

Visual LANSА 機能ヘルプ

このヘルプはVisual LANSА イベントやコンポーネントでF2が押された時や機能タブのツールバーで疑問符のアイコンが押された時にヘルプとして表示されるテキストです。

このテキストはガイドとして読むためではなく、LANSА サポート記述情報の検索結果として提供するのが目的です。

エディション日付：2013年5月2日

© LANSА

基本オブジェクト

基本オブジェクト。表示されないコンポーネントの祖先として使用します。

LANSAコンポーネント・クラス階層のルートです。

[基本オブジェクトのプロパティ](#)

[基本オブジェクトのイベント](#)

基本オブジェクトのプロパティ

ComponentPatternName プロパティ

ComponentClassName プロパティ

Component tag プロパティ

Component message set プロパティ

ComponentTypeName プロパティ

ComponentType プロパティ

ComponentMembers プロパティ

MessagesOfFeature プロパティ

MessagesOfComponent プロパティ

MessageSetOfFeature プロパティ

Name プロパティ

Owner プロパティ

Parent プロパティ

ComponentPatternName プロパティ

ComponentPatternNameは、コンポーネントのクラスを制限する際に使用します。

ComponentClassName プロパティ

ComponentClassNameは、コンポーネントのクラス名です。

Component tag プロパティ

LANSA製品センター内部専用です。

Component message set プロパティ
LANSA製品センター内部専用です。

ComponentTypeName プロパティ

ComponentTypeNameは、コンポーネントの完全修飾名です。

ComponentType プロパティ

ComponentType プロパティを使用して、コンポーネントのタイプ情報にアクセスします。

このプロパティでJAVAへの反映を参照する機能を有効にします。

Visual LANSAコンポーネントのクラスにアクセスしたり、その特定の質や機能をチェックするツールを書く際にのみこのプロパティを使用します。

ComponentMembers プロパティ

ComponentMembersは、このコンポーネントの全てのメンバー・コンポーネントへのアクセスを提供します。

ComponentMembers プロパティにより、現在のオーナー・コンポーネントのメンバー全ての集まりであるコレクションにアクセスできます。

ComponentMembersを利用して、オーナー・コンポーネントのコンテンツを調べたり、全てのメンバーに対する包括的なプログラミングをすることができます。

次の例では、フォーム上の全メンバー・コンポーネントを記録してグリッドに記載します。

```
For Each(#Current) In(#Com_Owner.ComponentMembers)
  Change Field(#STD_NAME) To('#CURRENT.NAME')
  Add_Entry To_List(#GRID_1)
Endfor
```

ComponentMembers プロパティは、他のコンポーネントのオーナー（詳細はオーナー・プロパティを参照してください）のコンポーネントにのみ使用できることに注意してください。例えば、リストはその中に含まれている列のオーナーではありません。（列のプロパティにアクセスするには、リストのColumns プロパティを利用します。）

MessagesOfFeature プロパティ
LANSA製品センター内部専用です。

MessagesOfComponent プロパティ

LANSA製品センター内部専用です。

MessageSetOfFeature プロパティ
LANSA製品センター内部専用です。

Name プロパティ

コンポーネントを識別する名前のプロパティです。名前はコンポーネントの作成時に指定します。最大20文字までの名前を指定することができます。

別のコンポーネント内でコンポーネントを利用する時（例えばフォームのプッシュ・ボタンなど）、LANSAではコンポーネントのリストそれぞれにPHBN_1、PHBN_2といったデフォルトの名前が付けられます。この名前を変更して意味のある名前にすることが可能です。例えば、このボタンがfetch操作を実行する場合、このボタンを"Fetch"と名付けることができます。

コンポーネントの名前は実行時は表示されません。ユーザーに表示される記述にはCaptionやLabelのプロパティを使用します。

（例えばリスト内の）現在の項目などの場合、現項目に含まれているテキストが名前になります。

Owner プロパティ

Ownerはこのコンポーネントを所有しています。

Owner プロパティは現在のコンポーネントを所有するコンポーネントを示します。

複数のフォームからなるアプリケーションを作成する際、実行時に FormOwner プロパティを利用してメンバー・フォームのオーナー・フォームを指定する必要があります。詳細は、「[FormOwner プロパティ](#)」を参照してください。

Parent プロパティ

Parent プロパティはどのコンポーネントに他のコンポーネントが含まれるかを示します。現在選択されているコンポーネントは親がありません。ですから、このParent プロパティはOwner（オーナー）の値に設定されます。

基本オブジェクトのイベント

CreateInstance イベント

DestroyInstance イベント

CreateInstance イベント

CreateInstanceに信号が送られるのは、コンポーネントのインスタンスが作成される時です。

コンポーネントのインスタンスが作成されると直後にCreateInstance event に信号が送られます。これは例えばフォームや再利用可能パーツが表示されようとする時に開始される最初のイベントです。

このCreateInstanceイベントはコンポーネントを表示する時に開始される初期化イベントより優先されます。ですから、コンポーネントが表示される前に処理されなければならないコードはこのCreateInstanceイベントに入れます。例えばサイズなどのフォームの特性を設定するルーチンなどは、初期化のイベントではジャンプ効果を作り出すので、CreateInstanceイベントに含むのが一番です。

非ビジュアル・コンポーネントの初期化用コードは、初期化イベントに信号を送らないので、CreateInstanceイベントに含みません。

DestroyInstance イベント

コンポーネントのインスタンスが破棄されようとする
と、DestroyInstanceに信号が送られます。

DestroyInstanceイベントに信号が送られるのは、
コンポーネントのインスタンスが破棄される直前です。

キー付きコレクション

コンポーネント・インスタンスのセット

キー付きコレクション(Keyed collections)では、キー値により識別されるコンポーネントが不規則な順番で並んでいます。キー値は重複できません。

キー付きコレクションではフィールド値がキーになっています。つまり、コレクション内のそれぞれの項目は一意的なフィールド値により識別できます。

例として、社員詳細を表示するフォームを使用してみましょう。このアプリケーションでは複数の社員個人用のフォームを同時に表示できるようにします。フォームの各個人のインスタンスは社員の社員番号により識別されます。

フォームのコレクションは、以下のように定義されます。

```
DEFINE_COM CLASS(#PRIM_KCOL) NAME(#DFORMS) COLLECTS(#F
```

または

```
DEFINE_COM CLASS(#PRIM_KCOL<#FRMDETAIL #EMPNO>) NAME(
```

このコレクションを定義する時、コレクション名とキーを指定することで、コレクション内の各個人の項目を参照することができます。次のコードでは現在選択された社員の#FRMDETAILS ウィンドウが表示されます。

```
INVOKE #DFORMS<#EMPNO>.SHOWFORM
```

そして以下のコードでは社員番号A0070の詳細が表示されます。

```
INVOKE #DFORMS<'A0070'>.SHOWFORM
```

同様にプロパティを設定する時にこのフォームのインスタンスに参照することも可能です。

```
set #DFORMS<#deptment> Caption(#deptment)
```

コレクションのイベント用の包括的なルーチンを書いて、コレクション内のすべての項目に対して開始させることも可能です。

```
EVTROUTINE HANDLING(#DFORMS<>.Closing)
```

フォームを保存する時は、コードは自動的に再フォーマットされます。
キー付きコレクションのプロパティ

キー付きコレクションのプロパティ

Collects プロパティ

KeepLast プロパティ

KeyedBy プロパティ

Style プロパティ

Collects プロパティ

Collectsではフォームの詳細を記述します。

Collectsプロパティを使ってコンポーネント・インスタンスを作成する時に見本となるコンポーネントのタイプを指定します。

KeepLast プロパティ

KeepLastでは管理するコンポーネントのインスタンス数を指定します。KeepLastプロパティを使ってコレクション内で管理する項目数を指定します。

KeepLastを0に設定すると、このプロパティは無視され、KeepLastチェックは行われません。この場合、コレクションに追加できる項目数で制限を受けるのはマシンやメモリの制限だけです。

KeepLastプロパティの最大数は99です。コレクションに追加された項目の数がKeepLast値を超える場合、KeepLastのカウントが再度設定数に満たされるまで、項目はコレクションから除外されます。この除外される項目は、過去に一番長い間アクセスされていないものです。

KeyedBy プロパティ

KeyedByにはフォーム・インスタンスの識別に使用されるフィールドを指定します。

コレクション内の項目を識別できるフィールドを指定するのにKeyedByプロパティを使います。

キーとして使用されたフィールドはリポジトリに登録されたフィールドでなければなりません。

Style プロパティ

Styleではコレクションが存在しないキーでアクセスされた場合にどうするかを指定します。

コレクションがコレクション内に存在しないキーでアクセスされた場合にどうするかをStyleプロパティに指定します。このパラメータは、次のいずれかの値を取ることができます。

FACTORY コレクションに存在しないキーを使ってコレクションがアクセスされた時に、Collectsプロパティにより指定されたタイプのコンポーネントの新しいインスタンスを作成したい場合はこの値を使用します。この新しい項目はキーに関連付けられ、コレクションに格納されません。FactoryはStyleプロパティの省略値です。

COLLECTION コレクションに存在しないキーを使ってコレクションがアクセスされた時に、*NULL値の参照を返したい場合はこの値を使用します。このコレクション・スタイルを使用する場合、SET_REFやIF_REFコマンドを使ってコレクションの処理を行う必要があります。このスタイルのコレクションではPRIM_OBJTなどの抽象的なクラスをコレクトすることができます。コレクションはコンポーネント・インスタンスを作成しないので、Visual LANSAアプリケーションによりインスタンスを作成したり、適切なキーを使ってコレクション内にインスタンスを割り当てたりします。

例：

```
DEFINE_COM CLASS(#PRIM_KCOL) NAME(#FORM_COL) COLLECTS
```

```
IF full time employee
```

```
SET_REF COM(#FORM_COL<#CUR_EMPLNO>) TO(*CREATE_AS #FO
```

```
if part time employee
```

```
SET_REF COM(#FORM_COL<#CUR_EMPLNO>) TO(*CREATE_AS#FOF
```

```
else
```

```
if contract employee
```

```
SET_REF COM(#FORM_COL<#CUR_EMPLNO>)
```

```
TO(*CREATE_AS #FORM_CT)
```

```
endif
```

```
endif
```

endif

注： #FORM_FT、 #FORM_PT、 FORM_CTは全てPRIM_FORMから継承するVisual LANSAフォームです。

基本コントロール

基本ビジュアルコントロールです。

LANSACOMPONENT・クラス階層のルートです。

[基本コントロールのプロパティ](#)

[基本コントロールのイベント](#)

基本コントロールのプロパティ

Alignment プロパティ	Hint プロパティ	SelectionEnd プロパティ
BusyCancelResult プロパティ	HintShow プロパティ	SelectionStart プロパティ
BusyUpdates プロパティ	HintShowOfParent プロパティ	ShowError プロパティ
BusyUpdatesOfParent プロパティ	Image プロパティ	ShowLines プロパティ
CanFocus プロパティ	ImageExpanded プロパティ	ShowSortArrow プロパティ
Caption プロパティ	ImageOverlay プロパティ	TabPosition プロパティ
Cursor プロパティ	ImageState プロパティ	TabStop プロパティ
DataClass プロパティ	Left プロパティ	Text プロパティ
DisplayPosition プロパティ	ModalResult プロパティ	Value プロパティ
DragCursor プロパティ	Modified プロパティ	VerticalAlignment プロパティ
DragStyle プロパティ	Parent プロパティ	Visible プロパティ
Enabled プロパティ	PopupMenu プロパティ	VisualStyle プロパティ
Focus プロパティ	ReadOnly プロパティ	VisualStyleEdit プロパティ
Handle プロパティ	ScreenLeft プロパティ	VisualStyleOfParent プロパティ
HasFocus プロパティ	ScreenTop プロパティ	Width プロパティ
Height プロパティ		

Alignment プロパティ

Alignmentはコンポーネントを整列したり中央に位置づけたりします。Alignmentプロパティを使って、コンポーネント内のテキストを左、中央、右揃えのいずれにするかを指定します。

BusyCancelResult プロパティ

このプロパティは、使用できません。

BusyUpdates プロパティ

BusyUpdatesはコントロール自体がどのように更新されるかを指定します。

BusyUpdates プロパティを使って、オペレーションがビジーな間にコンポーネントが変更された時にコンポーネント自体をどのように更新するかを指定します。

このパラメータは、次のいずれかの値を取ることができます：

Wait - (省略値) ビジュアル・コンポーネントはビジー状態のオペレーションが終了するまで待ちます。

Immediate - ビジュアル・コンポーネントは次の更新の機会に更新されます。

Interval01 - ビジュアル・コンポーネントは約2秒ごとに更新されます。

Interval02 - Interval01と同様ですが、間隔が2倍、つまり4秒ごとに更新されます。

Interval04 - Interval01と同様ですが、間隔が4倍、つまり8秒ごとに更新されます。

BusyUpdatesOfParent プロパティ

BusyUpdatesOfParentは親が更新を制御するかどうかを指定します。

BusyUpdatesOfParent プロパティを使用してコンポーネントの BusyUpdates プロパティをコンテナ/フォームのレベルで管理するかどうかを指定します。

このプロパティにTrueを設定すると、コンポーネントの親のBusyUpdatesが使用されます。

CanFocus プロパティ

CanFocusはコントロールにフォーカスを当てるかどうかを示す値を返します。

CanFocusはコントロールにフォーカスを当てることが可能な場合はTrueを返します。それ以外はFalseを返します。

コントロールにインプット・フォーカスを当てるためには、以下のような条件があります：

- コントロールが実際に使えるようになっていて、ウィンドウ・ハンドルが割り当てられていなければなりません。
- VisibleとEnabledプロパティでコントロールとその親のコントロールに対して両方ともTrueに設定されていなければなりません。
- コントロールがフォームである、もしくはコントロールの最外部の親がフォームでなければなりません。

Caption プロパティ

Captionはコンポーネントの表示されるテキストです。

コンポーネントの表示されるテキストがCaptionです。キャプションがどのように表示されるかはコンポーネントのタイプによります：プッシュ・ボタンの場合はボタン上に、フォームの場合はそのタイトルとして、チェックボックスの場合はその隣に、メニュー項目の場合はメニュー・オプションの名前として、グラフの場合はタイトルとしてテキストが表示されます。

Captionプロパティを使用して、コンポーネントにアクセス・キーを割り当てることができます。キャプション内でアクセス・キーとして指定する文字のすぐ前にアンパサンド(&)を付けます。この文字には下線が引かれます。ユーザーがALTキーと下線が引かれた文字を押すと、そのコンポーネントにフォーカスが移動します。そしてユーザーはEnterキーを押して、そのコンポーネントのClickイベントを起動させることができます。

FocusControlが指定されている場合は、アクセス・キーはラベルにのみ有効であることに注意してください。

Captionプロパティをクリックすると、省略記号(3つの点)の付いたボタンが表示されます。これをクリックして、キャプションに言語変数を選択できます。設計時、言語変数のテキストはオプション・メニューで現在の言語として指定された言語で表示されます。実行時は、区画で選択された言語でテキストが表示されます。

RDMLXのソースコードでキャプションを入力する場合は、テキストを単一引用符で囲みます。これをプロパティ・シートで入力した場合は、自動的に単一引用符が加えられます。

Cursor プロパティ

Cursorはカーソルのイメージを設定します。

Cursorプロパティを使用して、実行時にマウス・ポインタがコンポーネント上にある時に表示されるイメージを設定します。

このプロパティを使用して、マウス・ポインタがコンポーネント上を通過する際に機能の変更を示すことができます。LANSAリポジトリに定義されているシステム・カーソルやカーソルから選ぶことができます。これには製品と共に提供される標準カーソルとユーザーが定義した場合はそのカーソルも含まれます。

DataClass プロパティ

DataClassはコントロールに表示される情報を制御します。

DataClass プロパティを使用して、実行時のコンポーネントに表示される情報を指定します。

DataClass プロパティの値はフィールドか*NULLです。値を*NULLに指定した場合は、データクラスは指定されません。DataClassの値としてフィールドが指定されると、このフィールドの値がコンポーネントに表示され、入力された値にはそのフィールドの編集ルールが使われます。DataClassの値は、そのコントロールに適切なタイプであれば、どんなフィールドでも指定することができます。

DataClass プロパティの値としてフィールドを追加するには、DataClassの値をクリックし、省略記号（3つの点）の付いたボタンをクリックします。そしてフィールドを選択します。

DataClass プロパティの値としてフィールドを割り当てると、そのフィールドのデータ・クラス定義が作成されます。

```
DEFINE_COM CLASS(#STD_NUMBR) NAME(#STD_NUMBR)
```

データ・クラスとしてのフィールドとフォーム上に表示されるフィールドの違いを理解するために、フォームにドラッグされたフィールドは以下のようなコンポーネント定義を作成することに注意してください。

```
DEFINE_COM CLASS(#STD_NUMBR.Visual) NAME(#STD_NUMBR)
```

コンポーネント・クラスのビジュアル修飾子がコンポーネントのビジュアル表示が作成されたことを意味するのに対し、データ・クラス記述では単にそのフィールドのタイプと属性の非ビジュアルなコンポーネントのみを指定します。

データ・クラスのコンポーネントには次のような使用方法があります：

編集ボックス：フィールドそのものを使った方が簡単なので、通常はフィールドを編集ボックスのDataClass値として使用しません。

スピン編集ボックス：スピン編集ボタンがついた数値フィールドを表示する場合、フォームにスピン編集ボックスを追加し、そのDataClass値としてフィールドを指定します。例えば、フィールド#STD_NUMをフォームに追加すると、標準の入力/出力フィールドとして表示されます。このフィールドにスピン・ボタンをつけて表示する場合は、最初にスピン編集ボックスを追加し、フィールド#STD_NUMをスピン編集の

DataClassプロパティとして指定します。 スピン編集ボックスは、スピン・ボタンとは別に#STD_NUMフィールドと同様の動きをします。 スピン・ボタンは、数の増加に使用するので、スピン編集ボックスのDataClassプロパティの値には、数値フィールドのみが使用可能です。 コンボ・ボックス： DataClassを指定して、入力された値にコンポーネントのデータ・クラスの編集ルール（例えばタイプや長さなど）を使用します。

また、コンボ・ボックスの編集部分には、最初の列(DisplayPositionが1)の現在の項目の値が表示されます。 最初の列のソースとして使用されるフィールド以外のフィールド値を編集エリアに表示するには、このフィールドをDataClassとして指定します。 このフィールドは、コンボ・ボックスの他の列のソースとして使用されるフィールドのいずれか、またはリポジトリ内の他のフィールドです。

編集エリアに列を表示する場合、DisplayPositionを1に変更する方が簡単なので、通常はコンボ・ボックス内の列として使用するフィールドをDataClass値として指定することはありません。 より典型的な使用方法としては、複数列の現在値を結合させるフィールドや、DataClassとしてフィールドを使用して編集エリアで全ての値が表示されるようにする場合などです。 例えば、コンボ・ボックスに社員の名前と名字の列がある場合、この2つのフィールドの値を連結したフィールドをDataClassとして使用することができます。

プロパティ・シート： プロパティ・シートの全てのエントリーはデータ・クラスに基づいたものでなければなりません。 詳細は、プロパティ・シートのヘルプを参照してください。

DragStyle プロパティ

DragStyleはコンポーネントがどのようにドラッグされるかを決定します。

DragStyleプロパティを使用して、コンポーネントをどのようにドラッグするか制御します。このプロパティは、次の値を取ることができます：

Noneはコンポーネントでドラッグ操作の開始をサポートしません。

Automaticはマウスがいったん下に置かれ、事前に定義された距離を動くと、StartDragイベントが起動し、ドラッグ操作が始まります。

StartDragはこのコンポーネントのDragStyleがその親のDragStyleと同じことを意味します。親のDragStyleがAutomaticで、ドラッグ操作が子のコンポーネントで開始された場合、StartDragイベントが親側で開始されてドラッグ操作が始まります。

DisplayPosition プロパティ

DisplayPositionはコンポーネントがどのように表示されるかを決定します。

DisplayPositionポジションを使用することで、親コンポーネント内に表示されているコントロールに対し、指定のコンポーネントを相対的に前または後ろに置くことができます。

親コンポーネントはフォームだけでなく、どんな種類のリスト（リスト・ボックス、リスト・ビュー・ツリー・ビューまたはグリッド）、タブ・フォルダもしくはタブ・シート、ステータス・バーでもパネルでも可能です。つまり別のコンポーネントを持つことができるコンポーネントなら何でも可能です。

リストに関しては、列のDisplayPositionがリスト内のどの列に表示するかの順序を決定します。またこれはその列を表示させるかどうかも決定します。（VisibleプロパティがFalseに設定されている列は表示位置(DisplayPosition)が0になります。）

ツリー・ビューはレベルごとに列は1つだけしか表示できないことに注意してください。

タブ・フォルダーでは、DisplayPositionはタブ・シートの順番を決定します。

ステータス・バーでは、DisplayPositionはMessagePositionと共にコンポーネントの順序を決定します。

DragCursor プロパティ

DragCursorはドラッグ時のカーソル・イメージを設定します。

DragCursorプロパティを使用して、コンポーネントがドラッグされた際に表示するカーソル・イメージを設定します。

このプロパティを使うことで、ドラッグ・アンド・ドロップ操作時にビジュアルなフィードバックを提供できます。例えばコンポーネントが適切なターゲットにドラッグされていることを示すことが可能です。

DragCursorはユーザーがドラッグ・アンド・ドロップ操作を開始すると有効になります。典型例としては、MouseDownやDragOverイベントの一部としてこのプロパティを設定します。

Enabled プロパティ

Enableはコンポーネントをアクティブ化/非アクティブ化します。

Enableプロパティを利用して、ユーザーにより開始されたイベントにコンポーネントが反応するかどうかを指定します。

このプロパティで実行時にコンポーネントの有効/無効を設定します。
例えば、アプリケーションの現在の状態に当てはまらないコンポーネントを無効にしたりできます。

コンポーネントを無効にせずに書き込めないようにするには、ReadOnlyプロパティを使用します。

次のコードではプッシュ・ボタン#PHBN_1を無効にします：

```
SET COM(#PHBN_1) ENABLED(FALSE)
```

SET例161も参照してください：[フォームのコンポーネントの有効・無効化](#).

