

フォームでは、このJavaBeanのBean領域の項目に連結されたWhen_Custom_Item_Eventトリガーがこのカスタム・イベントに呼応して自動的に起動されます。

トリガー・コードでは、次のプロシージャが実行されます。このプロシージャでは、System.Custom_Item_Event_Parametersにアクセスして、JavaBeanコンテナに設定された新規の値（新しいアニメーション率など）がピック・アップされることに注意してください。

この例では、次にSet_Custom_Propertyビルトインを使用して、もう一方のJavaBeanに値が渡されます。

```
PROCEDURE Slider_Event_Trap IS
BeanHdl         Item;
       BeanValListHdl   ParamList;
       paramType       Number;
       EventName       VarChar2(20);
       CurrentValue     Number(4);
       NewAnimationRate Number(4);
Begin
       -- Update data items and Display fields with current radius information
       BeanValListHdl :=
get_parameter_list(:SYSTEM.Custom_Item_Event_Parameters);
       EventName     := :SYSTEM.Custom_Item_Event;
       :event_name    := EventName;
       if (EventName = 'ValueChanged') then
          get_parameter_attr(BeanValListHdl,'Value',ParamType,
CurrentValue);
          NewAnimationRate := (300 - CurrentValue);
          :Animation_Rate  := NewAnimationRate;
set_custom_item_property('Juggler_Bean','SetAnimationRate',
NewAnimationRate);
       elsif (EventName = 'mouseReleased') then
          get_parameter_attr(BeanValListHdl,'Value',ParamType,
CurrentValue);
set_custom_item_property('Juggler_Bean','SetAnimationRate',
CurrentValue);
end if;
End;
```

VBXの例

これは、Form BuilderでWhen-Custom-Item-Eventトリガーが起動されたときにコールされるプロシージャの例です。

```
DECLARE
 TabEvent  varchar2(80);
 TabNumber Number;
BEGIN
 TabEvent := :system.custom_item_event;
 /*
 ** After detecting a Click event, identify the
 ** tab selected, and use the user-defined Goto_Tab_Page
 ** procedure to navigate to the selected page.
 */
 IF (UPPER(TabEvent) = 'CLICK') THEN
  TabNumber := VBX.Get_Property('TABCONTROL','CurrTab');
  Goto_Tab_Page(TabNumber);
 END IF;
END;
```

When-Database-Recordトリガー

説明

Form Builderでレコードに挿入または更新のマークが最初に付けられると起動します。すなわち、このトリガーは、挿入または更新のときに次のポストまたはコミットでレコードを処理する必要があるとForm Builderで判断されると、すぐに起動します。一般的には、オペレータがレコード内の最初の項目を変更し、その項目の外へナビゲートしようとしたときに起こります。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

When-Database-Recordトリガーは、レコードに挿入または更新が初めてマークされるたびに動作を実行します。

失敗時

影響なし

When-Form-Navigateトリガー

説明

ユーザーが別のロード済みフォームにフォーカスを移したときなど、フォーム間のナビゲーションが発生したときに起動します。

定義レベル

フォーム

有効コマンド

制限付きビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

When-Form-Navigateトリガーは、フォーム全体にわたるナビゲーションが、ウィンドウをアクティブおよび非アクティブにするイベントに依らずに実行されるときに、動作を実行します。

失敗時

影響なし

When-Form-Navigateトリガーの例

これは、Form BuilderでWhen-Form-Navigateトリガーが起動されたときにコールされるプロシージャの例です。

```
DECLARE
 win_id WINDOW := FIND_WINDOW('WINDOW12');
BEGIN
 if (GET_WINDOW_PROPERTY(win_id,WINDOW_STATE) = 'MAXIMIZE' THEN
```

```
  SET_WINDOW_PROPERTY(win_id,WINDOW_STATE,MINIMIZE);
 else
  SET_WINDOW_PROPERTY(win_id,WINDOW_STATE,MAXIMIZE);
 end if;
END;
```

When-Image-Activatedトリガー

説明

オペレータがマウスを次のように使用したとき起動されます。

■ イメージ項目をシングルクリックしたとき

■ イメージ項目をダブルクリックしたとき

ダブルクリックの場合、When-Image-Pressedトリガーも同時に起動することに注意してください。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

失敗時

影響なし

When-Image-Pressedトリガー

説明

オペレータがマウスを次のように使用したとき起動されます。

■ イメージ項目をシングルクリックしたとき

■ イメージ項目をダブルクリックしたとき

ダブルクリックの場合、When-Image-Activatedトリガーも同時に起動することに注意してください。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Image-Pressedトリガーは、オペレータがイメージ項目をクリックまたはダブルクリックしたときに動作を実行します。