Focus プロパティ

Focusはコンポーネントがフォーカスされるどうかを決定します。

コンポーネントがフォーカスされるかどうかを決定するのにFocusプロパティを使います。このプロパティは、TrueまたはFalseに設定できます。

ユーザーはキーボードやマウスを使ってフォーカスを設定できます。SetFocusメソッドを利用してプログラム上でフォーカスを設定することもできます。

コンポーネントにフォーカスが当たると、GotFocusイベントが開始し、コンポーネントからフォーカスが外されるとLostFocusイベントが開始します。

Height プロパティ

Heightにはコンポーネントの高さを設定します。

Heightプロパティを利用してコンポーネントの高さをピクセルで指定します。

コンポーネントを選択して、サイズ変更グリップからドラッグすることで高さをビジュアル的に調整することも可能です。ピクセル値は自動的に調整されます。

複数のコンポーネントのサイズを一度に設定する場合は、編集メニューのサイズと整列を利用します。

Handle プロパティ

読み取り専用プロパティです。

Handle プロパティにより、Windowsのウィンドウの基礎操作ができるようになります。特定のActiveXコントロールもサポート可能になります。

Hint プロパティ

Hintはコンポーネントの簡単な説明です。

Hintパラメータを使用して、コンポーネントの簡単な説明を入力します。これは実行時にユーザーがコンポーネントの上のカーソルを通過する際に表示されます。

リスト、ツリービュー、グリッドやプロパティ・シートの項目のヒントを表示したい場合は、ItemHintTextイベントを利用してShowItemHintプロパティをTrueに設定します。

HasFocus プロパティ

HasFocusはコンポーネントにフォーカスがあるかどうかを示します。

HasFocusプロパティを使ってコンポーネントをアクティブにしたり、アクティブかどうかを調べることができます。このプロパティは実行時のみ使用可能です。例えば、HasFocusプロパティを利用してタブのどのタブ・シートがアクティブなのか調べることができます。

Image プロパティ

Imageではコンポーネント内で使用されるイメージを指定します。

コンポーネントで使用するイメージを指定するのにImageプロパティを使います。

プッシュ・ボタン、ツールバー・ボタンやイメージの場合、Imageプロパティの値はリポジトリに登録されたビットマップやアイコンになります。

ツリービュー、リスト・ビュー、タブ・シートの場合、Imageプロパティの値はリポジトリに登録されたアイコンになります。

ツリービューとリスト・ビューのイメージは各項目の前に置かれます。ツリービューでは、ツリービューの列の値、または実行時は各項目の値としてImageプロパティを指定することができます。リスト・ビューでは、実行時にのみ各項目にイメージを指定できます。

次のコードは、苗字がBrownのリスト項目に#IconBを設定し、残りの全てのリスト項目に#EmpIconを設定します。

```
SELECT FIELDS(#ltvw_1) FROM_FILE(pslmst)
```

```
add_entry to_list(#ltvw_1)
```

```
IF COND'#SURNAME *EQ BROWN'  
SET COM(#LTVW_1.CURRENTITEM) IMAGE(#ICONB)  
ELSE  
SET COM(#LTVW_1.CURRENTITEM) IMAGE(#EMPICON)  
ENDIF  
ENDSELECT
```

リスト・ビューの列の列ヘッダーにイメージを利用することもできます。このイメージはリポジトリに登録されたアイコンでなければなりません。列ヘッダーにイメージを割り当てると、その項目自体はアイコンがなかったとしても、最初の列はヘッダーのアイコンに必要なスペースを空けて全ての項目が右側に調整されることに注意してください。

ImageExpanded プロパティ

ImageExpandedは項目のステータスを示します。

ImageExpandedプロパティを使用してリポジトリに登録されたアイコンを展開されたツリービュー項目用に使用できます。項目が折りたたまれていると、Imageプロパティに指定されているイメージが表示されません。

次のコードは展開されたツリービュー項目のアイコンに#OPENFLRを設定します。

```
SET COM(#LTVW_1.CURRENTITEM) IMAGEEXPANDED(#OPENFLR)
```

ImageOverlay プロパティ

ImageOverlayはアイコンの上に別のイメージを重ねます。

ImageOverlayプロパティを使って、ツリービューやリストビュー項目のイメージの上に重ねる、リポジトリに登録されたアイコンを指定します。

上に重ねるイメージの例としてはバツ印や線の入った丸などで、イメージの上に重ねることで、特定の項目が使用できない、もしくは別で予約されている、無効になっていることなどを示すことができます。

ImageState プロパティ

ImageState プロパティを使って、リポジトリに登録されたアイコンを指定して、ツリービューやリストビュー項目のステータスを示します。この状態イメージは項目のイメージの前に置かれます。

この状態イメージの例として、LANSAフィールド/コンポーネント・リストの前にあるイメージを見てください。これはビルドの状態を示しています。

次のコードは現在のリスト項目の状態イメージに #OK を設定します。

```
SET COM(#LTVW_1.CURRENTITEM) ImageState(#OK)
```

Left プロパティ

Left プロパティを使って、コンポーネントの左端とコンテナの左端の間の距離をピクセルで指定します。

コンポーネントを選択しドラッグすることで、位置を調整することも可能です。プロパティのピクセル値は自動的に調整されます。

複数のコンポーネントの位置を一度に設定する場合は、編集メニューの整列を利用します。

ModalResult プロパティ

ModalResultはモーダル・フォームがどのように閉じられたかを示します。

設計時はプッシュ・ボタンのプロパティ、実行時はフォームのプロパティになります。

ShowModalFormメソッドを使用してフォームを表示する際、ModalResultプロパティを使うことでユーザーがそのフォームをどのようにして閉じたかを確認することができます。フォームが閉じられると、そのフォームのModalResultプロパティは、そのフォームを閉じる時に使用されたボタン（もしくは別のアクション）を示します。

設計時はフォームの各プッシュ・ボタンにModalResultを割り当て、Abort、Cancel、Ignore、No、None、OK、Retry、Yesのいずれかの値を選択します。実行時にこれらのプッシュ・ボタンが押されると、自動的にそのボタンがあるフォームにModalResult値が割り当てられます。

プッシュ・ボタン以外のコントロール・イベントにModalResult値を割り当てるイベント・ルーチンを書くことも可能です。

フォームを閉じることで、ModalResultにはCancelの値が設定されます。Noneという値はまだ何の結果も受け取っていないことを示し、そのフォームはモーダル状態で実行されるはずで

その他の値は事前に定義された意味はありません。

Modified プロパティ

実行時のプロパティです。

Modified プロパティを使ってコンポーネントの内容がユーザーによって変更されたかどうか確認します。

このプロパティはユーザーがコンポーネント内で変更をした時、自動的にTrueに設定されます。プログラム上でコンポーネントの内容が変更された場合は、Modifiedフラグはonに設定されません。

フォームとその他のコンテナ・コンポーネント（パネル、タブ、タブ・シート、グループ・ボックスなど）では、属するいずれかのコンポーネントのModifiedプロパティがTrueに設定されると、ModifiedプロパティがTrueに設定されます。コンテナのModifiedプロパティがFalseに設定されると、これに属する全てのコントロールのプロパティがリセットされます。

ユーザーによって次のような操作が行われた場合、ModifiedプロパティがTrueに設定されます：

- ラジオ・ボタンのButtonCheckedの状態、もしくはチェック・ボックスのButtonStateプロパティを変更する
- フィールド、編集ボックス、スピン編集ボックス、複数行編集ボックスやトラック・バーで値を変更する
- コンボ・ボックスの編集部分で値を変更する、もしくはコンボ・ボックス内の別の項目にフォーカスを移動する
- リスト・ボックス、リスト・ビュー、ツリー・ビューでフォーカス/選択を変更する リスト・ビューやツリー・ビューで項目のキャプションを変更する
- グリッド・セルやプロパティ・シート項目の値を変更する、もしくはグリッドやプロパティ・シート内でフォーカス/選択を別の項目に移動する。

ツールバー・ボタンではこのプロパティが自動的にTrueに設定されることはありません。（プログラムで設定します）

コントロールの内容を処理後、ModifiedプロパティをFalseに返します。

Parent プロパティ

Parentはコンポーネントのコンテナです。

このコンポーネントを含むコンポーネントがParentです。Parentは例えばフォームだったり、グループ・ボックスまたはパネルだったりします。コンポーネントはその親の外に移動することはできません。

コンポーネントをコンテナにドラッグすると、LANSAは自動的にそのコンテナを親として割り当てます。例えばラジオ・ボタンをグループ・ボックスにドラッグすると、そのコンポーネント定義にはParentプロパティが含まれます。

```
DEFINE_COM CLASS(#PRIM_RDBN) NAME(#RDBN_1) PARENT(#GPB
```

PopupMenu プロパティ

PopupMenuはポップアップ・メニューを呼び出します。

PopupMenuプロパティを使って、ユーザーがコンポーネントを選択してマウスで右クリックしたときに表示されるポップアップ・メニューの名前を指定します。

ポップアップ・メニューはpop-up menuコンポーネントを使用して作成します。

HintShowOfParent プロパティ

HintShowOfParentは親に基づいて自身のHintShowを設定します。

HintShowOfParent プロパティを使って、コンポーネントのコンテナのHintShow プロパティが適用されるかどうか決定します。

このプロパティがTrueに設定されていると、コンポーネントが含まれているフォーム、パネル、グループ・ボックス、タブ・フォルダのHintShow プロパティが、そのコンポーネントのhint プロパティをコントロールします。

HintShow プロパティ

HintShowはヒントのオンとオフを設定します。

HintShowプロパティを使って、コンポーネントのヒントの有無を設定します。このプロパティがTrueに設定されていると、そのコンポーネントのHintプロパティに定義された記述が表示されます。

HintParentShowプロパティがTrueに設定されている場合、このプロパティの値は無視されることに注意してください。

ScreenLeft プロパティ

ScreenLeftは画面の左からの距離を設定します。

このコンポーネントの画面の左側からの距離を設定するのに、ScreenLeftプロパティを使います。

ScreenLeftプロパティは現在の画面サイズに基づいてピクセルで計算されます。

これは複数のフォームを利用するアプリケーションでフォームの位置をコントロールする時などによく利用されます。

このプロパティがどのように使用されるかを見るには、次のコードを2つのフォームにコピー・貼り付けてください。

フォーム1:

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM FRAMESTYLE(Dialog) HEIGHT(89) LEFT(436) TOP(218) V
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#BTN_CLOSE) BUTTOND
EVTROUTINE HANDLING(#com_owner.Initialize)
SET COM(#com_owner) CAPTION(*component_desc)
ENDROUTINE
EVTROUTINE HANDLING(#BTN_CLOSE.Click)
INVOKE METHOD(#com_owner.CloseForm)
ENDROUTINE
END_COM
```

フォーム2:

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM CAPTION('Screen Positioning Example') HEIGHT(293) LEFT
DEFINE_COM CLASS(#FORM1) NAME(#POSTEST)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1) CAPTION('Shov
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_2) CAPTION('Shov
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_3) CAPTION('Shov
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_4) CAPTION('Shov
EVTROUTINE HANDLING(#PHBN_1.Click)
DEFINE FIELD(#TOP) TYPE(*DEC) LENGTH(7) DECIMALS(0)
DEFINE FIELD(#LEFT) TYPE(*DEC) LENGTH(7) DECIMALS(0)
CHANGE FIELD(#TOP) TO('#PHBN_1.screentop - #POSTEST.HEIGHT')
CHANGE FIELD(#LEFT) TO('#PHBN_1.screenleft - #POSTEST.WIDTH')
SET COM(#POSTEST) TOP(#TOP) LEFT(#LEFT)
```

```
INVOKE METHOD(#POSTEST.showform)
ENDROUTINE
EVTROUTINE HANDLING(#PHBN_2.Click)
CHANGE FIELD(#TOP) TO('#PHBN_2.screentop - #POSTEST.HEIGHT')
CHANGE FIELD(#LEFT) TO('#PHBN_2.screenleft + #PHBN_2.width')
SET COM(#POSTEST) TOP(#TOP) LEFT(#LEFT)
INVOKE METHOD(#POSTEST.showform)
ENDROUTINE
EVTROUTINE HANDLING(#PHBN_3.Click)
CHANGE FIELD(#TOP) TO('#PHBN_3.screentop + #PHBN_3.HEIGHT')
CHANGE FIELD(#LEFT) TO('#PHBN_3.screenleft - #POSTEST.width')
SET COM(#POSTEST) TOP(#TOP) LEFT(#LEFT)
INVOKE METHOD(#POSTEST.showform)
ENDROUTINE
EVTROUTINE HANDLING(#PHBN_4.Click)
CHANGE FIELD(#TOP) TO('#PHBN_4.screentop + #PHBN_4.HEIGHT')
CHANGE FIELD(#LEFT) TO('#PHBN_4.screenleft + #PHBN_4.width')
SET COM(#POSTEST) TOP(#TOP) LEFT(#LEFT)
INVOKE METHOD(#POSTEST.showform)
ENDROUTINE
EVTROUTINE HANDLING(#com_owner.closing)
INVOKE METHOD(#POSTEST.closeform)
ENDROUTINE
END_COM
```

ScreenTop プロパティ

ScreenTopは画面の上からの距離を設定します。

このコンポーネントの画面の上側からの距離を設定するのに、ScreenLeftプロパティを使います。

ScreenTopプロパティは現在の画面サイズに基づいてピクセルで計算されます。

これは複数のフォームを利用するアプリケーションでフォームの位置をコントロールする時などによく利用されます。

例は「[ScreenLeft プロパティ](#)」を参照してください。

ShowError プロパティ

ShowErrorはエラー状態のon/offを設定します。

ShowErrorプロパティを使って、コンポーネントのエラー状態の有無を設定します。このプロパティがTrueに設定されていると、コンポーネントの背景色がError BackColorに設定されます。

このコンポーネントがフィールドまたはグリッド・セルの場合、次のような事象によりShowErrorプロパティが暗黙のうちに設定されることに注意してください。

- 妥当性検査
- エラー状態をクリアするオプションを持つイベント・ルーチン

LANSAの妥当性検査の使用法については『LANSA テクニカル リファレンスガイド』またはオンライン・ヘルプを参照してください。

ShowLines プロパティ

ShowLinesはグリッド線を表示するかどうかを決定します。このプロパティをFalseに設定すると、グリッド線が表示されません。

このプロパティをグリッドないで有効にするには、列のSortOnClickプロパティがTrueに設定されていなければなりません。

SelectionStart プロパティ

SelectionStartは選択の開始位置を設定します。

実行時のみのプロパティです。

SelectionStartプロパティを利用して、選択が開始する文字の位置を指定します。

AutoSelectプロパティがTrueに設定されていると、編集ボックスの全ての内容が選択されます。ですから、SelectionStartおよびSelectionEndプロパティを利用する際は、AutoSelectプロパティをFalseに設定しておく必要があることに注意してください。

SelectionEnd プロパティ

SelectionEndは選択の終了位置を設定します。

実行時のみのプロパティです。

SelectionEndプロパティを利用して、選択が終了する文字の位置を指定します。

AutoSelectプロパティがTrueに設定されていると、編集ボックスの全ての内容が選択されます。ですから、SelectionStartおよびSelectionEndプロパティを利用する際は、AutoSelectプロパティをFalseに設定しておく必要があることに注意してください。

ShowMessages メソッド

LANSA製品センター内部専用です。

ShowMessagesSet メソッド

LANSA製品センター内部専用です。

ShowSortArrow プロパティ

ShowSortArrowはソートの方向を示す矢印を表示したり、非表示にしたりします。

ShowSortArrow プロパティを使って、列のヘッダーボタン上のソート方向を示す矢印を表示または非表示にします。

TabPosition プロパティ

TabPositionはフォーカスの順序を設定します。

TabPositionにより、ユーザーがTabキーをした時にどんな順番でコンポーネントがフォーカスされるかを指定します。

あるコンポーネントのTabPositionを1に設定したとすると、実行時ユーザーがTabキーを押したときに、このコンポーネントに最初にフォーカスが当たります。

省略値では、フォームやパーツにコンポーネントをドラッグする際にタブの順序が割り当てられます。新しいコンポーネントが加えられる度に、タブ順序の一番最後に置かれます。あるコンポーネントのTabPositionプロパティを変更すると、他のコンポーネントのタブ順序も調整されます。

メニューとタイマーを除く全てのコンポーネントはタブ順序に含まれます。実行時、表示されないもしくは無効のコンポーネントやフォーカスを受けないコンポーネントは省略されます。

TabStop プロパティ

TabStopはTabキーの使用を制御します。

TabStopプロパティを使って、Tabキーがオブジェクトのフォーカス移動に使用できるかどうかを決定します。このプロパティがTrueに設定されると、Tabキーは無効になります。このプロパティを使って、タブ順序にコンポーネントを追加したり削除したりできます。

Top プロパティ

Topはコンポーネントの上の距離を設定します。

Topプロパティを使って、コンポーネントの上端とコンテナの上端の間の距離をピクセルで指定します。

VerticalAlignment プロパティ

VerticalAlignmentはラベルを縦に整列します。

ラベルを縦に整列するにはVerticalAlignmentプロパティを使用します。
position(位置)はtop(上)、bottom(下)、またはcenter(中央)です。Topが省略値です。