失敗時

影響なし

When-List-Activatedトリガー

説明

TListとして表示されるリスト項目の要素をオペレータがダブルクリックすると起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-List-Activatedトリガーは、ドロップダウン・リストまたはコンボ・ボックス・スタイルのリスト項目の場合ではなく、TListスタイルのリスト項目の場合にのみ起動されます。リスト項目の表示スタイルは、「リスト形式」プロパティで決まります。

失敗時

影響なし

When-List-Changedトリガー

説明

エンド・ユーザーがリスト項目から別の要素を選択したとき、または現在選択されている要素を選択解除したときに起動されます。さらに、When-List-Changedトリガーがコンボ・ボックス・スタイルのリスト項目に付加されると、このトリガーはエンド・ユーザーが入力済みテキストを入力または変更するたびに起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

リストの値がエンド・ユーザーによって直接変更されたときに動作を開始するには、When-List-Changedトリガーを使用します。DUPLICATE_ITEMビルトインなどのプログラムによってリストの値が変更された場合、あるいはリストの値を変更するプロシージャの起動をエンド・ユーザーが促した場合には、When-List-Changedトリガーは起動しません。たとえばエンド・ユーザーが、DUPLICATE_ITEMビルトインにマッピングされたキーを使用して項目を複製しても、When-List-Changedトリガーは起動しません。

失敗時

影響なし

When-Mouse-Clickトリガー

説明

次のイベントの1つが発生すると、オペレータがマウスをクリックした後でこのトリガーが起動されます。

- フォームに付加されていて、フォームの中の任意のキャンバスまたは項目内でマウスをクリックしたとき
- ブロックに付加されていて、ブロックの中の任意の項目内でマウスをクリックしたとき
- 項目に付加されていて、項目内でマウスをクリックしたとき

When-Mouse-Clickトリガーが起動するには、次の3つのイベントが起こらなくてはなりません。

- マウスを押します。
- マウスを離します。
- マウスをクリックします。

これらのイベントに対応付けられるトリガーは、When-Mouse-Clickトリガーが起動する前に起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Mouse-Clickトリガーは、オペレータが項目またはキャンバス、あるいはその両方の中でマウスをクリックするたびに動作を実行します。

失敗時

影響なし

When-Mouse-DoubleClickトリガー

説明

次のイベントの1つが発生すると、オペレータがマウスをダブルクリックした後にこのトリガーが起動されます。

- フォームに付加されていて、フォームの中の任意のキャンバスまたは項目内でマウスをダブルクリックしたとき
- ブロックに付加されていて、ブロックの中の任意の項目内でマウスをダブルクリックしたとき
- 項目に付加されていて、項目内でマウスをダブルクリックしたとき

When-Mouse-DoubleClickトリガーが起動するには次の6つのイベントが起こらなくてはなりません。

- マウスを押します。
- マウスを離します。
- マウスをクリックします。
- マウスを押します。
- マウスを離します。
- マウスをダブルクリックします。

これらのイベントに対応付けられるトリガーは、When-Mouse-DoubleClickトリガーの起動前に起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Mouse-Clickトリガーは、オペレータが項目またはキャンバス、あるいはその両方の中でマウスをダブルクリックするたびに動作を実行します。

失敗時

影響なし

When-Mouse-DoubleClickトリガーの例

オペレータがマウスをクリックしたとき、アプリケーションには動作Aが必要で、ダブルクリックをしたときは動作Bが必要であるとします。たとえば、オペレータがマウスをクリックした場合は情報ウィンドウが表示され、オペレータがマウスをダブルクリックした場合はオンライン・ヘルプ・ウィンドウが表示されるようにします。

この例では、3つのトリガー（When-Mouse-Clickトリガー、When-Timer-Expiredトリガー、When-Mouse-DoubleClickトリガー）が使用されています。

```
/*
**  Trigger:When-Mouse-Click
**  Example:When the operator clicks the mouse, create a timer
**          that will expire within .5 seconds.
*/

DECLARE
   timer_id       TIMER;
   timer_duration  NUMBER(5) := 500;
BEGIN
   timer_id := Create_Timer('doubleclick_timer', timer_duration,
     NO_REPEAT);
END;

/*
**  Trigger:When-Timer-Expired
**  Example:When the timer expires display the online help
**          window if the operator has double-clicked the mouse
**          within .5 seconds, otherwise display the product
**          information window.
*/
BEGIN
   IF :Global.double_click_flag = 'TRUE' THEN
     Show_Window('online_help');
     :Global.double_click := 'FALSE';
   ELSE
```

```
    Show_Window('product_information');
  END IF;
END;

/*
**  Trigger:When-Mouse-DoubleClick
**  Example:If the operator double-clicks the mouse, set a
**          flag that indicates that a double-click event
**          occurred.
*/
BEGIN
  :Global.double_click_flag := 'TRUE';
END;
```