Visible プロパティ

Visibleはコンポーネントの表示/非表示を制御します。

Visibleプロパティを使って実行時にコンポーネントを表示するか、非表示するかを指定します。

コンポーネントを非表示にするにはこのプロパティにFalseを設定します。

コード内でこのプロパティを設定することで特定のイベントに反応してコンポーネントを表示したり、非表示にしたりすることができます。次のコードではプッシュ・ボタン#PHBN_1を非表示にします：

```
SET COM(#PHBN_1) VISIBLE(FALSE)
```

VisualStyle プロパティ

VisualStyleは外観の設定を行います。

VisualStyleプロパティを使用して、コンポーネントの見た目を制御します。

このビジュアル・スタイルはアプリケーションの色、フォントやその他のスタイル要素を構成します。

LANSARiポジトリに定義されているビジュアル・スタイルから選ぶことができます。これには製品と共に出荷される標準スタイルや、ユーザーが定義した場合はそのビジュアル・スタイルも含まれます。

SET例197も参照してください：[ビジュアル・スタイル](#)

VisualStyleEditプロパティ

VisualStyleEdit はコントロールの編集部分の見た目を設定します。

VisualStyleEditプロパティを使用して、コンポーネントの編集部分の見た目を制御します。

このビジュアル・スタイルはアプリケーションの色、フォントやその他のスタイル要素を構成します。

LANSARiポジトリに定義されているビジュアル・スタイルから選ぶことができます。これには製品と共に出荷される標準スタイルや、ユーザーが定義した場合はそのビジュアル・スタイルも含まれます。

SET例197も参照してください：[ビジュアル・スタイル](#)

VisualStyleOfParent プロパティ

VisualStyleOfParentは親のVisualStyleで外観を設定をします。

VisualStyleOfParentプロパティを使って、親コンポーネントのビジュアル・コンポーネントに割り当てられたビジュアル・スタイルに基づいて、コンポーネントの外観を制御します。

Width プロパティ

Width プロパティを利用してコンポーネントの幅をピクセルで指定します。

コンポーネントを選択して、サイズ変更グリップからドラッグすることで幅をビジュアル的に調整することも可能です。ピクセル値は自動的に調整されます。

複数のコンポーネントのサイズを一度に設定する場合は、編集メニューのサイズと整列を利用します。

リスト・ビューとグリッド列の場合、Width プロパティの値はWidthType プロパティの設定によって決まります。WidthType がScaleableまたはfixedの場合、Width プロパティはリストやグリッド内で列が占める割合をパーセンテージで表します。

例えば、リストビューに拡張可能(Scaleable)、もしくは固定(fixed)の同じ大きさの5つの列がある場合、それぞれの列のWidthは20になります。

WidthType がCharacterの場合、Width プロパティでは文字数で列の幅を設定します。WidthType がRemainderの場合、Width プロパティは無視されます。

ReadOnly プロパティ

ReadOnlyはコンポーネントを入力用に使用可能かどうかを制御します。

ReadOnlyを使って、コンポーネントを入力用に使用できるかどうかの制御を行います。このプロパティにTrueを設定すると、コンポーネントが読み取り専用になります。

Text プロパティ

Textには編集エリアに現在表示されているフォーマットされたデータが含まれます。

Textプロパティを使用して、編集エリアに現在表示されているフォーマットされたデータを取り出せます。このプロパティは読み取り専用です。

テキストを設定する、もしくはフォーマットされていないデータを取り出す場合は、編集ボックスのValueプロパティを利用します。

Value プロパティ

Valueプロパティを使って、フィールド・コンポーネント、フィールド・ビジュアル・コンポーネント、基本ビジュアル・コントロール（コンボ・ボックス、編集ボックス、スピン編集、プログレス・バー、トラック・バーなど）、グリッド・セルや基本データ・クラス（PRIM_ALPH、PRIM_NMBR、PRIM_DATEなど）から現在の値を取り出すことができます。

グリッド・セル以外はValueプロパティを使用してこれらのコンポーネントの現在値を変更することができます。

基本コントロールのイベント

Activate イベント	Exit イベント	MouseDown イベント
Changed イベント	Form パラメータ	MouseMove イベント
Click イベント	GotFocus イベント	MouseUp イベント
DoubleClick イベント	Initialize イベント	PrompterActivate イベント
DragDrop イベント	Initialize イベント	PrompterDeactivate イベント
DragOver イベント	KeyPress イベント	StartDrag イベント
EndDrag イベント	LostFocus イベント	

Activate イベント

Activate イベントはコンポーネントがアクティブになると実行されま
す。

コンポーネントが有効になるとActivate イベントが起動されます。

キーボード・フォーカスが設定されるとフォームはアクティブなウィ
ンドウになります。このアクティブ・ウィンドウがデスクトップの最上位
のウィンドウとなり、タイトル・バーと（ある場合は）枠線が強調表示
されます。

ユーザーはこれをクリックする、Alt+TabまたはAlt+Escキーの組み合わ
せもしくはタスクバーより選択することでウィンドウをアクティブにし
ます。

ユーザーがデスクトップ上のウィンドウの間を移動すると、このactivate
イベントが頻繁に発生します。キーボード・フォーカスがある時のみこ
のイベントを使用してフォームでアクションを実行するようにしてくだ
さい。フォームの初期化（リストを埋める、省略値を設定するなど）に
関するアクションは、フォームのInitialize イベントに置くようにしてく
ださい。

Changed イベント

Changedはコンポーネント内の変更です。

コンポーネントのコンテンツが変更されるとChangedイベントが起動されます。

グラフの場合、1つのレコードをロードしている時ではなく、全ての更新が完了した後に、このイベントが起動します。

リスト、ツリービュー、グリッドの場合、（個別の項目で起動される Selected/Focus イベントとは異なり）全ての更新が完了した時に Changed イベントが起動されます。

コンボ・ボックスでは、コンボ・ボックスの編集部分の内容が変更されたときに Changed イベントが起動します。 ComboChanged イベントは全ての更新が完了した時に起動します。

Click イベント

Clickはマウスのクリックです。

ユーザーがコンポーネント上で、マウス・ボタンを押し、手を離すと、clickイベントが起動されます。コンポーネントの値が変更されたときにもclickは起動されます。

Clickイベントの操作にはコードを書きます。例えば次のClickイベントのイベント・ルーチンでは、プッシュ・ボタン1(#PHBN_1)はプッシュ・ボタン2を非表示にします。

```
EVTROUTINE HANDLING( #PHBN_1.Click )
SET COM(#PHBN_2) VISIBLE(FALSE)
ENDROUTINE
```

フォームの場合、ユーザーがブランク領域または無効のコンポーネントでクリックすると、Clickイベントが起動されます。

その他のコンポーネントでは、ユーザーが左または右のマウス・ボタンをコンポーネントでクリックすると、このイベントが起動されます。チェック・ボックス、プッシュ・ボタン、ラジオ・ボタンでは、ユーザーが左のマウス・ボタンをクリックした時のみClickイベントが起動されます。

コンボ・ボックスやリスト・ボックス・コントロール内の項目をユーザーが矢印キーを押したりマウス・ボタンをクリックして選択した場合にもClickイベントが発生します。

同様に、プッシュ・ボタン、ラジオ・ボタンやチェックボックスがフォーカスされている時にユーザーがスペースバーを押した時や、フォームにプッシュ・ボタンがあり、ButtonDefaultプロパティがTrueに設定されている時にユーザーがEnterキーを押した時、更にフォームにプッシュ・ボタンがあり、ButtonCancelプロパティにTrueが設定されている時にユーザーがEscキーを押した時にもClickイベントが発生します。

ユーザーがコンポーネントのアクセス・キーを押した時にもClickイベントが起動します。例えば、プッシュ・ボタン・コンポーネントのキャプションが"&&Go"だった場合、ALT+Gを押すことでこのイベントが発生します。

リスト・ビューとグリッド列がClickイベントを受け取れることに注意してください。ユーザーが列ヘッダーまたはグリッド・ボタン上でクリッ

クするとこのイベントが起動します。Clickイベントは通常SortPositionやSortDirectionプロパティを変更するために利用され、ユーザーがリストを列でソートすることができます。このイベントを受け取るには、リスト・ビューのColumnHeadersプロパティがTrueに、またグリッドはColumnHeadersプロパティがTrueに設定されていなければなりません。

DoubleClick イベント

DoubleClickとはマウスを2度クリックすることです。

ユーザーがコンポーネント上でマウス・ボタンを押して離す、そして再び押して離すと、DoubleClickが起動します。

フォームの場合、ユーザーがブランク領域または無効のコンポーネントでダブル・クリックすると、DoubleClickイベントが起動されます。

リスト・ビューとグリッド列がDoubleClickイベントを受け取れることもできることに注意してください。このイベントは、ユーザーが列ヘッダーをダブル・クリックしたときに起動されます。このイベントを受け取るには、リスト・ビューのColumnHeadersプロパティがTrueに、またグリッドはColumnHeadersプロパティがTrueに設定されていなければなりません。

Exit イベント

Exit イベントはコンポーネントのフォーカスが失われた時に起動されま
す。

別のオブジェクトをクリックしたり、別のタブにするなどのユーザーア
クションにより、もしくはコード内でSetFocusメソッドを利用して
フォーカスを変更することで、フォーカスは移動します。

StartDrag イベント

StartDragとはマウスを押したまま移動することです。

コンポーネント上でマウス・ボタンが押され、マウスが移動した時にStartDragイベントが起動されます。

ドラッグ・アンド・ドロップ操作の例に関しては、SET例223および『Visual LANSA 開発者ガイド』を参照してください。

[StartDragEvent Source](#)

[DragDrop Aggregated Source](#)

[StartDragEvent Payload](#)

[StartDragEvent Continue](#)

[StartDragEvent DragList](#)

StartDragEvent Source

StartDragEvent操作のSourceとは、その操作が開始するコンポーネントのことです。

DragDrop Aggregated Source

DragDrop操作のAggregatedSourceとは、DragDrop操作のSourceを含む集約チェーンの上にあるコンポーネントです。

SourceコンポーネントのDragStyleがAutomaticの場合、AggregatedSourceコンポーネントとSourceコンポーネントは同じになります。

SourceコンポーネントのDragStyleがAggregatedの場合、AggregatedSourceコンポーネントはSourceコンポーネントの親のどれか1つになります。

StartDragEvent Payload

Payloadはドラッグしてドロップされたコンポーネントです。

Payloadとはドロップ・ターゲットが次の事項を判断できるようにするものです。

- ドロップ・ターゲットがドラッグされるものをサポートしているかどうか（これを可能にするには、一般的にpayload内にドロップ・サイトがテストできる指標のようなものがが必要です。この指標は単なる1つのテキストや数字であることもあるでしょうし、IF_REF経由でpayloadのクラスをテストするといった複雑なものもあります。）
- いったんドロップされると、このドロップ操作を処理するためにドロップ・ターゲットにどのようなデータが必要なのか。これを可能にするには、Payloadにドラッグされるオブジェクトのキーを含む単純な値、作業リスト、もしくはVisual LANSAコンポーネントのキー付きコレクションなどを含めます。

StartDragEvent Continue

このパラメータにTrueを設定すると、ドラッグ操作が進められます。

StartDragEvent DragList

処理のパラメータです。

Draglistは、ドラッグ操作が起きている時に表示されるイメージです。

DragListコンポーネントのヘルプを表示させるには、左側のツリーのDragListでダブル・クリックします。

DragDrop イベント

DragDropとはコンポーネントをドロップすることです。

DragDropは、フォームまたは別のコンポーネント上でコンポーネントがドラッグされ、マウス・ボタンが解放された時に起こります。

DragDropイベント・ルーチンを使用してドラッグ操作が完了した後に何をするかを指定します。例えば、ソース・コンポーネントを新しい場所に移動することもできますし、ある場所から別の場所にファイルをコピーすることもできます。

ドラック・アンド・ドロップの例については『Visual LANSA 開発者ガイド』を参照してください。

[DragDrop PosX](#)

[DragDrop PosY](#)

[DragDrop Source](#)

[DragDrop Aggregated Source](#)

[DragDrop Payload](#)

DragDrop PosX

PosXはX座標に対するマウスの位置を決定します。

DragDrop PosY

PosYはY座標に対するマウスの位置を決定します。

DragDrop Source

DragDrop操作のSourceとは、その操作が開始するコンポーネントのことです。

DragDrop Aggregated Source

DragDrop操作のAggregatedSourceとは、DragDrop操作のSourceを含む集約チェーンの上にあるコンポーネントです。

SourceコンポーネントのDragStyleがAutomaticの場合、AggregatedSourceコンポーネントとSourceコンポーネントは同じになります。

SourceコンポーネントのDragStyleがAggregatedの場合、AggregatedSourceコンポーネントはSourceコンポーネントの親のどれか1つになります。

DragDrop Payload

Payloadはドラッグしてドロップされたコンポーネントです。

Payloadとはドロップ・ターゲットが次の事項を判断できるようにするものです。

- ドロップ・ターゲットがドラッグされるものをサポートしているかどうか（これを可能にするには、一般的にpayload内にドロップ・サイトがテストできる指標のようなものがが必要です。この指標は単なる1つのテキストや数字であることもあるでしょうし、IF_REF経由でpayloadのクラスをテストするといった複雑なものもあります。）
- いったんドロップされると、このドロップ操作を処理するためにドロップ・ターゲットにどのようなデータが必要なのか。これを可能にするには、Payloadにドラッグされるオブジェクトのキーを含む単純な値、作業リスト、もしくはVisual LANSAコンポーネントのキー付きコレクションなどを含めます。

DragOver イベント

DragOverとはコンポーネント上でドラッグすることです。

これはドラッグ・アンド・ドロップの進行中に起こります。

このイベントを使ってマウス・ポインターの出入りや、マウス・ポインターが有効なターゲットのすぐ上で停止するのをモニターすることができます。マウス・ポインターの位置がこのイベントを受け取るターゲット・オブジェクトを決定します。

ドラッグ・アンド・ドロップの例は『*Visual LANSa* 開発者ガイド』の「[ドラッグ・アンド・ドロップ](#)」を参照してください。

[DragOverEvent PosX](#)

[DragOverEvent PosY](#)

[DragOverEvent Source](#)

[DragDrop Aggregated Source](#)

[DragOverEvent Payload](#)

[DragOverEvent AcceptDrop](#)

[DragOverEvent ShowDropHighlight](#)

[DragOverEvent DragCursor](#)

[DragOverEvent DragState](#)

DragOverEvent PosX

PosXはX座標に対するマウスの位置を決定します。

DragOverEvent PosY

PosYはY座標に対するマウスの位置を決定します。

DragOverEvent Source

DragOverEvent操作のSourceとは、その操作が開始するコンポーネントのことです。

DragOverEvent Payload

Payloadはドラッグしてドロップされたコンポーネントです。

Payloadとはドロップ・ターゲットが次の事項を判断できるようにするものです。

- ドロップ・ターゲットがドラッグされるものをサポートしているかどうか（これを可能にするには、一般的にpayload内にドロップ・サイトがテストできる指標のようなものがが必要です。この指標は単なる1つのテキストや数字であることもあるでしょうし、IF_REF経由でpayloadのクラスをテストするといった複雑なものもあります。）
- いったんドロップされると、このドロップ操作を処理するためにドロップ・ターゲットにどのようなデータが必要なのか。これを可能にするには、Payloadにドラッグされるオブジェクトのキーを含む単純な値、作業リスト、もしくはVisual LANSACコンポーネントのキー付きコレクションなどを含めます。

DragOverEvent AcceptDrop

このパラメータにTrueやFalseを設定して、ドラッグされたオブジェクトを受け取ったり、拒否したりできます。

DragOverEvent ShowDropHilght

このパラメータにTrueを設定すると、ドラッグされている項目を表示されます。

DragOverEvent DragCursor

DragCursorパラメータを使ってドラッグ操作中に表示されるカーソルを指定します。

DragOverEvent DragState

DragStateはDragStateイベントが起動された理由を示します。このパラメータは、次のいずれかの値を取ることができます：

DsEnter: ドラッグ操作中にマウス・ポインターがコンポーネントに入ってきたことを指します。これは常に一番始めに起こります。

DsMove: マウスが移動されたけれども、コンポーネントの範囲内に留まったことを示します。

DsHover: マウスは移動していないけれども、ドラッグにより管理される間隔の期限が過ぎ、ドラッグ・マネージャがコンポーネントにユーザーが上で止まっている状態であることを知らせます。一般的にこれはスクロールが必要なコンポーネントで利用されます。コンボ・ボックスではドロップダウンを開き、ツリービューではマウスが止まっている項目の子を展開します。

DsExit: マウスがコンポーネント外に出たことを示します。

EndDrag イベント

EndDragはドラッグが終了したことを意味します。

ドラッグ操作の最後にマウス・キーが解放された時にEndDragイベントが起動します。ドラッグの開始はStartDragイベントにより信号が送られます。

EndDragイベントは通常ドラッグ操作のソースにこの操作が終了したことを通知するために利用されます。

ドラッグ・アンド・ドロップ操作の例に関しては、SET例223および『Visual LANSA 開発者ガイド』を参照してください。

EndDragEvent Source

EndDragEvent操作のSourceとは、その操作が終了するコンポーネントのことです。

DragDrop Aggregated Source

DragDrop操作のAggregatedSourceとは、DragDrop操作のSourceを含む集約チェーンの上にあるコンポーネントです。

SourceコンポーネントのDragStyleがAutomaticの場合、AggregatedSourceコンポーネントとSourceコンポーネントは同じになります。

SourceコンポーネントのDragStyleがAggregatedの場合、AggregatedSourceコンポーネントはSourceコンポーネントの親のどれか1つになります。

EndDragEvent Payload

Payloadはドラッグしてドロップされたコンポーネントです。

Payloadとはドロップ・ターゲットが次の事項を判断できるようにするものです。

- ドロップ・ターゲットがドラッグされるものをサポートしているかどうか（これを可能にするには、一般的にpayload内にドロップ・サイトがテストできる指標のようなものがが必要です。この指標は単なる1つのテキストや数字であることもあるでしょうし、IF_REF経由でpayloadのクラスをテストするといった複雑なものもあります。）
- いったんドロップされると、このドロップ操作を処理するためにドロップ・ターゲットにどのようなデータが必要なのか。これを可能にするには、Payloadにドラッグされるオブジェクトのキーを含む単純な値、作業リスト、もしくはVisual LANSコンポーネントのキー付きコレクションなどを含めます。
- [EndDragEvent Source](#)
- [DragDrop Aggregated Source](#)
- [EndDragEvent Payload](#)
- [EndDragEvent DragResult](#)

EndDragEvent DragResult

このパラメータはドラッグ操作の結果を示します。このパラメータは、次のいずれかの値を取ることができます：

Accepted - ドロップが成功したことを示します。

Ended - ドロップはされたけれども、受け入れられた場所と違う場所で起こったことを意味します。

Cancelled - ドラッグが何かによって、例えばドラッグの最中にAlt+Tabキーが押されるなどにより、中断されたことを示します。

DragDrop Aggregated Source

DragDrop操作のAggregatedSourceとは、DragDrop操作のSourceを含む集約チェーンの上にあるコンポーネントです。

SourceコンポーネントのDragStyleがAutomaticの場合、AggregatedSourceコンポーネントとSourceコンポーネントは同じになります。

SourceコンポーネントのDragStyleがAggregatedの場合、AggregatedSourceコンポーネントはSourceコンポーネントの親のどれか1つになります。

Initialize イベント

Initializeはコンポーネントがロードされる時に起動されます。

コンポーネントが最初に表示されるとInitializeイベントが起動されます。例えばリストやツリー・ビューが最初に表示された時に、このイベントを使ってデータをロードします。