When-Mouse-Downトリガー

説明

次のイベントの1つが発生すると、オペレータがマウス・ボタンを押した後にこのトリガーが起動されます。

- フォームに付加されていて、フォームの中の任意のキャンバスまたは項目内でマウスを押したとき
- ブロックに付加されていて、ブロックの中の任意の項目内でマウスを押したとき
- 項目に付加されていて、項目内でマウスを押したとき

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Mouse-Downトリガーは、オペレータが項目またはキャンバス、あるいはその両方の中でマウスを押すごとに動作を実行します。

注意: 「マウスを押す」イベントの後には常に「マウスを離す」イベントが続きます。

失敗時

影響なし

When-Mouse-Downトリガーの制限

ウィンドウ・マネージャによっては、When-Mouse-Downトリガーの中のナビゲーション・コードが失敗することがあります。Microsoft Windows 95/NTの例では、オペレータがフィールド（Item_One）内でマウス・ボタンをクリックした場合、GO_ITEM（'item two'）をコールするWhen-Mouse-Downトリガーは失敗します。これは、When-Mouse-UpがItem_Twoで発生しているので、Windows 95/NTではItem_Twoではなく Item_Oneにフォーカスが戻るためです。

When-Mouse-Enterトリガー

説明

次のイベントの1つが発生すると、マウスが項目またはキャンバスに入ったときこのトリガーが起動されます。

- フォームに付加されていて、フォームの中の任意のキャンバスまたは項目にマウスが入ったとき
- ブロックに付加されていて、ブロックの中の任意の項目にマウスが入ったとき
- 項目に付加されていて、マウスが項目に入ったとき

定義レベル

フォーム、ブロックまたは項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Mouse-Enterトリガーは、マウスが項目またはキャンバスに入るごとに動作を実行します。

ウィンドウよりも大きなキャンバス上では、When-Mouse-Enterトリガーを使用しないでください。初期ウィンドウの下にあるキャンバス上のアイコン・ボタンおよびアイコン項目は選択できませ

ん。ユーザーはキャンバスをスクロールして項目を表示することができます。ただし、マウスがその領域に入るとただちに、トリガーが起動して、前ターゲットにフォーカスが戻るため、ユーザーはこの項目をクリックすることができません。

When-Mouse-Enterトリガーの「ツールチップ」プロパティを変更すると、表示前にツールチップが取り消されます。

When-Mouse-Enterトリガーからモーダル・ウィンドウをコールするときは注意してください。コールすることでモーダル・ウィンドウが不必要に表示されることがあります。

たとえば、マウスがCanvas_Oneに入るたびにWhen-Mouse-EnterトリガーがAlert_Oneを表示させるとします。また、アプリケーションにCanvas_OneおよびCanvas_Twoの2つのキャンバスが含まれるものとします。Canvas_OneおよびCanvas_Twoは、お互いに重複せず、画面に並んで表示されます。さらに、Alert_OneはCanvas_Twoの境界内に表示されるものとします。

最後に、マウスはCanvas_Oneに入ると、Alert_Oneを表示させるWhen-Mouse-Enterトリガーが起動されるものとします。

オペレータがメッセージ・ボックスを閉じたとき、続けてマウスでCanvas_Oneに入った場合、必要もないのにまたAlert_Oneが表示されます。さらに、オペレータがマウスをCanvas_Twoから移動させると、このイベントに関連するWhen-Mouse-Leaveトリガーが起動されます。このような動作は、不要なはずです。

失敗時

影響なし

When-Mouse-Leaveトリガー

説明

次のイベントの1つが発生すると､マウスが項目またはキャンバスを出た後でこのトリガーが起動されます。

- フォームに付加されていて、フォームの中の任意のキャンバスまたは項目からマウスが出たとき
- ブロックに付加されていて、ブロックの中の任意の項目からマウスが出たとき
- 項目に付加されていて、マウスが項目から出たとき

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Mouse-Leaveトリガーは、マウスが項目またはキャンバス、あるいはその両方を出るたびに動作を実行します。

失敗時

影響なし

When-Mouse-Moveトリガー

説明

次のイベントの1つが発生すると、マウスが移動するたびにこのトリガーが起動されます。

- フォームに付加されていて、フォームの中の任意のキャンバスまたは項目内部でマウスが移動したとき
- ブロックに付加されていて、ブロックの中の任意の項目内でマウスが移動したとき
- 項目に付加されていて、項目内でマウスが移動したとき

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Mouse-Moveトリガーを使用すると、オペレータがマウスを移動するたびに動作が実行されます。

When-Mouse-Moveトリガーが何回も起動される可能性があるため、パフォーマンスに影響が生じることがあります。

失敗時

影響なし

When-Mouse-Upトリガー

説明

次のイベントの1つが発生すると、オペレータがマウス・ボタンを押して離すたびに起動されます。

- フォームに付加されていて、フォームの中の任意のキャンバスまたは項目内で「マウスを離す」イベントを受け取ったとき
- ブロックに付加されていて、ブロックの中の任意の項目内で「マウスを離す」イベントを受け取ったとき
- 項目に付加されていて、項目内で「マウスを離す」イベントを受け取ったとき