InitializeイベントはCreateInstanceイベントの後に起動されます。

CreateInstanceイベントはコンポーネントが作成される時に、Initializeイベントは最初に情報が表示された時に起動されます。多くの場合、この違いを区別する必要はありません。しかしながら、非ビジュアル・コンポーネントを扱う時や表示前にコンポーネントで情報を処理したい時などは、CreateInstanceイベントを利用します。

Form パラメータ

Formパラメータを使って、プロンプタ・フォームの名前を指定します。

GotFocus イベント

GotFocusはコンポーネントがアクティブになる時に起動されます。

デスクトップ上でコンポーネントがフォーカスを受けると、GotFocusイベントが起動されます。

ユーザーはマウスのクリックまたはTabキーを使って、コンポーネントにフォーカスを設定できます。

フォームはそのコントロールが無効になっている時のみフォーカスを受けることができます。

SetFocusメソッドを利用してプログラム上でフォーカスを設定することもできます。

LostFocus イベント

LostFocusはコンポーネントがアクティブでなくなった時に起動されます。

デスクトップ上でコンポーネントがフォーカスを失うと、LostFocusイベントが起動されます。

ユーザーはマウスのクリックまたはTabキーを使って、コンポーネントにフォーカスを移動できます。

フォームはそのコントロールが無効になっている時のみフォーカスを受けることができます。

SetFocusメソッドを利用してプログラム上でフォーカスを設定することもできます。

MouseDown イベント

MouseDownはマウス・ボタンが押されたことを意味します。

ユーザーがマウス・ボタンを押すとMouseDownイベントが起動されます。

MouseDownやMouseUpイベント処理を使って、指定のマウス・ボタンが押されたり、解放されたりした時のアクションを指定します。ClickやDoubleClickイベントとは異なり、MouseDownとMouseUpイベントはマウス・ボタンの左、右、中央の区別ができます。またShift、Ctrl、Altなどのキーボード修飾子を使うマウスとキーボードの組み合わせにもコードを書くことができます。

MouseUp イベント

MouseUpはマウス・ボタンが解放されたことを意味します。

ユーザーがマウス・ボタンを解放するとMouseUpイベントが起動されます。

MouseDownやMouseUpイベント処理を使って、指定のマウス・ボタンが押されたり、解放されたりした時のアクションを指定します。ClickやDoubleClickイベントとは異なり、MouseDownとMouseUpイベントはマウス・ボタンの左、右、中央の区別ができます。またShift、Ctrl、Altなどのキーボード修飾子を使うマウスとキーボードの組み合わせにもコードを書くことができます。

MouseMove イベント

MouseMoveはマウスの動作のことです。

ユーザーがマウスを移動するとMouseMoveが起動します。

MouseMoveイベントはコンポーネント上をマウス・ポインタが移動するのに合わせて絶えず起動されます。別のコンポーネントがこのマウスを捕えない限り、その枠線内にマウス位置がある時はコンポーネントが常にMouseMoveイベントを認識します。

ドラック・アンド・ドロップの例については『Visual LANSA 開発者ガイド』を参照してください。

PrompterActivate イベント

PrompterActivateイベントは、ユーザーがプロンプター・ボタンをクリックしたときに起動されます。

このイベントを利用して、表示前にプロンプター・フォームにアクセスして、エントリーをフィルターするなどの処理を行います。

PrompterDeactivateイベント

PrompterDeactivateイベントはプロンプタ・フォームが閉じられようとする時に起動します。

このイベントを利用して、閉じらる前にプロンプタ・フォームにアクセスします。

KeyPress イベント

KeyPressはコントロールにフォーカスがある間にキーボードのキーが押された時に起動されます。

印字可能な全ての文字のキー入力によりKeyPressイベントが起動されます。さらに、[KeyCodes](#)にリストされているいずれのキーもこのイベントを起動することができます。

KeyPressは他の処理前にキーが押されるたびに起動されます。このイベントにより、アプリケーションでキーが押されたことや、別のキーが押されたことをチェックすることができます。

例

次の例ではEnterキーが押された時のみ、処理を行います。

```
Evtroutine Handling(#EDIT_1.KeyPress)

Options(*NOCLEARMESSAGES *NOCLEARERRORS) Com_Sender(#cur_
* Process when the 'Enter' key is pressed
If Cond('#KeyCode = Enter')
Invoke Method(#com_owner.ProcessOnEnterKey)
Endif
Endroutine
```

次の例ではAという文字がタイプされると全て感嘆符に変更されます。

```
Evtroutine Handling(#EDIT_2.KeyPress)

Options(*NOCLEARMESSAGES *NOCLEARERRORS) Com_Sender(#cur_
* Change all "a" key presses to "!"
If Cond("#char = \"a\"")
Set Com(#char) Value('!')
Endif
Endroutine
```

[KeyCode](#) パラメータ

[Char](#) パラメータ

[IsAltDown](#) パラメータ

[IsControlDown](#) パラメータ

[IsShiftDown](#)パラメータ

Handled パラメータ

KeyCodes

Enter	Escape	F12	Plus	Subtract
BackSpace	Insert	F13	Minus	Divide
UpArrow	F1	F14	Add	Decimal
DownArrow	F2	F15	Application	NumPad0
LeftArrow	F3	F16	LeftWindows	NumPad1
RightArrow	F4	F17	RightWindows	NumPad2
Tab	F5	F18	CapsLock	NumPad3
BackTab	F6	F19	NumLock	NumPad4
Space	F7	F20	ScrollLock	NumPad5
Home	F8	F21	Pause	NumPad6
End	F9	F22	Print	NumPad7
PageUp	F10	F23	Help	NumPad8
PageDown	F11	F24	Multiply	NumPad9
Delete				

KeyPress イベント

KeyCode パラメータ

KeyCode パラメータ

KeyCodeパラメータは押されたキーがどんな種類のキーかを返します。
この値には次のようなものがあります。

IsChar

および[KeyCodes](#)にリストされているすべてのキー

Char パラメータ

CharパラメータはKeyCodeがsCharだった場合にその文字を返します。

IsAltDown パラメータ

IsAltDownパラメータは常にFalseです。

IsControlDown パラメータ

IsControlDownパラメータは常にFalseです。

IsShiftDownパラメータ

IsShiftDownパラメータは、Shiftキーが押されるとTrueに設定されます。

Handled パラメータ

HandledパラメータをTrueに設定すると、KeyPressが処理され、コントロールはキー入力进行处理する必要がないことを示します。

基本コントロールのメソッド

UpdateDisplay メソッド Realize メソッド
SetFocus メソッド Unrealize メソッド

UpdateDisplay メソッド

UpdateDisplayは表示を即座に更新します。

UpdateDisplayメソッドを使用して、コンポーネントの表示を即座に更新します。このビジュアル・コンポーネントが別のコンポーネントの親である場合は、このコンポーネントも更新されます。例えば、グループ・ボックスやパネルにフィールドがある場合、表示がUpdateDisplayを使って更新されると、そのフィールドも更新されます。

このメソッドは、例えばグリッドをロードして、100レコード追加されるごとにグリッドの表示を更新する時などに使用すると便利です。ただしこれによりアプリケーションの速度が落ちるので、このメソッドをいつ、どこで使用するかを注意深く検討する必要があります。

次のイベント・ルーチンは#Num_1と#Num_2という2つのフィールドの数値を増やしていきます。この2つのフィールドは最初はnullに設定され、ループ・カウンターとして使用されるフィールド#Std_Numは500に設定されます。ループ内ではフィールドの値が加算されます。フィールド#Num_1は数値が加算されるにつれ動的に表示が更新されます。フィールド#Num_2はイベント・ルーチンが終了した時だけ更新されません。

```
EVTROUTINE HANDLING(#PHBN_1.Click)
Change Field(#Num_1 #Num_2) To(*Null)
Change #Std_Num 500
begin_loop to(#Std_Num)
Change #Num_1 '#Num_1 + 1'
Change #Num_2 '#Num_2 + 2'
invoke #Num_1.UpdateDisplay
end_loop
ENDROUTINE
```

SetFocus メソッド

SetFocusはコンポーネントにフォーカスを設定します。

SetFocusメソッドはプログラムでコンポーネントにフォーカスを設定するのに使用します。（ユーザーはキーボードやマウスを使ってフォーカスを設定できます。）

SetFocusメソッドの構文は、以下のとおりです：

```
INVOKE #PHBN_1.SetFocus
```

Realize メソッド

Realizeメソッドはコンポーネントを動的に作成します。

Realizeメソッドを使って、動的にコンポーネントを表示させます。通常このメソッドはコンポーネントのコレクションと共に使用され、コレクションのインスタンスを動的に実現化します。

例えば、フィールド#EMPNOの値をキーにしたチェック・ボックスのコレクションを定義し、社員ごとにチェックボックスを作成することができます。このためには、以下のように定義します：

```
DEFINE_COM CLASS(#PRIM_KCOL) NAME(#KCOL_1) COLLECTS(#PI
```

そして、このコレクションのプロパティを実行時に設定できます。この例では、チェック・ボックスはフィールド#SURNAMEの現在値がキャプションとして利用されます。

```
Set COM(#KCOL_1<#EMPNO>) Parent(#COM_OWNER) Caption(#SURN/
```

次にこのチェック・ボックスを表示します。

```
Invoke #KCOL_1<#EMPNO>.Realize
```

多くのビジュアル・コンポーネントを作成する場合、それぞれ個別に実行するのではなく、新しいコンポーネント全てを含むフォームの ShowFormメソッドを起動するだけでよいことに注意してください。 ShowForm/ActivateForm/MaximizeFormなどのメソッドは全て、実際の要求処理を実行する前に全てのコンテンツが実現化されていることを確認します。

メニュー項目はその親が設定されると実現化されることにも注意してください。メニュー・バーはフォームに割り当てられてそのフォームが実現化された時にのみ実現化されます。ポップアップ・メニューは最初に必要となった時に実現化されます。サブ・メニューは関連するメニュー項目が実現化されると、実現化されます。

Unrealize メソッド

Unrealizeはコンポーネントを取り除きます。

コンポーネントを除去するのにUnrealizeメソッドを使用します。通常は、Realizeメソッドを利用して動的に作成されたコレクションを削除する際にこのメソッドを使います。

コンテナ・コンポーネント

コンテナ・コンポーネント・プロパティ

コンテナ・コンポーネント

システム内部でのみ使用

コンテナ・コンポーネント・プロパティ

[ComponentControls](#) プロパティ

[EnableChildren](#) プロパティ

[LayoutManager](#) プロパティ

ComponentControlsプロパティ

ComponentControlsはこのコンポーネントに含まれる全てのコントロールのコレクションです。

ComponentControlsプロパティにより、このコンポーネントに含まれる全てのコントロールのコレクションにアクセスできます。

ComponentControlsを使って、全てのコントロールに共通のプログラミングをすることができます。

EnableChildren プロパティ

EnableChildrenは親がEnabledプロパティを設定するかどうかを指定します。

EnableChildrenプロパティを利用して、コンポーネントの親がそのコンポーネント内に含まれるコンポーネントのEnabledプロパティを設定できるかどうかを指定します。

このプロパティがTrueに設定されていると、このコンポーネントのEnabledプロパティの設定(TrueまたはFalse)により、このコンポーネント内に含まれるすべてのコンポーネントが有効または無効になります。

多くの場合、コンポーネント内のEnabledプロパティをそれぞれ個別に設定するより、親コンポーネントのEnabledプロパティで設定する方がコードの効率が良く、管理し易いです。

LayoutManager プロパティ

LayoutManager プロパティを使用して、フォーム、グループ・ボックス、パネルやタブ・シートといったコンポーネントをコンテナ上に位置付けます。レイアウト管理の選択ダイアログ・ボックスからレイアウト・マネージャの1つを選択することで、レイアウト・マネージャを設定できます。使用可能なレイアウト・マネージャは以下のとおりです：

添付：添付レイアウトは、コンテナを5つのセルに分割します。中央のセルは、使用可能なスペースをすべて埋めることができます。

フロー・レイアウト：フローレイアウト・マネージャは、アイテムを行または列に分割し、アイテムを適切な行/列に次々と移動させ、コンポーネントを表示エリアに配置します。

Direction プロパティにより行と列の振り分けが制御され、ItemsPerDivision プロパティにより行/列の数が制御されます。

グリッド・レイアウト：グリッド・レイアウト・マネージャは行や列のグリッドのコンポーネントを位置付けるのに使用します。

分割レイアウト：分割レイアウト・マネージャは、コンポーネントの子を横または下の表示エリアに分割されるよう、コンポーネントの表示エリアを管理します。Orientation プロパティは、分割の方向を指定します。オプションとして、コンポーネントのサイズが変更されるとこのレイアウト・マネージャにより管理される項目のサイズを自動的に変更し、分割サイズ調整を表示してユーザーが自分の好みに合わせて分割を調整できるようにすることも可能です。

各レイアウト・マネージャには、プロパティがあります。レイアウト・マネージャのプロパティの説明を見るには、コンポーネントのLayoutManager プロパティの値としてプロパティを選択し、アウトライントップを表示、レイアウト・マネージャを選択し、機能ヘルプで確認します。

プッシュ・ボタン

プッシュ・ボタン・プロパティ

プッシュ・ボタンは、ユーザーがアクションを実行できるようにします。プッシュ・ボタンにテキストを表示するには、Captionプロパティを設定します。ボタンのコードはClickイベントに書き込みます。

ユーザーがEnterを押した時のボタンのClickイベントを起動するには、ButtonDefaultプロパティをTrueに設定します。ユーザーがEscキーを押すことでボタンを選択できるようにするには、Cancel propertyをTrueに設定します。

この他にユーザーがアクションを実行できる方法が2つあります：メニュー項目とツールバー・ボタンです。

プッシュ・ボタン・プロパティ

ButtonCancel プロパティ

ButtonDefault プロパティ

ImagePosition プロパティ

Menu プロパティ

MenuPosition プロパティ

MenuSeparator プロパティ

ButtonCancel プロパティ

ButtonCancelはEscキーの使用を可能にします。

ButtonCancelにより、ユーザーはEscキーを押すことでこのボタンがクリックできるようにします。

ユーザーがEscキーを押すと、ButtonCancelプロパティがTrueに設定されたプッシュ・ボタンのClickイベントのコードが実行されます。通常、変更をキャンセルするプッシュ・ボタンのこのプロパティにTrueを設定します。このプロパティをTrueに設定できるのは1つのボタンだけです。

ButtonDefault プロパティ

ButtonDefaultはEnterキーの使用を可能にします。

ButtonDefaultにより、ユーザーはEnterキーを押すことでこのボタンがクリックできるようにします。

ユーザーがEnterキーを押すと、ButtonDefaultプロパティがTrueに設定されたプッシュ・ボタンのClickイベントのコードが実行されます。通常ユーザーがよく使用するプッシュ・ボタンのこのプロパティにTrueを設定します。このプロパティをTrueに設定できるのは1つのボタンだけです。

ユーザーが同じフォーム上の別のプッシュ・ボタンにフォーカスを移動した場合、Enterキーは省略値のプッシュ・ボタンではなくフォーカスのあるプッシュ・ボタンを選びます。

ImagePosition プロパティ

ImagePositionはキャプションに関連したイメージの位置を設定します。ImagePositionプロパティを利用して、ボタンのキャプションに関するイメージの位置を設定します。

このプロパティはAutoSizeプロパティがTrueに設定されている場合は無視されることに注意してください。

Menu プロパティ

Menuオプションを使ってボタンに関連付けられたポップアップ・メニューを指定します。

通常はツールバーのメニューと共にボタンを使用します。

ポップアップ・メニューはpop-up menuコンポーネントを使用して作成します。

MenuPosition プロパティ

MenuPositionプロパティを使用してボタンの右もしくは下にメニューを表示するかどうかを指定します。

MenuSeparator プロパティ

MenuSeparator プロパティを使って、メニューと残りのボタンを線で分けるかどうかを指定します。

ラジオ・ボタン

ラジオ・ボタンを使って、互いに排反する選択グループを表示します。
ButtonChecked プロパティを使用して、ボタンの選択状態を制御します。

```
Set #rdbn_1 ButtonChecked(True)
This code checks whether the radio button is checked:
If '#rdbn_1.ButtonChecked = True'
Change #std_text 'Selected'
Endif
```

ラジオ・ボタンは、ユーザーがグループの中から1つしか選択できない場合に使用します。

ラジオ・ボタンとチェック・ボックスは機能は似ているように見えますが、次のような重要な違いがあります：ユーザーがあるラジオ・ボタンを選択すると、同じグループの他のラジオ・ボタンは自動的に選択されていない状態になります。これとは対照にチェック・ボックスはいくつでも選択できます。

ラジオ・ボタンの高度な例は、SET例65を参照してください。

[ButtonChecked プロパティ](#)

ButtonChecked プロパティ

ButtonCheckedはラジオ・ボタンが選択されているかどうかを示します。Trueはボタンが選択されていることを示し、False（省略値）はボタンが選択されていないことを示します。

チェック・ボックス

チェック・ボックスは、Yes/Noの選択を表すために使用されます。

チェック・ボックスを利用してYes/Noの選択を表します。通常チェック・ボックスは、2つの値から1つを選ぶフィールドの表示スタイルとして使用されます。別の目的で使用する場合は、他により良いコントロールはないかよく考えてください。

チェック・ボックスは選択されると、チェック・マークが表示されます。チェック・ボックスがクリアされると、チェック・マークは消されます。ButtonStateプロパティを使ってボックスが選択されたかどうかを確認し制御します。

```
if cond('#ckbx_1.ButtonState *eq Unchecked')
  change #std_bool 'False'
else
  change #std_bool 'True'
endif
```

チェック・ボックスのグループを使用することで、複数の選択肢が表示され、ユーザーは1つまたは複数の選択ができます。（相互排他的な選択の表示にはラジオ・ボタンを使用します。）

[ButtonState プロパティ](#)

ButtonState プロパティ

ButtonStateプロパティは、チェック・ボックスがチェックされているかどうかを示します。

ボタンの状態には、checked、unchecked、grayedの3つがあります。省略値はuncheckedです。

クリックすることでユーザーがこの値を変更できるようにするには、コンポーネントのClickイベントでcheckedとuncheckedの値を設定します。

通常grayedの状態はこのチェック・ボックスの選択が競合する設定であることを示します。

ツールバー・ボタン

ツールバー・ボタン・プロパティ

ツールバーのショートカット・ボタンです。

ツールバー（スピード）ボタンは、ユーザーがアクションを早く実行できるようにします。

アプリケーションで頻繁に使用する大部分のアクションでツールバー・ボタンを利用します。アクションを起動する唯一の手段がツールバー・ボタンにならないようにしてください。全てのアクションはメニューからもアクセスできるようになっていなければなりません。

ツールバー作成を開始するには、パネル・コンポーネント(PRIM_PANL)をフォームにドラッグし、サイズ調整します。通常、ツールバー・パネルは、フォーム・メニュー・バーの下に置かれます。

ツールバー・ボタンをグループ化する場合は、別のパネルをツールバー・パネルに追加できます。

ツールバーを作成するには、ツールバー・ボタンをパネルにドラッグします。リポジトリ登録済みのビットマップを指定して表示するには、ボタンのImageプロパティを使用します。Captionプロパティを使用してボタンにテキストを追加したり、Hintプロパティでそのヒントを記述することもできます。

ツールバー・ボタンのサイズは、ビットマップのサイズに調整されます。通常、ツールバー・ボタンで使用されるビットマップは、16 x 16ピクセルです。

ツールバー・ボタンの間に区切りを作成したい場合は、ボタンを追加して、ButtonStyleをSeparatorに設定します。

動的にツールバーを作成する方法の例については、SET例70を参照してください。

ツールバー・ボタン・プロパティ

ButtonStyle プロパティ

GroupIndex プロパティ

Check プロパティ

AllowAllUp プロパティ

ComponentVersion プロパティ

ButtonStyle プロパティ

ButtonStyleプロパティはボタンを3Dまたは平らにします。

ButtonStyleプロパティを使ってツールバー・ボタンを立体ボタン(Button)またはフラット・ボタン(FlatButton)として表示します。フラット・ボタンのボーダーはカーソルがボタン上に移動するまでは表示されません。