When-Mouse-Upトリガーが起動するには次の2つのイベントが起こらなくてはなりません。

- マウスを押します。
- マウスを離すします。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Mouse-Upトリガーは、オペレータがマウスを押して放すたびに動作を実行します。

「マウスを離す」イベントは、常に「マウスを押す」イベントを受け取った項目に対応付けられます。たとえば、Item_Oneに付加されたWhen-Mouse-Upトリガーがあるとします。オペレー

タがItem_One上でマウスを押してからItem_Two上でマウスを放した場合、「マウスを離す」トリガーは、Item_Twoではなく Item_Oneに対して起動されます。

失敗時

影響なし

When-New-Block-Instanceトリガー

説明

別のブロック内の項目に入力フォーカスが移ったときに起動されます。特に、以前に入力フォーカスのあったブロックとは別のブロックに入力フォーカスが移ることをForm Builderで受け入れる準備ができたときに起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

When-New-Block-Instanceトリガーは、Form Builderで新しいブロックに移るたびに動作を実行します。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「入力のための戻り」を参照してください。

When-New-Form-Instanceトリガー

説明

フォームの起動時、最初にナビゲート可能なブロックで最初にナビゲート可能な項目が、Form Builderによってナビゲートされます。When-New-Form-Instanceトリガーは、ナビゲーションの初期順序に起動される任意のナビゲーション・トリガーが正常終了した後に起動されます。

このトリガーは、コール先フォームからコール側フォームに制御が戻ったときには起動されません。

マルチフォームのアプリケーションでは、あるフォームから別のフォームにフォーカスが変更されるときには起動されません。

定義レベル

フォーム

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「フォームの実行」を参照してください。

When-New-Form-Instanceトリガーの制限

1 FORMS_OLE.GET_INTERFACE_POINTERビルトインをWhen-New-Form-Instanceトリガーからコールすると、SYNCHRONIZEビルトインによってOLE項目またはActiveXコントロールを初期化するのでない限り、例外（ORA-305500）が発生します。

2 この時点ではForm BuilderでOLE項目が初期化されていないため、例外が発生します。新しいフォームがコールされると、そのフォームは画面上のデフォルトのx-y位置に表示されます。この位置が希望と異なる場合、x-y座標を変更できます。ただし、このWhen-New-Form-Instanceトリガーではその座標を変更できません（このトリガーの起動順が遅いためです）。座標を変更するには、Pre-Formトリガーを使用します。

When-New-Form-Instanceトリガーの例

例

この例では、動的イメージを表示するルーチンをコールし、画面クロックをリフレッシュするためのタイマーを始動してから、最初のブロックを問い合わせます。

```
BEGIN
  Populate_Dynamic_Boilerplate;
  Start_OnScreen_Clock_Timer;
  Go_Block('Primary_Ord_Info');

  /*
  ** Query the block without showing
  ** the working message.
  */
  :System.Suppress_Working := 'TRUE';
  Execute_Query;
  :System.Suppress_Working := 'FALSE';
END;
```

When-New-Item-Instanceトリガー

説明

入力フォーカスが項目に移ったときに起動されます。特に、項目のナビゲーション後、以前に入力フォーカスがあった項目とは別の項目への入力を、Form Builderで受け入れる準備ができたときに起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限付きビルトイン・サブプログラム、制限なしビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-New-Item-Instanceトリガーは、項目に入力フォーカスが移ったときに動作を実行します。When-New-Item-Instanceトリガーは、制限付き（ナビゲーションの）ビルトインをコールする場合に特に役立ちます。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「入力のための戻り」を参照してください。

When-New-Item-Instanceトリガーの制限

このトリガーを起動するための条件は、次の状況の下では適合されません。

- 入力の受入れを停止せずに、Form Builderで項目がナビゲートされる場合
- 入力フォーカスが警告ウィンドウ内のフィールドまたはForm Builderメニューの任意の部分に移される場合

When-New-Record-Instanceトリガー

説明

以前に入力フォーカスのあったレコードと違うレコード内の項目に入力フォーカスが移されるときに起動されます。特に、レコード内の項目のナビゲーション後、以前に入力フォーカスがあったレコードとは別のレコードへの入力を、Form Builderで受け入れる準備ができたときに起動されます。

Form Builderで新しいレコードにナビゲートすると起動されます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-New-Record-Instanceトリガーは、新しいレコードにForm Builderで移るたびに起動されます。たとえば、オペレータが「下へ」を押してレコード・セットをスクロールするとき、入力フォーカスが次のレコードに移るたびにForm Builderでこのトリガーが起動されます。すなわち、Form Builderでブロック内に新しいレコードに移るたびにこのトリガーが起動されます。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「入力のための戻り」を参照してください。