ボタンの間に区切りを作成したい場合は、ボタンを追加して、ButtonStyleをSeparatorに設定します。こうすると、このボタンは縦の線として表示されます。ButtonStyleにSeparatorを設定したボタンは単に目印として使用されるだけで、テキストやイメージを含むことはできませんし、イベントに応答することもできません。

GroupIndex プロパティ

ツールバー・ボタンのGroupIndexプロパティを使用して、ひとまとめでのみ選択する（押す）ことができるボタンのグループを設定します。

GroupIndexの省略値は0で、グループがないことを意味します。グループを形成するには、グループ内全てのボタンのGroupIndexを同じ数字に設定します。例えば、ボタンのグループが2つある場合、最初のグループのボタンのGroupIndexを1に、2つめのグループのボタンのGroupIndexを2に設定します。

このグループ・インデックスはツールバー・ボタンの親に固有のもので、す。ですから、例えばツールバーが2つのパネルから成る場合、両方のパネルでGroupIndexが1のグループ・ボタンを設定できます。

グループ内の全てのボタンの選択を同時に解除できるようにするには、全てのボタンのAllowAllUpプロパティをTrueに設定します。

（ユーザーがクリックすると下に押されたままになる）切り替えボタンを1つ作成するには、そのボタンに独自のグループ・インデックスを付けます。もしくはボタンのCheck プロパティを使用します。

Check プロパティ

ボタンのCheckプロパティを使用して、ボタンの選択や選択解除、またボタンが選択されているかどうかを確認できます。 CheckプロパティがTrueに設定されていると、そのボタンは選択されて（押されて）います。

Clickイベント内でボタンのCheckプロパティを利用して、切り替えボタン（ユーザーがクリックすると下に押されたままになる）を作成することも可能です。 もしくはボタンのGroupIndexを使用しても作成可能です。

AllowAllUp プロパティ

AllowAllUpプロパティを使用して、同じGroupIndexの全てのボタンのチェックを同時に外すことができるかどうかを指定します。

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 11.0 以前のバージョンです。

1 はLANSA 11.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

ComponentVersionが1の場合、ツールバー・ボタンにAutoSize (False) 設定されていると、ツールバーの境界より大きなイメージはサイズ調整できません。

スタティック

スタティックのプロパティ

線と形です。

スタティック・コンポーネントを利用して、フォームに線や形を描きます。

スタティックのプロパティ

DrawType プロパティ

FrameHeightBottom プロパティ

FrameHeightTop プロパティ

FrameWidthLeft プロパティ

FrameWidthRight プロパティ

LineType プロパティ

LineWidth プロパティ

DrawType プロパティ

DrawTypeは線や形のタイプを指定します。

DrawTypeプロパティを利用して作成したい線や形の種類を指定します。

使用できる形は次の通りです：

線： 横線(horizontal)か縦線(vertical)です。また、平ら(plain)やとがった(edged)線にもなります。

バー： 浮き出た(raised)またはくぼんだ(sunken)バーです。

フレーム： 浮き出た(raised)またはでこぼこ(indented)の枠です。

楕円： 楕円(Elliptical shape)や円(circle)です。

長方形： 簡素な(plain)長方形または枠付きの(edged)長方形にもなります。

三角形： 三角形の角を異なる方向に向けることができます。

FrameHeightBottom プロパティ

FrameHeightBottom

FrameHeightBottom プロパティを利用してフレームの下の高さを制御します。

FrameHeightTop プロパティ

FrameHeightTopはフレームの上部を制御します。

FrameHeightTopプロパティを利用してフレームの上の高さを制御します。

FrameWidthLeft プロパティ

FrameWidthLeftはフレームの左側を制御します。

FrameWidthLeftプロパティを利用してフレームの左側の幅を制御します。

FrameWidthRight プロパティ

FrameWidthRightはフレームの右側を制御します。

FrameWidthRightプロパティを利用してフレームの右側の幅を制御します。

LineStyle プロパティ

LineStyleでは線をどのように描くかを指定します。

LineStyleプロパティを使用して、線の引き方を指定できます。

例えば、破線(dashed)、点線(dotted)、実線(solid)などがあります。

LineWidth プロパティ

LineWidthは線の幅を制御します。

LineWidthプロパティを利用して線の幅をピクセルで指定します。

ラベル

テキスト記述です。

ユーザーが直接変更できないテキストを表示するには、ラベルを使用します。

コードを書いて、実行時にイベントの応答内でラベルにより表示されたテキストを変更することも可能です。例えば、アプリケーションである変更を実行するのに数分かかるような時に、ラベルを使ってメッセージを表示することができます。また編集ボックスなどcaptionプロパティがないコントロールの識別にラベルを使用することもできます。

ラベルのプロパティ

ラベルのプロパティ

Alignment プロパティ

FocusControl プロパティ

Ellipses プロパティ

WrapStyle プロパティ

Alignment プロパティ

Alignmentプロパティを利用して、ラベルのテキストを右、左または中央に揃えます。

FocusControl プロパティ

FocusControlプロパティを使って、ユーザーがラベルのアクセス・キー（Altキー + ラベルのキャプション内の下線が引かれた文字）を押した時にフォーム上のどのコントロールにフォーカスを当てるかを指定します。

このプロパティを利用することで、キャプションのないコンポーネントにキーボードを使ってフォーカスを移動させることができるようになります。例えばリスト、ツリー・ビュー、コンボ・ボックスやグリッドにはキャプションがありません。

Ellipses プロパティ

Ellipses プロパティを使って、ラベルにテキストが入りきらなかった場合にどのように処理するかを制御します。このプロパティは、次の値を取ることができます：

None: テキストは切り捨てられます。

Word: ラベルに入りきらないテキストは切り捨てられ、省略記号が追加されます。

End: 文字列の一番最後がラベルに入りきらない場合、これが切り捨てられ、省略記号が追加されます。文字列の一番最後ではない単語がラベルの制限を超える場合は、省略記号なしに切り捨てられます。

Path: ラベルに入るように、文字列の中央部分の文字が省略記号に置き換えられます。文字列にバックスラッシュ(\)の文字が含まれる場合、最後のバックスラッシュの後に来る文字列はできる限り残されます。

このプロパティの使用方法を確認するには、次のソースをコピーしてフォームに貼り付け、フォームをコンパイルして実行してください。

```
FUNCTION options(*DIRECT)
BEGIN_COM role(*EXTENDS #PRIM_FORM) CAPTION('Label Ellipse Pr
DEFINE_COM class(#PRIM_LABL) name(#LABL_1) CAPTION('Text too l
DEFINE_COM class(#PRIM_LABL) name(#LABL_2) CAPTION('C:\Progra
DEFINE_COM class(#PRIM_GPBX) name(#GPBX_1) CAPTION('Ellipses')
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_END) CAPTION('End')
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_NONE) BUTTONCHECKED
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_PATH) CAPTION('Path
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_WORD) CAPTION('Wo
EVTROUTINE handling(#com_owner.Initialize)
SET com(#com_owner) CAPTION(*component_desc)
ENDROUTINE
SUBROUTINE name(DOELLIPSES)
IF cond('#RDBN_END.Buttonchecked = true')
SET com(#LABL_1) ELLIPSES(end)
SET com(#LABL_2) ELLIPSES(end)
ENDIF
IF cond('#RDBN_WORD.Buttonchecked = true')
SET com(#LABL_1) ELLIPSES(WORD)
SET com(#LABL_2) ELLIPSES(WORD)
ENDIF
```

```
IF cond('#RDBN_path.Buttonchecked = true')
SET com(#LABL_1) ELLIPSES(path)
SET com(#LABL_2) ELLIPSES(path)
ENDIF
IF cond('#RDBN_NONE.Buttonchecked = true')
SET com(#LABL_1) ELLIPSES(none)
SET com(#LABL_2) ELLIPSES(none)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_END.Click)
EXECUTE subroutine(DOELLIPSES)
ENDROUTINE
EVTROUTINE handling(#RDBN_NONE.Click)
EXECUTE subroutine(DOELLIPSES)
ENDROUTINE
EVTROUTINE handling(#RDBN_PATH.Click)
EXECUTE subroutine(DOELLIPSES)
ENDROUTINE
EVTROUTINE handling(#RDBN_WORD.Click)
EXECUTE subroutine(DOELLIPSES)
ENDROUTINE
END_COM
```

WrapStyle プロパティ

WrapStyle プロパティは単語をどのように折り返すかを指定します。

WrapStyle プロパティを使って単語の折り返しを制御します。

WrapStyle の Word ではテキストは単語の境界で折り返され、Continuous の場合は単語の境界に関わりなくテキストが折り返されます。

編集ボックス

編集ボックスは情報を入力・表示するために使用します。

編集ボックスを使って、情報を表示し、ユーザーが情報を入力できるようにします。

設計時にテキストを編集ボックスに入力することができ、実行時はユーザーにより、またはプログラムを使って入力します。

編集ボックスはフィールドの表示スタイルの省略値です。アプリケーション内で編集ボックスを使用する場合は、既存のフィールドを使用したり、新しいフィールドを作成する方がより適切でないかどうかを事前によく検討してください。

[編集ボックスのプロパティ](#)

[編集ボックスのメソッド](#)

編集ボックスのメソッド

SelectText メソッド

SelectText メソッド

SelectTextメソッドは、Length文字列のStartPositionから始まるテキストを選択します。

テキストが選択される前にコントロールにフォーカスがある時にのみこれが表示されることに注意してください。

このメソッドの構文は、以下のとおりです：

Invoke (#Meditbox.SelectText) StartPosition(10) Length(8)

または

Invoke (#Meditbox.SelectText) StartPosition(10)

[SelectText StartPosition](#)

[SelectText Length](#)

SelectText StartPosition

StartPositionは選択の最初の文字を指定します。

SelectText Length

Lengthは選択の文字数を指定します。Lengthが0もしくは指定されない場合、カーソルはStartPositionで指定された位置に置かれます。

編集ボックスのプロパティ

[DisplayAlignment](#) プロパティ

[EditAlignment](#) プロパティ

[CharacterCase](#) プロパティ

[HasSelection](#) プロパティ

[PasswordChar](#) プロパティ

[ReadOnly](#) プロパティ

DisplayAlignment プロパティ

DisplayAlignmentは、フォーカスがない時に編集ボックスの整列を設定します。

DisplayAlignmentプロパティを使用して、編集ボックスがフォーカスされていない時に編集ボックスのテキストをどのように整列するかを制御します。

このプロパティは、Right（右）、Left（左）またはCenter（中央）に設定できます。

EditAlignment プロパティ

EditAlignmentはフォーカスのある編集ボックスの整列を設定します。
EditAlignmentプロパティを使用して、編集ボックスにフォーカスがある時に編集ボックスのテキストをどのように整列するかを制御します。
このプロパティは、Right（右）またはLeft（左）に設定できます。

CharacterCase プロパティ

CharacterCaseは文字列のケース（大文字・小文字）を設定します。

CharacterCaseは文字列を小文字、大文字と小文字、または大文字で表示します。

HasSelection プロパティ

読み取り専用プロパティです。

HasSelectionはテキストが選択されているかどうかを示します。

PasswordChar プロパティ

PasswordCharはパスワードを非表示にします。

PasswordCharはパスワード・フィールドに使用された文字をプレースホルダーとして表示します。

このプロパティはパスワード・フィールドを作成する際に使用します。どんな文字でも使用することができるのですが、多くのWindowsベースのアプリケーションではアスタリスク(*)を使います。

このプロパティはTextプロパティには影響を与えません。Textにはユーザーがタイプしたその通りの文字やコードで設定されたものが入ります。入力されている時にパスワードを表示させるには、このプロパティを省略値のzero-length string("")に設定してください。

このプロパティにはどんな文字列でも割り当てることができますが、最初の文字に非常に重要な意味があり、残りは無視されます。

ReadOnly プロパティ

ReadOnlyはコンポーネントを入力用に使用可能かどうかどうかを制御します。

ReadOnlyを利用して、コンポーネントが入力可能かどうかどうかを制御します。このプロパティにTrueを設定すると、コンポーネントが読み取り専用になります。

スピン編集ボックス

スピン編集ボックスは値を調整するためのものです。

スピン編集を使用して、数字を加算・減算するスピン・ボタンがついた編集ボックスを表示します。またこのボタンを利用して、スクロールさせて様々な値を行ったり来たりさせることもできます。

[スピン編集ボックスのプロパティ](#)

[スピン編集ボックスのメソッド](#)

スピン編集ボックスのプロパティ

Increment プロパティ

MinimumValue プロパティ

MaximumValue プロパティ

Wrap プロパティ

Increment プロパティ

Incrementにはスピン・ボタンがどのようにして値を調整するかを指定します。

Increment プロパティにはスピン・ボタンにより影響を受ける変更量を指定します。例えば、incrementが2の時、編集ボックスの数字はスピン・ボタンが押される度に2ずつ加算または減算されます。

MinimumValue プロパティ

MinimumValueはスピン編集ボックスに含むことのできる最小値を指定します。

MaximumValue プロパティ

MaximumValueはスピン編集ボックスに含むことのできる最大値を指定します。

Wrap プロパティ

Wrapプロパティはスピン編集の範囲の最後の値に届いた時にどうなるかを指定します。

スピン編集ボックスのメソッド

Dec [メソッド](#)

Inc [メソッド](#)

Dec メソッド

Decメソッドを使用してスピン編集ボックスの値を減算します。値は1ずつ減算されます。

Inc メソッド

Incメソッドを使用してスピン編集ボックスの値を加算します。値は1ずつ加算されます。

複数行編集ボックス

複数行編集ボックスには1行以上のテキストが含まれます。

複数行編集ボックスを使うと、ユーザーが長いテキストを入力したり、編集したりできるようになります。

複数行編集ボックスは簡素な編集ボックスと似ているようにユーザーには見えますが、リストとして機能しています。複数行編集ボックスの各行はリストのエントリーに相当します。ですから、複数行編集ボックスを埋めるには、SELECTを使い、行を追加するにはADD_ENTRYを使います。同様にファイルに編集ボックスの内容を書くにはSELECTLISTとINSERTのコマンドを使用します。

SET例200も参照してください：[複数行編集ボックスの利用](#)

[複数行編集ボックスのプロパティ](#)

[複数行編集ボックスのメソッド](#)

複数行編集ボックスのプロパティ

AddEntryMode プロパティ	EnablePaste プロパティ	MaximumLines プロパティ
CharacterPosition プロパティ	HasSelection プロパティ	OnOutput プロパティ
ComponentVersion プロパティ	HorizontalScroll プロパティ	OutputText property プロパティ
CurrentLine プロパティ	LineCount プロパティ	ShowLineNumbers プロパティ
DisableNoScroll プロパティ	MarginLeft プロパティ	VerticalScroll プロパティ
EnableCut プロパティ	MaximumLineLength プロパティ	WordWrap プロパティ

AddEntryMode プロパティ

AddEntryModeは編集ボックスにどのようにデータが追加されるかを決定します。

AddEntryModeを使用して、複数行編集ボックスでデータをどのように処理するかを制御します。

このプロパティにOnePerLineを設定して各行ごとに1エントリーを処理することもできますし、MultiplePerLineで各行複数のエントリーを処理することもできます。

例えばリスト・タイプの情報を表示したい場合はOnePerLineの値を使います。この値により、複数行編集ボックスのMaximumLineLengthプロパティに、ColumnRoleがDataに設定されている列のフィールドの長さが設定されます。

データベースの内容をひと続きのテキストとして表示したい時は、MultiplePerLineの値を使います。データがファイルから編集ボックスに取り込まれる場合は、エントリーはまとめて行に埋め込まれます。反対に、SELECTLISTコマンドを使ってテキストが編集ボックスから抽出される時は、ColumnRoleがDataに設定されている列のフィールドに相当する長さの文字列まで、このテキストが分割されます。データを入力されたフォーマットで保つ場合、編集ボックスにColumnRoleがLineNumberの列とColumnRoleがLineContinuationの列がなければなりません。

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 10.0 以前のバージョンです。

1 はLANSA 10.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

ComponentVersionが1の場合、複数行編集コントロールはデータ列として定義されたフィールドの属性を確認し、フィールドに*LC属性がないと、全てのテキストが大文字になります。

CurrentLine プロパティ

実行時の読み取り専用プロパティです。CurrentLineプロパティを使って、カーソルが位置する行を決定します。

CharacterPosition プロパティ

実行時のプロパティです。CharacterPositionプロパティを使って、行の上のカーソル位置を決定するまたは設定します。

複数行編集ボックスをクリアする時、CharacterPositionを0に設定してカーソルがボックスの左上に位置するようにします。

EnableCut プロパティ

EnableCutプロパティをTrueに設定すると、ユーザーバテキストを切り取ることができます。

DisableNoScroll プロパティ

DisableNoScroll プロパティを利用して、必要でない時にスクロールバーを非表示にするかどうかを制御します。このプロパティがFalseに設定されている時、スクロールバーはコンテンツがコントロールに入りきらない時にのみ表示されます。

EnablePaste プロパティ

EnablePaste プロパティをTrueに設定すると、ユーザーはテキストを編集ボックスに貼り付けることができます。

HasSelection プロパティ

読み取り専用プロパティです。

HasSelectionはテキストが選択されているかどうかを示します。

HorizontalScroll プロパティ

HorizontalScrollは水平スクロールバーを表示したり、非表示にしたりします。

このプロパティにTrueを設定すると、水平スクロールバーが表示されます。

スクロールバーはコンテンツがコントロールより幅が広い時に有効になります。使い易さの面から、水平スクロールバーは避けるようにしてください。

複数行編集ボックスでは編集ボックスの幅を行幅に合わせるようにしてください。またはWordWrapプロパティをTrueに設定します。

LineCount プロパティ

実行時の読み取り専用プロパティです。LineCountプロパティを使うと、複数行編集ボックスにある行数が分ります。

MarginLeft プロパティ

MarginLeftプロパティを使って編集ボックスの左端とテキストの間のマージンを設定します。マージンはピクセルで指定します。

MaximumLineLength プロパティ

MaximumLineLengthプロパティを使って、1行に入力できる文字数を制限します。

このプロパティを設定する時は、データが入れられるフィールドの長さを考慮に入れてください。また水平スクロールバーを避けるためにも、編集ボックスの幅も考慮にいれる必要があります。

AddEntryモードがOnePerLineの場合、このプロパティは自動的にColumnRoleがDataに設定されている列のフィールドの幅に設定されません。

ユーザーはMaximumLineLengthで許可された長さ以上の文字はタイプできません。ただし編集ボックスが自動的に埋められる時は、制限を超えることができます。

MaximumLines プロパティ

MaximumLinesプロパティを使うと、複数行編集ボックス・コントロールの最大行数が分ります。

OnOutput プロパティ

OnOutputはテキストの出力フォーマットを決定します。

OnOutputプロパティを使用してデータがどのようなフォーマットで保存されるかを制御します。このプロパティは、次の値を取ることができます：

None: テキストは変更されません。

AddReturn: キャリッジ・リターンが各行の行末に追加されます。

AddLineFeed: ライン・フィードが各行の行末に追加されます。

AddNewLine: キャリッジ・リターンとライン・フィードが各行の行末に追加されます。

AddNull: Nullが各行の行末に追加されます。

AddSpace: スペースが各行の行末に追加されます。

OutputText property プロパティ

OutputTextは複数行編集ボックス内にあるすべてのテキストです。
読み取り専用プロパティです。 OutputText プロパティを利用して編集
ボックスのコンテンツを取り出します。