When-Radio-Changedトリガー

説明

マウス・ボタンをクリックするかキーボードを使用するかして、オペレータがラジオ・グループの別のラジオ・ボタンを選択するか、現行の選択ラジオボタンを選択解除するときに起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

When-Radio-Changedトリガーは、ラジオ・グループの状態によって異なる動作を実行します（ラジオ・グループ内でラジオボタンを選択解除するとラジオ・グループの値はNULLに設定されます。オペレータは「問合せ入力」モードでこの方法を使用し、問合せからラジオ・グループを除外します）。

オペレータがラジオ・グループ内の項目をクリックすると、その項目の内部値は、ナビゲーションが正常に終了するまで変更されません。したがって、When-Radio-Changedトリガーはラジオ・

グループの変更された値を登録するための最初のトリガーになります。When-Radio-Changedトリガーの前に起動されるすべてのナビゲーション・トリガーに対して、ラジオ・グループの値はオペレータがナビゲートするまで保持されます。

失敗時

影響なし

When-Remove-Recordトリガー

説明

オペレータが、またはアプリケーションでレコードを消去または削除すると起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

When-Remove-Recordトリガーは、Form Builderでレコードが消去または削除されるたびに動作を実行します。

失敗時

Form Builderでブロック・レベルがナビゲートされます。現行の操作によって妥当性チェックがされたりされなかったりします。また、ターゲット・ブロックにカーソルが置かれます。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「CLEAR_RECORD」および「DELETE_RECORD」を参照してください。

When-Tab-Page-Changedトリガー

説明

あるタブ・ページから別のタブ・キャンバスに明示的に項目へまたはマウスでナビゲーションが実行されると、このトリガーを起動します。

定義レベル

フォーム

有効コマンド

制限付きビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- When-Tab-Page-Changedトリガーは、項目またはマウスのナビゲーション中に任意のタブ・ページが変更されるときに動作を実行します。
- When-Tab-Page-Changedトリガーは、タブ・ページのナビゲーションが明示的な場合にのみ起動され、暗示的なナビゲーションには応答しません。たとえば、マウスまたはキーボードを使用してタブ・ページをナビゲートすると、このトリガーは起動されますが、エンド・ユーザーが「次の項目へ」（タブ）を押して、あるフィールドから同じブロックでも異なるタブ・ページの別のフィールドにナビゲートするときには、このトリガーは起動しません。
- When-Tab-Page-Changedトリガーは、タブ・ページがプログラムによって変更されたときには起動しません。

失敗時

影響なし

When-Tab-Page-Changedの例

```
/* Use a When-Tab-Page-Changed trigger to dynamically
** change a tab page's label from lower- to upper-case
** (to indicate to end users if they already have
** navigated to the tab page):
*/
DECLARE
  tp_nm   VARCHAR2(30);
  tp_id   TAB_PAGE;
```

```
  tp_lb   VARCHAR2(30);

BEGIN
  tp_nm := GET_CANVAS_PROPERTY('emp_cvs', topmost_tab_page);
  tp_id := FIND_TAB_PAGE(tp_nm);
  tp_lb := GET_TAB_PAGE_PROPERTY(tp_id, label);

  IF tp_lb LIKE 'Sa%' THEN
    SET_TAB_PAGE_PROPERTY(tp_id, label, 'SALARY');
  ELSIF tp_lb LIKE 'Va%' THEN
    SET_TAB_PAGE_PROPERTY(tp_id, label, 'VACATION');
  ELSE null;
  END IF;
END;
```

When-Timer-Expiredトリガー

説明

タイマーが時間切れになると起動します。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

タイマーはCREATE_TIMERビルトイン・プロシージャのコールによって、プログラムで作成されます。

- When-Timer-Expiredトリガーは、トリガー、ナビゲーション、トランザクションなどの処理中には起動できません。
- When-Timer-Expiredトリガーは、イベントの開始、項目値の更新、指定した間隔で発生させるタスクなどを実行するときに使用します。

■ When-Timer-Expiredトリガー内でGET_APPLICATION_PROPERTY（TIMER_NAME）をコールすると、最後に時間切れになったタイマーの名前が判別できます。

失敗時

影響なし

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「時間切れタイマーの処理」を参照してください。

When-Timer-Expiredトリガーの制限

When-Timer-Expiredトリガーは、ユーザーが現行のメニューをナビゲートしているときは起動しません。

When-Timer-Expiredトリガーの例

次の例では、タイマーの時間切れになるたびにメッセージ・ボックスを表示します。この例は電話販売のアプリケーションの例で、販売のために電話をかける時刻になるとメッセージ・ボックスが表示されて、電話をかける各段階で販売担当員に確認します。メッセージ・ボックスは、タイマーの時間切れになるたびに表示されます。

```
DECLARE
  timer_id  TIMER;
  alert_id  ALERT;
  call_status NUMBER;
  msg_1      VARCHAR2(80) := 'Wrap up the first phase of your
                             presentation';
  msg_2      VARCHAR2(80) := 'Move into your close.';
  msg_3      VARCHAR2(80) := 'Ask for the order or
                             repeat the close.'
  two_minutes NUMBER(6) := (120 * 1000);
  one_and_half NUMBER(5) := (90 * 1000);
BEGIN
  :GLOBAL.timer_count := 1
  BEGIN
    timer_id := FIND_TIMER('tele_timer');
    alert_id := FIND_ALERT('tele_alert');
  IF :GLOBAL.timer_count = 1 THEN
    Set_Alert_Property(alert_id, ALERT_MESSAGE_TEXT, msg_1);
  call_status := Show_Alert(alert_id);
  IF call_status = ALERT_BUTTON1 THEN
    Delete_Timer(timer_id);
    Next_Record;
```

```
  ELSIF
    call_status = ALERT_BUTTON2 THEN
    :GLOBAL.timer_count := 0;
  ELSE
    Set_Timer(timer_id, two_minutes, NO_CHANGE);
  END IF;
ELSIF :GLOBAL.timer_count = 2 THEN
    Change_Alert_Message(alert_id, msg_2);
    call_status := Show_Alert(alert_id);
    IF call_status = ALERT_BUTTON1 THEN
      Delete_Timer(timer_id);
      Next_Record;
    ELSIF
      call_status = ALERT_BUTTON2 THEN
      :GLOBAL.timer_count := 0;
    ELSE
      Set_Timer(timer_id, one_and_half, NO_CHANGE);
    END IF;
  ELSE
    Change_Alert_Message(alert_id, msg_3);
    call_status := Show_Alert(alert_id);
    IF call_status = ALERT_BUTTON1 THEN
      Delete_Timer(timer_id);
      Next_Record;
    ELSIF
     call_status = ALERT_BUTTON2 THEN
     :GLOBAL.timer_count := 0;
    ELSE
      Set_Timer(timer_id, NO_CHANGE, NO_REPEAT);
    END IF;
  END IF;
  :GLOBAL.timer_count = 2;
  END;
END;
```

When-Tree-Node-Activatedトリガー

説明

オペレータがノードをダブルクリックしたとき、あるいはノードが選択された状態で[Enter]キーを押したときに起動します。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

- SYSTEM.TRIGGER_NODEはユーザーがクリックするノードです。SYSTEM.TRIGGER_NODEからはタイプNODEの値が戻されます。
- プログラムによる動作でWhen-Tree-Node-Activatedトリガーが起動されることはありません。エンド・ユーザーの動作のみがイベントを生成します。

失敗時

影響なし

When-Tree-Node-Expandedトリガー

説明

ノードが拡張または縮小されたときに起動します。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

- SYSTEM.TRIGGER_NODEはユーザーがクリックするノードです。SYSTEM.TRIGGER_NODEからはタイプNODEの値が戻されます。

■ プログラムによる動作でWhen-Tree-Node-Expandedトリガーが起動されることはありません。エンド・ユーザーの動作のみがイベントを生成します。

失敗時

影響なし

When-Tree-Node-Selectedトリガー

説明

ノードが選択または選択解除されたときに起動します。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

■ SYSTEM.TRIGGER_NODEはユーザーがクリックするノードです。SYSTEM.TRIGGER_NODEからはタイプNODEの値が戻されます。

■ プログラムによる動作でWhen-Tree-Node-Selectedトリガーが起動されることはありません。エンド・ユーザーの動作だけがイベントを生成します。

失敗時

影響なし

When-Validate-Itemトリガー

説明

「項目の妥当性チェック」プロセス中に起動します。特に、新規または変更のチェック状態の項目に対する妥当性チェックの最後の部分で起動されます。

定義レベル

フォームまたは、ブロック、項目

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

- When-Validate-Itemトリガーは、Form Builderでデフォルト項目の妥当性チェックのプロセスを補足するために使用します。
- 作成したWhen-Validate-Itemトリガーによって、Form Builderが妥当性チェックを行っている項目の値が変更されてしまう可能性があることに注意してください。妥当性チェックが正常終了すると、Form Builderは変更された項目に有効のマークを付けるので、変更した項目をもう一度妥当性チェックすることはありません。この動作は、妥当性チェックのループを防ぐために必要です。 ただし他方ではユーザーのアプリケーションが無効な値をデータベースにコミットしてしまう可能性があります。
- 「Defer_Required_Enforcement」(遅延を必須強制)プロパティは、項目の妥当性チェックからレコードの妥当性チェックまでに「必須」プロパティの施行を延期します。項目の「必須」プロパティが「はい」に設定されていると、デフォルトでは、有効な値が入力されるまでその項目外にナビゲーションすることはできません。しかし、「Defer_Required_Enforcement」プロパティを「PROPERTY TRUE」に設定しておくと、レコード内の項目間を自由に移動できます。