ShowLineNumbers プロパティ

ShowLineNumbersは行番号の表示を制御します。

ShowLineNumbersプロパティを利用して、自動の行ナンバリングを表示するかどうかを制御します。

VerticalScroll プロパティ

VerticalScrollは垂直スクロールバーを表示したり、非表示にしたりします。

このプロパティにTrueを設定すると、垂直スクロールバーが表示されます。

垂直スクロールバーはコンテンツがコントロールの高さに入りきらなかった時に表示されます。

WordWrap プロパティ

WordWrap は折り返しを制御します。

WordWrap プロパティにTrueを設定すると、単語の折り返しができます。通常複数行編集ボックスでは、MaximumLineLengthが編集ボックスの幅より長い時に折り返しをし、すべてのテキストが表示されるようにします。そして水平スクロールバーを使用する必要がないようにします。ラベルを複数行で表示したい時はラベルで折り返しを設定します。

複数行編集ボックスのイベント

Positionchanged イベント

LineChanged イベント

Positionchanged イベント

Positionchangedはカーソルが移動すると起動します。

Positionchangedイベントはカーソルが行の上の別の文字に移動した時に起動します。

LineChanged イベント

LineChangedはカーソルが別の行に移動すると起動します。

LineChangedイベントはカーソルが別の行に移動されると起動します。

複数行編集ボックスのメソッド

Cut メソッド

Copy メソッド

Paste メソッド

Undo メソッド

Redo メソッド

SelectAll メソッド

SelectMultiLineText メソッド

複数行テキストの選択

Find メソッド

Replace メソッド

Print メソッド

Cut メソッド

Cutはテキストを削除し、これをクリップボードに置きます。

Cutメソッドは選択されたテキストを削除し、これをクリップボードに置きます。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.Cut
```

Copy メソッド

Copyはテキストをクリップボードにコピーします。

Copyメソッドは選択されたテキストをクリップボードにコピーします。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.Copy
```

Paste メソッド

Pasteはコンテンツをクリップボードに貼り付けます。

Pasteメソッドはクリップボードの内容を複数行編集ボックスに貼り付けます。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.Paste
```

Undo メソッド

Undoメソッドは最後のアクションを元に戻します。

Undoメソッドは複数行編集ボックス内の最後のアクションを無効にします。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.Undo
```

Redo メソッド

Redoは元に戻されたアクションを再度適用します。

Redoメソッドは、複数行編集ボックス内でUndoメソッドを使って元に戻されたアクションを再度実行します。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.Redo
```

SelectAll メソッド

SelectAllは全てのテキストを選択します。

SelectAllメソッドは複数行編集ボックス内の全てのテキストを選択します。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.SelectAll
```

SelectMultiLineText メソッド

SelectMultiLineTextは複数行のテキストを選択します。

SelectMultiLineTextメソッドは、StartLineNumber、StartPositionから始まり、EndLineNumber、EndPositionまでのテキストを選択します。

テキストが選択される前にコントロールにフォーカスがある時にのみこれが表示されることに注意してください。

このメソッドの構文は、以下のとおりです：

```
Invoke (#Meditbox.SelectMultiLineText) StartLineNumber(2) StartPosition(10)
```

複数行テキストの選択

StartLineNumber パラメータ

StartPosition パラメータ

EndLineNumber パラメータ

EndPosition パラメータ

StartLineNumber パラメータ

StartLineNumberは選択の最初の行を指定します。

StartPosition パラメータ

StartPositionは選択の最初の位置を指定します。

EndLineNumber パラメータ

EndLineNumberは選択の最後の行です。

EndLineNumberを使って選択の最後の行を指定します。

EndPosition パラメータ

EndPositionは選択の最後の文字の位置です。

EndPositionを使って選択の位置を指定します。

Find メソッド

Findは文字列を検索します。

Findメソッドは複数行編集ボックス内の文字列を検索するために使用するダイアログ・ボックスを出します。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.Find
```

Replace メソッド

Replaceは文字列を検索し置き換えます。

Replaceメソッドは複数行編集ボックス内の文字列を検索するために使用するダイアログ・ボックスを出し、これを別のものに置き換えます。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.Replace
```

Print メソッド

Printメソッドはテキストをプリンターに送ります。

Printメソッドは複数行編集ボックスのコンテンツをプリンターに送信します。

このメソッドの構文は、以下のとおりです：

```
Invoke #Meditbox.Print
```

フォーム

フォームはウィンドウ、ダイアログやメッセージ・ボックスに代表される、表示エリアのことです。フォームを使ってアプリケーションのウィンドウを作成します。フォームのプロパティでは、位置、サイズや色などの外見と、サイズ変更可能・不可能などの動作を決定します。

フォームはユーザーにより開始されたイベントやシステムにより起動されたイベントに応答することができます。例えば、ユーザーがクリックすることでフォームの色を変更することができるようなClickイベント・ルーチンのコードをフォームに書くことができます。

[フォームのプロパティ](#)

[フォームのイベント](#)

[フォームのメソッド](#)

フォームのプロパティ

BorderIcons プロパティ	FormStyle プロパティ	NormalTop プロパティ
ClientHeight プロパティ	FrameStyle プロパティ	NormalWidth プロパティ
ClientWidth プロパティ	Icon プロパティ	ScalingFactorHor プロパティ
ComponentVersion プロパティ	Menubar プロパティ	ScalingFactorVer プロパティ
FormOwner プロパティ	NormalHeight プロパティ	TitleBarVisible プロパティ
FormPosition プロパティ	NormalLeft プロパティ	WindowState プロパティ

BorderIcons プロパティ

BorderIconsはフォームの右上の角に表示されるアイコンを指定します。このプロパティは、次のどのアイコンがフォームの右上の角に表示されるかを指定します。

Help (ヘルプ)

Maximize (最大化)

Minimize (最小化)

None (なし)

System Menu (システム・メニュー)

BorderIconsへの変更は設計時ではなく、実行時のみ表示されることに注意してください。MinimizeまたはMaximizeのアイコンが表示されている時はヘルプのアイコンは指定できません。

実行時にフォームが最大化される場合は、最大化ボタンが自動的に元のサイズに戻すボタンに変更されます。

ClientHeight プロパティ

ClientHeight プロパティはフォームのクライアント・エリアの高さを定義します。

フォームのクライアント・エリアは、フォームのサイズ変更枠、タイトル・バーおよびメニュー・バーエリアを除いた部分です。

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 10.0 以前のバージョンです。

1 はLANSA 10.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

ComponentVersionが1の時、FormStyleがStayOnTopとStayOnTopChildのフォームが一番上になります。また、フォームの設計時に使用されたデスクトップと異なる縮尺単位 (pixel per inch) でフォームがデスクトップで実行された場合、そのフォームおよびコンテンツは自動的に実行しているデスクトップの縮尺単位に合わせられます。

ClientWidth プロパティ

ClientWidthはフォームのクライアント・エリアの幅を指定します。

The ClientWidthプロパティはフォームのクライアント・エリアの幅を定義します。

フォームのクライアント・エリアは、フォームのサイズ変更枠、タイトル・バーおよびメニュー・バーエリアを除いた部分です。

NormalTopプロパティ

NormalTopは元のサイズに戻したフォームのTopプロパティについての情報を返します。

NormalTopプロパティは、フォームが最大化、最小化されている時でも、元のサイズに戻したフォームのTopプロパティの情報を返します。

NormalLeft プロパティ

NormalLeftは元のサイズに戻したフォームのLeftプロパティについての情報を返します。

NormalLeftプロパティは、フォームが最大化、最小化されている時でも、元のサイズに戻したフォームのLeftプロパティの情報を返します。

NormalHeight プロパティ

NormalHeightは元のサイズに戻したフォームのHeightプロパティについての情報を返します。

NormalHeightプロパティは、フォームが最大化、最小化されている時でも、元のサイズに戻したフォームのHeightプロパティの情報を返します。

NormalWidth プロパティ

NormalWidthは元のサイズに戻したフォームのWidthプロパティについての情報を返します。

NormalWidthプロパティは、フォームが最大化、最小化されている時でも、元のサイズに戻したフォームのWidthプロパティの情報を返します。

FormOwner プロパティ

FormOwnerはフォームのオーナーを指定します。

実行時のプロパティです。

FormOwnerプロパティを使って、マルチフォーム・アプリケーションで所有されているフォームのオーナーを指定します。

メンバー・フォームを表示する前に、FORMOWNERプロパティを使用してオーナー・フォームを設定してください。これはアプリケーションの基本構造に関係なく、デフォルトでアプリケーションで実行する最初のフォームがアプリケーションのすべてのメンバー・フォームのオーナーになるからです。複雑なアプリケーションでは、オーナー・フォームを設定しないと、予期せぬ結果が発生する場合があります。

次のコードでは表示前にメンバーのオーナーを設定します。

```
EVTROUTINE HANDLING(#MDetails.Click)
SET COM(#frmdetail) formowner(#com_owner)
invoke #frmdetail.ShowForm
ENDROUTINE
```

FormPosition プロパティ

FormPositionはフォームの位置を指定します。

このプロパティは、次の値を取ることができます：

Designed: 設計時のフォームの位置・サイズを使用します。

Default: フォームの省略値の位置・サイズを使用します。

DefaultSizeOnly: 省略値のサイズを使用します。位置は設計時に設定されます。

DefaultPositionOnly: 省略値の位置を使用します。サイズは設計時に設定されます。

ScreenCenter: 画面の中央にフォームを配置します。

フォームのデフォルトのサイズと位置は#STD_FORMコンポーネントで指定でき、これは変更可能です。

FrameStyle プロパティ

FrameStyleはフォームのフレームのスタイルを設定します。

このプロパティには次のいずれかの値を設定できます：

Dialog: サイズ変更はできません。タイトルバーがあります。ヘルプ・最大化・最小化ボタンを含むことができます。システム・メニューはありません。

None: フォームにフレームもタイトルバーもありません。

Sizable: 枠線のアイコンもしくはマウスを使用してサイズ変更できません。

Single: コントロール・メニュー・ボックス、タイトル・バー、最大化及び最小化ボタンを表示できます。最大化、最小化ボタンを使用してのみサイズ変更が可能です。

SizeToolWindow: 閉じるボタンを表示し、縮小されたフォント・サイズでタイトルバ・テキストを表示します。Windowsタスクバーにフォームは表示されません。サイズ変更可能です。

ToolWindow: 閉じるボタンを表示し、縮小されたフォント・サイズでタイトルバ・テキストを表示します。Windowsタスクバーにフォームは表示されません。

FormStyle プロパティ

FormStyleはフォームのスタイルを指定します。

このプロパティで実行時のフォームの動作を指定します。

Normal フォームは標準フォームです。

Owned このフォームが他のフォームのメンバー・フォームの時、オーナー・フォームが閉じられると閉じられます。

StayOnTop ウィンドウがフォーム・オーナーの上に表示されます。これは通常アプリケーションによって表示される最初のフォームです。

フォームのスタイルをNormalChild、OwnedChild、StayOnTopChildに設定すると、親のフォームが非表示や最小化されると、自動的にこのフォームが非表示、最小化され、親が閉じられると、自動的に閉じられます。OwnedChildフォームはオーナー・フォームの上に残り、タスクバー上には表示されません。

NormalChildとOwnedChildフォームの違いを見るには、次のソースをコピーして、FORMOWNEXという名前のフォームに貼り付け、コンパイルして実行してください。

```
FUNCTION options(*DIRECT)
BEGIN_COM role(*EXTENDS #PRIM_FORM) HEIGHT(150) LEFT(296) T
DEFINE_COM class(#PRIM_PHBN) name(#PHBN_1) CAPTION('Show No
DEFINE_COM class(#PRIM_PHBN) name(#PHBN_2) CAPTION('Show Ow
DEFINE_COM class(#PRIM_CKBX) name(#CKBX_1) CAPTION('Fail Clos
DEFINE_COM class(#FORMOWNEX) name(#NORMALFORM) FORMSTY
DEFINE_COM class(#FORMOWNEX) name(#OWNEDFORM) FORMSTY
EVTROUTINE handling(#PHBN_1.Click)
IF_REF com(#Com_Owner.FormOwner) is(*Null)
CHANGE field(#STD_NUM) to('#Com_Owner.Left + #Com_Owner.Width +
ELSE
CHANGE field(#STD_NUM) to('#Com_Owner.Left + 10')
ENDIF
SET com(#NormalForm) FORMOWNER(#Com_Owner) LEFT(#Std_Num)
INVOKE method(#NormalForm.ShowForm)
ENDROUTINE
EVTROUTINE handling(#PHBN_2.Click)
```

```
IF_REF com(#Com_Owner.FormOwner) is(*Null)
CHANGE field(#STD_NUM) to('#Com_Owner.Top + #Com_Owner.Height +
SET com(#OwnedForm) TOP(#Std_Num)
CHANGE field(#STD_NUM) to('#Com_Owner.Left + #Com_Owner.Width +
ELSE
CHANGE field(#STD_NUM) to('#Com_Owner.Left + 10')
ENDIF
SET com(#OwnedForm) FORMOWNER(#Com_Owner) LEFT(#Std_Num)
INVOKE method(#OwnedForm.ShowForm)
ENDROUTINE
EVTROUTINE handling(#COM_OWNER.CloseQuery) options(*NOCLEAR
IF cond('#CKBX_1.ButtonState = Checked')
SET com(#Option) VALUE(False)
ENDIF
ENDROUTINE
END_COM
```

Icon プロパティ

Iconはフォームのアイコンを設定します。

このプロパティを使って、タスクバーやフォームの左上の角に小さなアイコンとして表示されるリポジトリに登録されたアイコンを指定します。

このアイコンが小さなアイコン(16 x 16 ピクセル)として識別可能なことを確認してください。アイコンを指定しない場合は、フォームのデフォルトのアイコンが使用されます。

Menubar プロパティ

Menubarはフォームのメニューバーを設定します。

Menubarプロパティを使って、フォームにどのメニューバーを使用するかを指定します。このプロパティの省略値は、フォーム内で最初に作成されたメニューバーです。

普通は1フォームにつき1つのメニューバーを作成します。

ScalingFactorHor プロパティ

ScalingFactorHorは水平スケール・ファクタを取り込みます。

ScalingFactorHorプロパティは、フォームが設計されるデスクトップのスケール・ファクタを取りこみます。

値はデスクトップの水平ピクセル/インチから取られます。このプロパティは実行時にVisual LANSАにより使用され、デスクトップ上で実行されているフォームのスケール・ファクタがフォームの設計に使用されたデスクトップの値と一致しない場合に、自動的にフォームやコンテンツの拡大縮小を調節します。

ScalingFactorVer プロパティ

ScalingFactorVerは垂直スケール・ファクタを取り込みます。

ScalingFactorHVerプロパティは、フォームが設計されるデスクトップのスケール・ファクタを取りこみます。

値はデスクトップの垂直ピクセル/インチから取られます。このプロパティは実行時にVisual LANSAにより使用され、デスクトップ上で実行されているフォームのスケール・ファクタがフォームの設計に使用されたデスクトップの値と一致しない場合に、自動的にフォームやコンテンツの拡大縮小を調節します。

TitleBarVisible プロパティ

TitleBarVisibleはタイトルバーを表示・非表示します。

このプロパティにFalseを設定すると、タイトルバーが表示されません。

WindowState プロパティ

WindowStateはウィンドウがどのように表示されるかを制御します。このプロパティを使って実行された時のフォームのビジュアル状態を制御します。 WindowStateがNormalの場合、サイズはheightとwidthプロパティに設定されます。

フォームが最初に表示される時にWindowStateがMinimizedに設定されている場合、フォームがデスクトップ上に表示されない状態を避けるため、Normalとして表示されることに注意してください。

フォームのイベント

Closing イベント

CloseQuery イベント

Deactivate イベント

Closing イベント

Closingイベントはユーザーがフォームを閉じた時に起動します。

画面からフォームが取り除かれようとする時にClosingイベントが起動されます。ユーザーはフォームの左上にあるシステム・メニュー、または右上にある閉じるのボーダー・アイコン、またはアプリケーションによってはメニュー・オプションやボタンを使ってフォームを閉じることができます。

[Action パラメータ](#)

Action パラメータ

Actionパラメータはフォームが閉じられた後取るアクションを指定します。

Actionパラメータを使って、Closingイベントが起動された後にどのようなアクションを取るかを指定します。このパラメータにはNone、Hide、MinimizeおよびFreeの値を取ることができます。

省略値はFreeです。ウィンドウを閉じ、使われたリソースを解放します。

Hideはフォームを非表示にしますが、すぐに再表示できるように、背景でアクティブな状態を保ちます。

Noneはフォームが閉じられないようにします。

Minimizeはフォームを最小化します。

フォームがモーダルで表示されている時、NoneとFreeの両方でウィンドウが閉じられ、使用されたリソースが解放されることに注意してください。この時にフォームが閉じられないようにするにはCloseQueryイベントを使用します。

CloseQuery イベント

CloseQueryはフォームやアプリケーションが閉じられる前に起動されます。

CloseQueryイベントはフォームが閉じる前に起動します。このイベントにはContinueというパラメータがあり、TrueまたはFalseに設定できます。これを使って、Closeイベントを続行するかどうかユーザーに確認します。

このイベントの使用例については、『Visual LANSA 開発者ガイド』のフォームの章を参照してください。

[Continue パラメータ](#)

Continue パラメータ

Continueパラメータはフォームを閉じる時の確認に使用します。

CloseQueryEventのContinueパラメータを使って、フォームを閉じるかどうかの確認を行います。このパラメータには、TrueまたはFalseが設定できます。

次のイベント・ルーチンでは、#Dialogという名前のフォームが閉じられようとする時に別のフォームを表示します。#Dialogから返されたModalResultの値がNoの場合、フィールド#OptionにFalseが設定され、この値がContinueパラメータの値として使用されるので、フォームは閉じられません。ModalResultの値がNoでなければ、フォームは閉じられます。

```
EvtRoutine Handling(#Com_Owner.CloseQuery) Continue(#Option)
  Invoke #Dialog.ShowModalForm
  Define #QResult Reffld(#Std_Texts)
  Change #QResult #Dialog.modalResult
  Case #QResult
  When '= No'
  Invoke #Com_Owner.RestoreForm
  Set #Option Value(FALSE)
  Otherwise
  Set #Option Value(TRUE)
  EndCase
EndRoutine
```

Deactivate イベント

フォームが無効になるとDeactivateイベントが起動されます。

フォームが無効化されるとフォームのタイトルバーの色が変わります。

フォームのメソッド

CloseForm メソッド

CloseFormQuery メソッド

ActivateForm メソッド

HideForm メソッド

MaximizeForm メソッド

MinimizeForm メソッド

ShowForm メソッド

ShowModalForm メソッド

RestoreForm メソッド

CloseForm メソッド

CloseFormはフォームを閉じます。

CloseFormメソッドを使用して、フォームを閉じます。

CloseFormメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.CloseForm
```

CloseFormメソッドを使用している時、フォームは表示されませんが、ロジック、変数、コンポーネントと値のみがアクティブな状態に保たれます。CloseFormメソッドを使って閉じられたフォームを表示する時、CreateInstanceとInitializeイベントは起動しません。

CloseFormとHideFormは非常に良く似ていますが、CloseFormの方が内部のリソースをより効果的に利用します。

CloseFormQuery メソッド

CloseFormQueryは、確認後にフォームを閉じます。

CloseFormQueryメソッドは、CloseQueryイベントを起動してユーザーが本当にフォームを閉じたいのかどうかを確認した後にのみ使用され、フォームを閉じる要求をします。

CloseFormQueryメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.CloseFormQuery
```

ActivateForm メソッド

ActivateFormメソッドを使って、フォームをアクティブにします。フォームがアクティブになると、デスクトップの手前になり、そのフォームに現在のフォーカスが当たります。

ActivateFormメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.ActivateForm
```

このアクティブなフォームはキーボード・フォーカスを受け取り、タイトルバーとボーダーが強調表示されます。このフォームが最小化されていると、アクティブにされた時に元のサイズに戻されないことに注意してください。アクティブ化された時にフォームが元のサイズになっているようにするために、ウィンドウのWindowStateの値がMinimizedかどうかを確認し、アクティブ化する前にサイズを元に戻します：

```
If cond'(#MForm.WindowState *eq Minimized)'  
  Invoke #MForm.RestoreForm  
  Invoke #MForm.ActivateForm  
Endif
```

フォームを表示するためには、ShowFormメソッドを使用することに注意してください。このActivateFormメソッドは、全ての開いているウィンドウの前にフォームを持ってくるよう設計されています。

HideForm メソッド

HideFormメソッドを使って、フォームを視界から見えないように隠します。

フォームが隠されている時、フォームは画面から取り除かれ、VisibleプロパティはFalseに設定されます。 HideFormメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.HideForm
```

MaximizeForm メソッド

MaximizeFormメソッドを使用して、フォームのウィンドウを全画面サイズに拡大します。MaximizeFormメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.MaximizeForm
```

MinimizeForm メソッド

MinimizeFormメソッドを使用して、フォームのウィンドウをアイコンに最小化します。 MinimizeFormメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.MinimizeForm
```

ShowForm メソッド

ShowFormメソッドを使って、フォームを表示します。

フォームを初めて表示する場合は、このメソッドによりフォームも開かれます。

ShowFormメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.ShowForm
```

フォームのWindowStateがMinimizedの場合、ShowFormによりフォームのサイズは元に戻されませんので注意してください。フォームを元のサイズに戻すには、RestoreFormメソッドを使用します。

フォームが既に関いている場合、ShowFormメソッドによりフォームはアクティブ化されますが、他の開いているウィンドウの前には持って来られません。これを行うには、ActivateFormメソッドを使用します。

ShowModalForm メソッド

ShowModalFormメソッドはフォームを表示し、モーダルにします。

ShowModalFormメソッドを使って、フォームが表示され、モーダルな状態になります。モーダル・フォームでは、フォーカスが別のフォームに移動する前にユーザーが何かしらのアクションを実行する必要があります。フォームを初めて表示する場合は、このメソッドによりフォームも開かれます。

ShowModalFormメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.ShowModalForm
```

プッシュ・ボタンやフォームの[ModalResult プロパティ](#)も参照してください。

RestoreForm メソッド

RestoreFormメソッドを使って、フォームが最小化または最大化される前のサイズに戻します。 RestoreFormメソッドの構文は、以下のとおりです：

```
INVOKE #FormB.RestoreForm
```

パネル

別のコンポーネントのコンテナです。

パネルはボーダーやキャプションのない単純なコンテナで、ボタン、チェック・ボックス、テキスト・フィールドなどの他のコンポーネントをグループ化するのに使用します。パネルは別のパネルの中にネストすることも可能です。

[パネルのプロパティ](#)

パネルのプロパティ

HorizontalScrollPos プロパティ

VerticalScrollPos プロパティ

HorizontalScrollInc プロパティ

VerticalScrollInc プロパティ

HorizontalScrollPos プロパティ

HorizontalScrollPosは水平スクロールバーの位置を設定します。

このプロパティは、HorizontalScrollプロパティがTrueに設定され、水平スクロールバーが表示されるように設定されている時に利用できます。

HorizontalScrollPosを使って、パネル内のコンポーネントがパネルのサイズを超えた時にパネルのスクロール位置を制御します。このプロパティを使用して、パネルの境界より外にあるコンポーネントをプログラムでスクロールして表示することもできます。

VerticalScrollPos プロパティ

VerticalScrollPosは垂直スクロールバーの位置を設定します。

このプロパティは、VerticalScrollプロパティがTrueに設定され、垂直スクロールバーを表示するよう設定されている時に利用できます。

VerticalScrollPosを使って、パネル内のコンポーネントがパネルのサイズを超えた時にパネルのスクロール位置を制御します。このプロパティを使用して、パネルの境界より外にあるコンポーネントをプログラムでスクロールして表示することもできます。

HorizontalScrollInc プロパティ

HorizontalScrollIncは水平スクロールバーのピクセル単位の増分を指定します。

HorizontalScrollIncプロパティを使って、水平スクロールバーが表示されている時の増分をピクセルで指定します。

VerticalScrollInc プロパティ

VerticalScrollIncは垂直スクロールバーのピクセル単位の増分を指定します。

VerticalScrollIncプロパティを使って、垂直スクロールバーが表示されている時の増分をピクセルで指定します。

グラフ

グラフ内のデータです。

グラフを使って1つまたは複数のフィールド内のデータを視覚的に表示します。

グラフはVisual LANSAのリストと同じように処理します：まず表示するデータを定義します。これはグラフ内にフィールドをドラッグするだけでできます。そして、そのデータを取り出すためにSELECTやADD_ENTRYのステートメントを使用します。

GraphTypeプロパティを使って、描くグラフのタイプ：棒、折れ線、または円グラフ、を定義します。グラフ・タイプには関連付けられた様々なスタイルがあり、グラフの外観を操作できます。

次のコードは、LANSAデモンストレーション・システム(PSLMST)のフィールドSALARYからのデータを使って円グラフを作成します。

```
DEFINE_COM CLASS(#PRIM_GRPH) NAME(#GRPH_1) GRAPHTYPE(P  
DEFINE_COM CLASS(#PRIM_GRCL) NAME(#GRCL_1) DISPLAYPOSIT  
EVTROUTINE HANDLING(#GRPH_1.INITIALIZE)  
SELECT FIELDS(#GRPH_1) FROM_FILE(PSLMST) WHERE(#SALARY <  
ADD_ENTRY TO_LIST(#GRPH_1)  
ENDSELECT  
ENDROUTINE
```

[グラフのプロパティ](#)

[グラフのメソッド](#)

グラフのプロパティ

AreaStyle プロパティ

BarStyle プロパティ

CurveGranularity プロパティ

CurveOrder プロパティ

CurveStyle プロパティ

GraphType プロパティ

LineStyle プロパティ

PieStyle プロパティ

ScatterStyle プロパティ

StatisticalStyle プロパティ

SurfaceStyle プロパティ

ValueAt プロパティ

XCaption プロパティ

YCaption プロパティ

GraphType プロパティ

GraphTypeはグラフのタイプを設定します。

データを異なったタイプのグラフで表示するには、このプロパティを変更します。使用可能なグラフ・タイプは以下のとおりです：

Area Pie

Bar Scatter

Line Surface

Area（面）グラフは、プロット値の合計とともに全体に対する一部分の関係も示します。データ・ポイントは線で結ばれ、線と軸の間のエリアは埋められます。このエリアはこれを構成する項目により更に小さなエリアに分けられ、その項目の割合を示します。例えば、面グラフは売上の推移を示すとともに、異なる部署が全体に対しどの程度貢献しているかの比率を示す場合などに使用できます。

Bar（棒）グラフは、データが棒により表示されます。これはそれぞれの項目間の比較を示す場合に使用されます。棒グラフは1つのラベル列および複数のデータ列を表示できます。

Line（折れ線）グラフはデータ・ポイントを線で結びます。機能的には棒グラフと似ています。

Pie（円）グラフは1つの欄からのデータを1切れのパイのように示します。このパイの1切れの大きさは、この項目の全体に対する比率を示します。1つのデータ・フィールドを表示できるのは、このグラフ・タイプだけです。円グラフは、例えば全体の売上に対する各部署が占める割合をパーセントで示す際などに使用されます。

Surface（等高線）グラフは2つのデータ欄を結合させて3Dで表示します。このグラフは2セットのデータの最適な結合を見つけ出すのに使われます。

Scatter（散布図）はデータ欄のペア間の関係を示します。

DisplayPositionが奇数の欄はX軸に、DisplayPositionが偶数の欄はY軸に、データの群れとして表示されます。散布図は通常、科学的なデータに使用されます。例えば予想値と実測値を比較したりします。

これらの異なるグラフ・タイプにはそれぞれ関連する特定の設定があることに注意してください。例えば棒グラフの外見を変更する場合は、BarStyleプロパティを利用します。

XCaption プロパティ

XCaptionはグラフの下側のタイトルを設定します。

XCaptionプロパティを使って、グラフの下側のタイトルを設定します。

YCaption プロパティ

YCaptionはグラフの左側の縦のタイトルを設定します。

YCaptionプロパティを使って、グラフの左側のタイトルを設定します。

CurveStyle プロパティ

CurveStyleは曲線のタイプを制御します。

このプロパティは表示される曲線のタイプ（可変次数(variable-order)、多項式(polynomial)、対数(logarithmic)、指数(exponential)など）を制御します。

CurveOrder プロパティ

variable-order polynomial (可変次数多項式) 曲線の場合、この値が1だと直線が引かれます(線形近似と同じ)。値がポイントの数より1小さい場合、全てのポイントを通る曲線が描かれます。

移動平均の中心や終点の曲線では、最初のポイントから始まり、移動平均により均等にされるデータ・ポイントの範囲がこの値になります。

その他の全ての曲線タイプでは、このプロパティは無視されます。

CurveGranularity プロパティ

CurveGranularityは曲線を作成するステップ数を制御します。

平均移動以外の全ての曲線タイプでは、CurveGranularity プロパティは曲線を作り上げるステップ数、つまり直線の数に制御します。数が大きければ大きいほど、曲線はなだらかになります。省略値は50です。

StatisticalStyle プロパティ

StatisticalStyleは曲線のタイプを示します。

StatisticalStyle プロパティは、描かれる統計線または曲線のタイプを指定します：mean（平均）、min/max（最小/最大）、standard deviation（標準偏差）

AreaStyle プロパティ

AreaStyleは面グラフのスタイルを設定します。

AreaStyleのプロパティ・リストを使用して、面グラフのスタイルを設定します。AreaStyleのドロップダウン・リストで次の中から複数の設定を選択できます：

NoLabels: グラフのX軸とY軸両方のラベルを非表示にします。

XGrid: グラフの水平グリッド線を表示します。

YGrid: グラフの垂直グリッド線を表示します。

SymbolColor: 凡例のテキストを記号と同じ色にします。

BarStyle プロパティ

BarStyleは棒グラフのスタイルを設定します。

BarStyleのプロパティ・リストを使用して、棒グラフのスタイルを設定します。BarStyleのドロップダウン・リストで次の中から複数の設定を選択できます：

3D: グラフが立体に見えるようにします。

Horizontal: 棒グラフの棒を横に表示します。

NoLabels: グラフのX軸とY軸両方のラベルを非表示にします。

Stacked: 積み重ねグラフを描きます。積み重ねグラフは全体に対する各個別の項目の関係を示します。例えば、#SALES96と#SALES97という2つのフィールドを使ってグラフを描くとすると、ファイルは#SALESPERSONがキーとなり、1人のセールス・パーソンにつき1つの棒が積み重ねグラフに表示されます。この棒は2つの部分から成っており、1つは#SALES96、もう1つは#SALES97の売上が積み重なって表示されます。

StackedPercentile: パーセント付きの積み重ねグラフを描きます。パーセントの積み重ねグラフは全体に対する各個別の項目の関係を百分率で示します。上記の例のスタイルをStackedPercentileに変更すると、棒の2つの部分の相対的なサイズがパーセンテージで表示されます。

XGrid: 棒グラフの水平グリッド線を表示します。

YGrid: 棒グラフの垂直グリッド線を表示します。

全てのスタイル設定をクリアするには、このプロパティにDefaultを設定します。これをプログラム上で行うには、次の構文を使います。

```
SET #GRAPH2 BARSTYLE(DEFAULT)
```

LineStyle プロパティ

LineStyleは折れ線グラフのスタイルを設定します。

LineStyleのプロパティ・リストを使用して、折れ線グラフのスタイルを設定します。LineStyleのドロップダウン・リストで次の中から複数の設定を選択できます：

NoLabels: グラフのX軸とY軸両方のラベルを非表示にします。

Stick: データ・ポイントからX軸に縦線を引きます。

XGrid: グラフの水平グリッド線を表示します。

YGrid: グラフの垂直グリッド線を表示します。

全てのスタイル設定をクリアするには、このプロパティにDefaultを設定します。これをプログラム上で行うには、次の構文を使います。

```
SET #GRAPH2 LINSTYLE(DEFAULT)
```

PieStyle プロパティ

PieStyleは円グラフのスタイルを設定します。

PieStyleのプロパティ・リストを使用して、円グラフのスタイルを設定します。 PieStyleのドロップダウン・リストで次の中から複数の設定を選択できます：

3D: グラフが立体に見えるようにします。

AutoArrangeLabels: ラベルが重ならないように整理します。

Exploded: 全ての円グラフのセグメントをパイの中心から少しだけ離します。

NoLabels: 円グラフの異なるセクションのラベルを非表示にします。

PerCent: ラベルを実際の大きさではなく、全体のパーセントで示します。

PerCentSymbol: 円グラフのラベルにパーセント記号(%)を表示します。

全てのスタイル設定をクリアするには、このプロパティにDefaultを設定します。これをプログラム上で行うには、次の構文を使います。

```
SET #GRAPH2 PIESTYLE(DEFAULT)
```

ScatterStyle プロパティ

ScatterStyleは散布図のスタイルを設定します。

ScatterStyleのプロパティ・リストを使用して、散布図のスタイルを設定します。 ScatterStyleのドロップダウン・リストで次の中から複数の設定を選択できます：

Curve: それぞれのデータ・セットを通る曲線を描きます。

NoLabels: グラフのX軸とY軸両方のラベルを非表示にします。

Pattern: Curve (曲線) 設定が選択されている時に、曲線にパターン線を使用します。

Solid: データ・ポイントを実線で結びます。

Stick: データ・ポイントからX軸まで縦の線を引きます。

SymbolColor: 凡例のテキストを記号と同じ色にします。

SymbolAtPoints: それぞれのポイントで記号を描きます。 実際に描かれる記号 (ダイヤモンド、四角、丸や箱など) はそれぞれのグラフ欄のプロパティです。

SymbolThick: Curve (曲線) 設定が選択されている時に、曲線に太い線を使用します。

XGrid: グラフの水平グリッド線を表示します。

YGrid: グラフの垂直グリッド線を表示します。

全てのスタイル設定をクリアするには、このプロパティにDefaultを設定します。 これをプログラム上で行うには、次の構文を使います。

```
SET #GRAPH2 SCATTERSTYLE(DEFAULT)
```

SurfaceStyle プロパティ

SurfaceStyleは等高線グラフのスタイルを設定します。

SurfaceStyleのプロパティ・リストを使用して、等高線グラフのスタイルを設定します。SurfaceStyleのドロップダウン・リストで次の中から複数の設定を選択できます：

ConnectLines: 線を結ぶだけで、表面を埋めません。

ConnectLinesInBlack: 黒の線で結びます。

NoLabels: グラフのX軸とY軸両方のラベルを非表示にします。

SolidSideWalls: 横の断面を描きます。

XGrid: グラフの水平グリッド線を表示します。

YGrid: グラフの垂直グリッド線を表示します。

全てのスタイル設定をクリアするには、このプロパティにDefaultを設定します。これをプログラム上で行うには、次の構文を使います。

```
SET #GRAPH2 SURFACESTYLE(DEFAULT)
```

ValueAt プロパティ

行・列の指標を使って、グラフのセルにアクセスします。

RESULT

リスト要素のバリエーション値です。

ROW

アクセスする行の整数インデックス

COLUMN

アクセスする列の整数インデックス

例

```
#OutputValue := #Graph1.ValueAt( 2 3 )
```

グラフのメソッド

SetValueメソッド

SetValueメソッド

グラフのセル値を設定します。

RESULT

設定が成功(TRUE)したかどうかを示すブール値です。

ROW

設定する行の整数インデックス

COLUMN

設定する列の整数インデックスVALUE

設定する値

例

```
If (#graph1.SetValue ( 2 1 "abc" ) <> True)
#OutputValue1 := "invalid value for graph"
Endif
```

基本リスト

基本リストは表示されません。基本リストはリストに関するヘルプ項目の1つのソースとしての役割を果たすために存在しているだけです。

[基本リストのプロパティ](#)

[基本リストのイベント](#)

[基本リストのメソッド](#)

基本リストのプロパティ

Appearance プロパティ	Messages プロパティ	ColumnResize プロパティ
Columns プロパティ	Modified プロパティ	DragColumns プロパティ
ColumnEllipses プロパティ	KeyboardPositioning プロパティ	EditorSizing プロパティ
ColumnHeaders プロパティ	ModifiedRules プロパティ	PrompterTabStop プロパティ
ColumnScroll プロパティ	NotificationStyle プロパティ	ViewpopupMenu プロパティ
FullRowSelect プロパティ	ShowItemHint プロパティ	ViewStyle プロパティ
Items プロパティ	ShowSelectionHilight プロパティ	Value parameter イベント
CurrentItem プロパティ	ShowSelection プロパティ	PrompterPosition プロパティ
Entries プロパティ	SelectionStyle プロパティ	PrompterImage プロパティ
FocusItem プロパティ	UsePicklist プロパティ	ShowPrompter プロパティ

Appearance プロパティ

Appearanceはリストやグリッドの外観の設定をします。

Appearanceプロパティを使って、リストやグリッドを3D効果あり（省略値）もしくはなし(Flat)で表示するかを指定します。

Columns プロパティ

Columnsにより、カラムのプロパティにアクセスできます。

実行時のみのプロパティです。

Columnsプロパティを使って、リスト、グリッドやツリービューのカラムのプロパティにアクセスします。

Columnsはリストのカラム全てのコレクションを代表するものです：例えば最初のカラムのプロパティにアクセスするにはColumnsとして参照します。

次のステートメントでは、リストビューの最初のカラムのDisplayPositionとWidthを取り出します。

```
Set Com(#std_num) Value(#ltvw_1.columns.displayposition)
```

```
Set Com(#std_num_1) Value(#ltvw_1.columns.width)
```

カラムの属性を記録しておく、後に元の状態に戻すときに便利です。次のコードでは、For/Endforループを利用してリスト内のカラムを反復し、DisplayPositionとWidth属性を取り出してグリッドに入れます。このコードをコピーして貼り付けることもできます。

```
Function Options(*DIRECT)
```

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('For/Endfor') Height(
```

```
Define_Com Class(#PRIM_LTVW) Name(#LTVW_1) Displayposition(1) Full
```

```
Define_Com Class(#PRIM_GRID) Name(#GRID_1) Captionnoblanklines(Tru
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_1) Displayposition(1) Paren
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_2) Displayposition(2) Paren
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_3) Displayposition(3) Paren
```