  注意: 「Defer_Required_Enforcement」が「TRUE」であっても、その他のすべての項目の妥当性チェックは通常どおりに実行されます。「必須」プロパティのチェックのみが、オペレータがレコードを放すまで延期されます。

失敗時

ナビゲーションによって開始される妥当性チェックの一部としてこのトリガーが起動されると、ナビゲーションは失敗し、フォーカスは元の項目上に残ります。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「項目の妥当性チェック」を参照してください。

When-Validate-Itemトリガーの例

次の例では、フォームのEMPLOYEEブロック内のcommcode項目の現行の設定値に基づいたCOMMPLAN表で、コミッション計画を検出し、コードが有効であるかをチェックします。COMMPLAN表の中でコードの位置が決まると、COMMPLANのdescription項目が取得され、データベース以外の Description 項目に預けられます。それ以外の場合は、エラーが発生します。

```
** Method 1:Using a SELECT...INTO statement, the trigger
**         looks more readable but can be less efficient
**         than Method 2 because for ANSI Standard
**         compliance, the SELECT...INTO statement must
**         return an error if more than one row is
**         retrieved that matches the criteria.This
**         implies PL/SQL may attempt to fetch data twice
**         from the table in question to insure that there
**         aren't two matching rows.
*/
BEGIN
  SELECT description
    INTO :Employee.Commplan_Desc
    FROM commplan
   WHERE commcode = :Employee.Commcode;
EXCEPTION
  WHEN No.Data_Found THEN
    Message('Invalid Commission Plan, Use <List> for help');
    RAISE Form_Trigger_Failure;
  WHEN Too_Many_Rows THEN
    Message('Error.Duplicate entries in COMMPLAN table!');
    RAISE Form_Trigger_Failure;
END;

/*
** Method 2:Using an Explicit Cursor looks a bit more
**         daunting but is actually quite simple.The
**         SELECT statement is declared as a named cursor
**         in the DECLARE section and then is OPENed,
**         FETCHed, and CLOSEd in the code explicitly
**         (hence the name).Here we guarantee that only a
**         single FETCH will be performed against the
**         database.
*/
DECLARE
  noneFound BOOLEAN;
  CURSOR cp IS SELECT description
```

```
                FROM commplan
               WHERE commcode = :Employee.Commcode;
BEGIN
  OPEN cp;
  FETCH cp INTO :Employee.Commplan_Desc;
  noneFound := cp%NOTFOUND;
  CLOSE cp;
  IF noneFound THEN
    Message('Invalid Commission Plan, Use <List> for help');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

When-Validate-Recordトリガー

説明

「レコードの妥当性チェック」プロセス中に起動します。特に、新規または変更のチェック状態のレコードに対して妥当性チェックの最後の部分で起動しれます。

定義レベル

フォームまたはブロック

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム

問合せ入力モード

不可

使用上の注意

When-Validate-Recordトリガーは、Form Builderでデフォルトのレコードの妥当性チェックのプロセスを補足するために使用します。

作成したWhen-Validate-Recordトリガーによって、Form Bilderが妥当性チェックを行っているレコード内の項目の値が変更されてしまう可能性があることに注意してください。妥当性チェックが正常に終了すると、Form Bilderはレコードおよびすべてのフィールトに有効のマークを付け、もう一度妥当性チェックすることはありません。この動作は、妥当性チェックのループを防ぐために必要です。ただし他方ではユーザーのアプリケーションが無効な値をデータベースにコミットしてしまう可能性があります。

失敗時

ナビゲーションによって開始される妥当性チェックの一部としてこのトリガーが起動されると、ナビゲーションは失敗し、フォーカスは元の項目上に残ります。

起動条件

オンライン・ヘルプの「Form Builderのフローチャート」から「レコードの妥当性チェック」を参照してください。

When-Validate-Recordトリガーの例

次の例では、Start_DateがEnd_Dateよりも小さいことを検証します。これら2つのテキスト項目は、値が関連しているため、各項目を別々にチェックするよりもレコード・レベルで1回チェックした方が便利です。このコードは、両方の日付項目が必須でどちらもNULLではないものとします。

```
/* Method 1:Hardcode the item names into the trigger.
**         Structured this way, the chance this code will
**         be reusable in other forms we write is pretty
**         low because of dependency on block and item
**         names.
*/
BEGIN
  IF :Experiment.Start_Date > :Experiment.End_Date THEN
    Message('Your date range ends before it starts!');
    RAISE Form_Trigger_Failure;
  END IF;
END;

/* Method 2:*/
BEGIN
  IF :Experiment.Start_Date > :Experiment.End_Date THEN
    Message('Your date range ends before it starts!');
    RAISE Form_Trigger_Failure;
  END IF;
END;

/* Method 2: Call a generic procedure to check the date
**         range.This way our date check can be used in
**         any validation trigger where we want to check
**         that a starting date in a range comes before
**         the ending date.Another bonus is that with the
**         error message in one standard place, i.e. the
**         procedure, the user will always get a
```

```
**          consistent failure message, regardless of the
**          form they're currently in.
*/
BEGIN
  Check_Date_Range(:Experiment.Start_Date,:Experiment.End_Date);
END;

/*
** The procedure looks like this
*/
PROCEDURE Check_Date_Range( d1 DATE, d2 DATE ) IS
BEGIN
  IF d1 > d2 THEN
    Message('Your date range ends before it starts!');
    RAISE Form_Trigger_Failure;
  END IF;
END;
```

When-Window-Activatedトリガー

説明

ウィンドウがアクティブ・ウィンドウになると起動します。これは、フォームの起動時および別のウィンドウにフォーカスが移ると発生します。一部のウィンドウ・マネージャでは、タイトル・バーをクリックすることでウィンドウをアクティブにできます。この操作は、そのウィンドウ内の項目へのナビゲーションとは独立しています。したがって、別のウィンドウ内の項目へナビゲートすると、いつもそのウィンドウがアクティブになります。ただし、ウィンドウはナビゲーションとは関係なくアクティブにすることができます。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

このトリガーは、次のタイプのタスクを実行するときに使用します。

- GET_WINDOW_PROPERTYビルトインによって、「ウィンドウ」プロパティの初期設定値を取得します。
- ウィンドウがアクティブになるたびに、特定の項目へのナビゲーションを強制します。
- SYSTEM.EVENT_WINDOWから変数またはグローバル変数に値を割り当てて、最後に起動されたウィンドウ・トリガーを記録します。

失敗時

影響なし

When-Window-Closedトリガー

説明

オペレータがウィンドウ・マネージャ固有の「閉じる」コマンドを使用して、ウィンドウを閉じるときに起動します。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

このトリガーは、オペレータがウィンドウ・マネージャの「閉じる」コマンドを発行したときにプログラムによってウィンドウを閉じるときに使用します。

現行の項目を含むウィンドウを非表示にできます。

失敗時

影響なし

When-Window-Closedトリガーの例

次の例では、オペレータがウィンドウ・マネージャの操作によってウィンドウを閉じるたびに、このトリガーからSET_WINDOW_PROPERTYが常にコールされ、ウィンドウを閉じます。

```
Set_Window_Property('window_name', VISIBLE, PROPERTY_OFF);
```

When-Window-Deactivatedトリガー

説明

オペレータが入力フォーカスを同じフォーム内の別のウィンドウに設定してウィンドウをアクティブ解除にするときに起動します。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

このトリガーは、オペレータが入力フォーカスを別のウィンドウに設定してウィンドウをアクティブ解除にするたびにウィンドウの状態を監査するときに使用します。

このフォームが別のフォームを開く場合、このアクティブ解除トリガーは即時には起動しないので注意が必要です。即時ではなく、制御がこのフォームに戻ってから起動します（このウィンドウにアクティブ化トリガーが同時に存在する場合、制御がこのフォームに戻ると、まずアクティブ化トリガーが起動し、それに続いて直ちにアクティブ解除トリガーが起動します）。

失敗時

影響なし

When-Window-Resizedトリガー

説明

オペレータが、またはプログラムによって、RESIZE_WINDOWまたはSET_WINDOW_PROPERTYのコールを使用して、ウィンドウのサイズが変更される場合に起動します（ウィンドウが現在表示されていない場合でも、プログラムによりウィンドウのサイズが変更されると、When-Window-Resizedトリガーが起動されます）。このトリガーは、フォー

ムの起動時にルート・ウィンドウが最初にコールされるときにも起動されます。ウィンドウがアイコン化されているときには起動しません。

定義レベル

フォーム

有効コマンド

SELECT文、制限なしビルトイン・サブプログラム、制限付きビルトイン・サブプログラム

問合せ入力モード

可

使用上の注意

このトリガーは、次のタイプのタスクの1つを実行するときに使用します。

- 幅または、高さ、x座標、y座標のような変更されたウィンドウ・プロパティを取得します。
- オペレータの操作を監査します。
- ターゲット・ウィンドウの項目に入力フォーカスを設定します。
- ウィンドウが不適切にサイズ変更された場合、ウィンドウ・サイズを再設定して、特定の可視基準を保持します。

失敗時

影響なし

索引

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

い

う

え

お

か

こ

し

せ

た

ち

ゆ

ら

り

れ

ろ