```
Define_Com Class(#PRIM_GDCL) Name(#GDCL_1) Caption('DisplayPositic
```

```
Define_Com Class(#PRIM_GDCL) Name(#GDCL_2) Caption('Width') Captio
```

```
Define_Com Class(#PRIM_PHBN) Name(#PHBN_1) Caption('Get Column A
```

```
Define_Com Class(#STD_NUM) Name(#STD_NUM)
```

```
Define_Com Class(#STD_AMNT) Name(#STD_AMNT)
```

```
EvtRoutine Handling(#com_owner.Initialize)
```

```
Select Fields(#LTVW_1) From_File(PSLMST)
```

```
Add_Entry To_List(#LTVW_1)
```

```
Endselect
```

```
Endroutine
```

```
EvtRoutine Handling(#PHBN_1.Click)
```

```
For Each(#Current) In(#ltvw_1.Columns)
```

```
Set Com(#std_num) Value(#current.displayposition)
```

```
Set Com(#std_amnt) Value(#current.width)
Add_Entry To_List(#GRID_1)
Endfor
Endroutine
End_Com
```

ColumnEllipses プロパティ

ColumnEllipses プロパティを使って、カラム・ヘッダーのテキストが入りきらなかった場合にどのように処理するかを制御します。このプロパティは、次の値を取ることができます：

None: テキストは切り捨てられます。

End: 文字列の一番最後がカラムに入りきらない場合、これが切り捨てられ、省略記号が追加されます。

ColumnHeaders プロパティ

ColumnHeadersはカラム・ヘッダーのオン・オフの設定をします。

ColumnHeadersプロパティを使用して、カラム・ヘッダーを表示したり、非表示にしたりします。

リストビューの場合、このプロパティはViewStyleがReportの時のみ適用されます。

ColumnScroll プロパティ

ColumnScrollは水平スクロールを制御します。

ColumnScrollプロパティを使って、水平のスクロールを制御します。このプロパティがTrueの時、コントロールはカラムごとにスクロールされます。このプロパティがFalseの時、コントロールはピクセルごとにスクロールされます。

FullRowSelect プロパティ

FullRowSelectはフル行選択のオン・オフを設定します。

FullRowSelectプロパティを使用して、コントロールがフル行選択（最初の
カラムからだけでなく、行のどのカラムからでも選択できる）をサポート
するかどうかを制御します。

Items プロパティ

Items プロパティにより、コンポーネント内の項目の属性にアクセスできます。

実行時のみのプロパティです。

Items プロパティを使って、リスト、グリッド、ツリービュー、コンボ・ボックス、プロパティ・シートやメニューの項目の属性にアクセスできます。

Items はコンポーネントの項目全てのコレクションを代表するものです：例えば最初の項目のプロパティにアクセスするには Items として参照します。

次のステートメントは submenu 内の最初のメニュー項目を無効にします：

```
Set Com(#smnu_1.items) Enabled(False)
```

For/EndFor ループを利用して、コンポーネントの全ての項目のプロパティにアクセスすることもできます。次の例では、#PHBN_1 がクリックされると #SMNU_1 サブメニューのメニュー項目はすべて無効になり、#PHBN_2 がクリックされるとメニュー項目は有効になります。

```
Evtroutine Handling(#PHBN_1.Click)
For Each(#Current) In(#smnu_1.Items)
Set Com(#Current) Enabled(False)
Endfor
Endroutine
Evtroutine Handling(#PHBN_2.Click)
For Each(#Current) In(#smnu_1.Items)
Set Com(#Current) Enabled(True)
Endfor
Endroutine
```

CurrentItem プロパティ

CurrentItemは現在処理中の項目です。

実行時のみ有効です。

CurrentItemは、現在アプリケーションで操作されているリスト・ボックス、リスト・ビュー、コンボ・ボックス、またはグリッドの項目です。CurrentItemは、（FocusItemと同じように）ユーザーが項目のフォーカスを移動すると設定され、またプログラムのGET_ENTRYコマンドやSELECTLISTコマンドで設定することもできます。

現在のアイテムの値は、例えばデータベースI/Oコマンドのキーとして使用されます。

ツリー左側のCurrentItemでダブルクリックして、リスト項目にあるプロパティ、メソッドやイベントを見ることができます。

Entries プロパティ

Entriesはリストのエントリ数を表示します。

実行時のみ、読み取り専用プロパティです。

Entriesプロパティを使って、リスト・タイプのコンポーネント、例えばコンボ・ボックス、グリッド、複数行編集ボックスやリスト・ボックスなど、にあるエントリーの数を知ることができます。

AddEntryModeがMultiplePerLineの複数行編集ボックスでは、1行に複数のエントリーが含まれる可能性があることに注意してください。

FocusItem プロパティ

FocusItemはフォーカスの当たっている項目です。
実行時のみ有効です。

FocusItemはリスト、ツリー・ビュー、またはグリッド内のフォーカスが当たっている項目です。

リスト内の全ての項目にはFocusプロパティがあり、TrueまたはFalseに設定できます。FocusプロパティにTrueが設定されている項目がFocusItemです。一度に1つの項目にだけフォーカスが当たります。

項目のFocusプロパティがTrueに設定されていると、SelectedプロパティもTrueになります。（詳細は「[CurrentItem プロパティ](#)」を参照してください。）

ツリー・ビュー項目には追加のプロパティがあり、フォーカスが当たっている項目のこれらのプロパティを設定する際にCurrentItemを使うと便利です。例えば、フォーカスのある項目に"expand"を設定する場合、次のようなステートメントを書きます。

```
set com(#trvw_1.FocusItem) Expanded(True)
```

ツリー左側のFocusItemでダブルクリックして、リスト項目にあるプロパティ、メソッドやイベントを見ることができます。

Messages プロパティ

Messagesはステータスバーのメッセージのコレクションです。

Messagesプロパティを使って、ステータスバー内の全てのメッセージのコレクションにアクセスします。

次のClickイベント・コードは、メッセージ・コレクション内を繰り返し通り、一次レベルのメッセージ・テキストをリストに追加します。

```
Evtroutine Handling(#STbr_1.click) Options(*NOCLEARERRORS *NOCLE  
For Each(#Message) In(#stbr_1.messages)  
Change Field(#STD_TEXT) To('#MESSAGE.FIRSTLEVELTEXT')  
Add_Entry To_List(#TRACE)  
Endfor  
Endroutine
```

コレクションのメッセージには、FirstLevelText、SecondLevelText、そしてHasSecondLevelText (True/False)というプロパティがあります。

Modified プロパティ

Modifiedは項目またはセルが変更されたかどうかを示します。

Modifiedプロパティを使って、項目またはグリッド・セルのコンテンツが変更されたかどうかを確認します。

このプロパティはユーザーがコンテンツに変更を加えると、自動的にTrueに設定されます。プログラム上でコンテンツが変更された場合は、Modifiedフラグは設定されません。

グリッドでは、ユーザーがセルの内容を変更した時にセルのModifiedプロパティがTrueに設定されます。この時、グリッド項目とグリッドそのもののModifiedプロパティがTrueに設定されます。グリッドのModifiedプロパティをFalseに設定すると、全ての項目とセルのModifiedプロパティもFalseに設定されます。グリッド項目のModifiedプロパティをFalseに設定すると、その項目の全てのセルのModifiedプロパティがFalseに設定されます。

次のイベント・ルーチンは、現在のグリッド項目のModifiedの状態をチェックし、変更された情報をファイルに挿入します。

```
evtroutine handling(#UPDATE_BUTTON.Click)
  selectlist #grid_1
  if cond('#grid_1.currentitem.Modified *eq TRUE')
    update fields(#surname #givenname) in_file(pslmst) with_key(#empno) val_err
    if_status is_not(*okay)
  upd_entry #grid_1
  endif
endif
endselect
endroutine
```

KeyboardPositioning プロパティ

KeyboardPositioningは、リストがキーボード文字から項目を検索する方法を設定します。

KeyboardPositioningプロパティを使って、リスト・ビューまたはツリー・ビューがキーボード文字からどのように項目を検索するかを指定します。2つの値が選択できます：

- FirstColumnを指定すると、入力されたキーボード文字に一致するものを探す時に、カラムの一番左にあるカラムのテキストが使用されます。
- SortColumnを指定すると、入力されたキーボード文字に一致するものを探す時に、プライマリ・ソートのカラムのテキストが使用されます。

ModifiedRules プロパティ

ModifiedRulesはModifiedプロパティがTrueに設定されている時の制御を行います。

リスト、コンボ・ボックス、ツリー・ビューやプロパティ・シート内で、フォーカスや選択がリスト内の別の項目に移された時にModifiedプロパティがTrueに設定されるべきかどうかをModifiedRulesで指定します。

フォーカスや選択が変更された時、省略値ではModifiedプロパティはTrueに設定されます。ModifiedRulesプロパティは、フォーカス(IgnoreFocusChanges)や選択(IgnoreSelectionChanges)が変更された時に、フォーカスの変更を無視するように設定することができます。

NotificationStyle プロパティ

NotificationStyleはイベントがどのように起動されるかを制御します。

省略値では、リスト、コンボ・ボックス、ツリーまたはグリッド項目がプログラム上で処理される時、項目のイベント（例えば ItemChanged、ItemGotSelection、ItemGotFocus、ItemLost focusやItemLostSelectionなど）は起動されません。ユーザーにより項目が操作された時（マウスやキーボードを使用して）のみ、これらのイベントは起動されます。

項目をプログラム上で処理している時に項目のイベントを起動したい時は、コントロールのNotificationStyleをProgramに設定します。こうすることで、プログラムであれ、ユーザーであれ、項目が操作された時はいつも同じイベントが起動されるようになります。

次のサンプル・アプリケーションをコピーし貼り付けて、ツリー・ビュー項目がプログラムでどのように選択されるかを見ることが出来ます。

```
FUNCTION OPTIONS(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Height(306) Left(333) Top(14
Define_Com Class(#PRIM_TRVW) Name(#TRVW_1) Componentversion(1)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_1) Level(1) Parent(#TRVW
Define_Com Class(#PRIM_TVCL) Name(#TVCL_2) Displayposition(1) Key|
Define_Com Class(#PRIM_TVCL) Name(#TVCL_3) Displayposition(1) Key|
Define_Com Class(#PRIM_PHBN) Name(#PHBN_1) Caption('Click Here to |
EVTROUTINE handling(#com_owner.Initialize)
SET #com_owner caption(*component_desc)
select *all deptab
SELECT FIELDS(#SURNAME #GIVENAME) FROM_FILE(PSLMST1) WI
use bconcat with_args(#givenname #surname) to_get(#fullname)
add_entry #trvw_1
endselect
endselect
ENDROUTINE
EVTROUTINE HANDLING(#PHBN_1.Click)
selectlist #trvw_1
IF COND('#fullname = "VERONICA BROWN"')
set #trvw_1.CurrentItem selected(True)
leave
endif
```

```
endselect
ENDROUTINE
EVTROUTINE HANDLING(#TRVW_1.ItemGotSelection) OPTIONS(*NOC
USE BUILTIN(MESSAGE_BOX_SHOW) WITH_ARGS(OK OK INFORMA
set #trvw_1.CurrentItem ensurevisible(true)
ENDROUTINE
END_COM
```

ShowItemHint プロパティ

ShowItemHintはヒントのオン・オフを切り替えます。

ShowItemHintプロパティを使って、項目のヒントを表示するかどうかの制御を行います。ヒントのテキストは、ItemHintTextイベントのCaptionパラメータで指定されます。

ShowSelectionHilight プロパティ

ShowSelectionHilightは選択された項目がどのように表示されるかを制御します。

ShowSelectionHilightプロパティを使用して、選択の濃い青の協調表示をオンまたはオフに設定します。強調表示がFalseに設定されていると、選択は灰色で表示されます。

ShowSelection プロパティ

ShowSelectionは選択を表示したり非表示にしたりします。

ShowSelectionを利用して、フォーカスがない時に、コントロールに選択を表示するかどうかを指定します。

このプロパティにTrueを設定すると、コントロールが選択されていない時に選択を表示します。

SelectionMode プロパティ

SelectionModeにはSingle、MultipleまたはExtendedがあります。

SelectionModeプロパティを使って、項目が選択される方法を設定します。

SelectionModeがSingleの場合、ユーザーは一度に1項目だけしか選択できません。

SelectionModeがMultipleの場合は、ユーザーは複数の項目を選択できます。

SelectionModeがExtendedの場合は、ユーザーはひとまとまりのエントリーを選択できます。SelectionModeはリストではサポートされないことに注意してください。

グリッドではSelectionModeはWholeRowに設定することもできます。このスタイルでは行全体を一度に選択します。

UsePicklist プロパティ

UsePicklistはピックリストを利用するかどうかを指定します。

UsePicklistプロパティを利用して、フィールドにピックリスト・ビジュアルライゼーションがある時に列にピックリスト・ビジュアルライゼーションを使用するかどうか制御します。

このプロパティにFalseを設定すると、列は通常の列として表示されます。

ColumnResize プロパティ

ColumnResize プロパティを使って、列のサイズが調整できるかどうかの制御を行います。

DragColumns プロパティ

DragColumnsプロパティを使用して、ユーザーがマウスをドラッグして列の位置を変更できるかどうかを制御します。

このプロパティがTrueに設定されていると、ユーザーは列の相対的な位置を変更することができます。

EditorSizing プロパティ

EditorSizing プロパティを使って、EditorPartとして使用される再利用可能パーツをどのように表示するかを指定します。

このプロパティの値がDefaultの時、再利用可能パーツはグリッドの行と列の高さの範囲内に表示されます。

このプロパティの値がEditorの時、再利用可能パーツはグリッドの行と列の高さに関係なく、元のサイズで表示されます。

PrompterTabStop プロパティ

PrompterTabStopプロパティを使用して、プロンプター・ボタン上のタブ・ストップを有効・無効にします。

ユーザーがキーボードからプロンプター・ボタンにアクセスできるようにするにはこのプロパティをTrueに設定します。

このプロパティはTabStopとShowPrompterがTrueに設定されている時のみ有効です。

ViewpopupMenu プロパティ

ViewpopupMenuは表示エリアのポップアップ・メニューを指定します。

ViewpopupMenuは、ユーザーがコントロールの表示エリアで右クリックした時にコンテキスト依存のポップアップ・メニューを表示します。

表示エリアとは、リスト項目に使用されていない"白いスペース"のことです。リスト・ビュー・コントロールの場合は一番右端の列の右側のカラム・ヘッダーも表示エリアで、グレーのカラム・ヘッダー・ボタンとして表示されます。

ViewStyle プロパティ

ViewStyleはリストやツリーの表示方法を設定します。

ViewStyleプロパティを使って、リスト・ビューまたはツリー・ビューがどのように表示されるかを指定します。リスト・ビューは次の4種類のスタイルから選択できます：

- Iconスタイルは、大きいアイコンで項目を表示します。
- Smalliconスタイルは、小さいアイコン（16 x 16 ピクセル）がリスト項目の前に表示されます。
- Listスタイルは、全ての項目が別途一列に並べられること以外は、Smalliconスタイルと同じです。
- Reportスタイルはカラム・ヘッダー付きのリストにリスト項目を表示します。

ツリー・ビューの表示スタイルはLevelledまたはUnLevelledです。

通常のツリー・ビューはLevelledです。

UnLevelledツリーでは、カラム・ヘッダー付きの複数のデータ列が表示されます。（これは事実上ツリーとリスト・ビューが結合されたものです。）Unlevelledツリー・ビューは、LANSAエディターで広く使用されています。（例えばファイル保守のフィールド・ツリー・ビューなど）

次の例は、Unlevelledツリー・ビューがどのようにして作成されるかを示しています。これを利用するには、フルRDMLX対応のフォームが必要です。

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(471) Clientwidth
Define_Com Class(#PRIM_ATLM) Name(#ATLM_1)
Define_Com Class(#PRIM_TRVW) Name(#TRVW_1) Columnbuttonheight(1
Define_Com Class(#PRIM_ATLI) Name(#ATLI_1) Attachment(Center) Mana
Define_Com Class(#PRIM_TVCL) Name(#TVCL_1) Caption('Name') Captio
Define_Com Class(#PRIM_TVCL) Name(#TVCL_2) Caption('Description') C
Evtroutine Handling(#com_owner.CreateInstance)
Set Com(#com_owner) Caption(*component_desc)
#Com_owner.Load_tree
Endroutine
Mthroutine Name(Load_tree)
#Com_owner.Add_departments
Endroutine
```

```

Mthroutine Name(Add_Departments)
Define_Map For(*Input) Class(#prim_tvit) Name(#Department_item) Reference(*dynamic)
Select Fields(#deptment #deptdesc) From_File(deptab)
#com_owner.Add_Entry Column_1(#Deptment) Column_2(#Deptdesc) Tree_Item_1
#Com_Owner.Add_Sections( #deptment #department_item )
Endselect
Endroutine
Mthroutine Name(Add_Sections)
Define_Map For(*Input) Class(#deptment) Name(#Department)
Define_Map For(*Input) Class(#prim_tvit) Name(#Parent_item) Mandatory(*input)
Define_Map For(*Input) Class(#prim_tvit) Name(#Section_item) Reference(*dynamic)
Select Fields(#section #secdesc) From_File(sectab) With_Key(#department)
#com_owner.Add_Entry Column_1(#section) Column_2(#secdesc) Parent_Item_1
#com_owner.Add_employees( #Department #section #Section_item )
#trvw_1.currentitem.MarginBottom := 5
Endselect
#section_item.marginbottom := 5
Endroutine
Mthroutine Name(Add_Employees)
Define_Map For(*Input) Class(#deptment) Name(#Employee_Department)
Define_Map For(*Input) Class(#Section) Name(#Employee_Section)
Define_Map For(*Input) Class(#prim_tvit) Name(#Parent_item) Mandatory(*input)
Select Fields(#Empno #Givenname #surname) From_File(pslmst1) With_Key(#deptment)
#com_owner.Add_Entry Column_1(#Empno) Column_2(#Givenname.trim + ' ' + #surname)
Endselect
Endroutine
Mthroutine Name(Add_Entry)
Define_Map For(*Input) Class(#std_Name) Name(#column_1)
Define_Map For(*Input) Class(#std_Descl) Name(#column_2)
Define_Map For(*Input) Class(#prim_tvit) Name(#Parent_item) Mandatory(*input)
Define_Map For(*Input) Class(#prim_icon) Name(#Item_Icon) Mandatory(*input)
Define_Map For(*output) Class(#prim_tvit) Name(#Tree_item) Mandatory(*input)
#std_name := #Column_1
#std_Descl := #Column_2
Add_Entry To_List(#trvw_1)
#trvw_1.Currentitem.image <= #item_icon
#trvw_1.Currentitem.parentitem <= #parent_item
#tree_item <= #trvw_1.currentitem
Endroutine

```

End_Com

Value parameter イベント

Value parameterは変更されたエディターの内容を返します。

返される値はバリエーションです。variantファンクションを使用して、返された値のタイプを決定できます。

PrompterPosition プロパティ

PrompterPositionはプロンプターの位置を設定します。

PrompterPositionプロパティを利用して、このコンポーネントのプロンプターをどこに位置づけるかを制御します。

プロンプターの位置はコンポーネントの上部、下部、左、右に表示できます。このプロパティの値がNoneに設定されていると、プロンプターは表示されません。

プロンプターを使用できるようにするには、BEGIN_COM Role(*prompter)を定義し、それに関連するコーディングを手作業で行わなければならないことに注意してください。

PrompterImage プロパティ

PrompterImageはプロンプト・ボタンのイメージを設定します。

PrompterImageプロパティを使用して、省略値のプロンプト・イメージを別のビットマップに変更します。

ShowPrompter プロパティ

ShowPrompterはプロンプターの表示のオン・オフを切り替えます。

ShowPrompter プロパティを利用して、このフィールドのプロンプターを表示するかどうかを制御します。このプロパティの値は、TrueまたはFalseになります。

プロンプターは、ユーザー定義のフォームです。フィールドにプロンプターがある場合、省略記号の付いたボタンがエントリー・フィールドに追加され、ユーザーがプロンプターを表示できます。実行時このプロンプター・ボタンがクリックされると、プロンプター・コンポーネントが開き、その中から値を選択できるようになります。

プロンプターを使用できるようにするには、BEGIN_COM Role(*prompter)を定義し、それに関連するコーディングを手作業で行わなければならないことに注意してください。

例：

```
Begin_Com Role(*prompter #qap100) Name(#PROMPTER)
Defaultprompter(True)
End_Com
```

プロンプターについての詳細は、『*Visual LANSА 開発者ガイド*』の「[フィールド・クラス・リスト](#)」のプロンプター・クラスを参照してください。

基本リストのイベント

ItemGotFocus イベント ト	ItemLostFocus イベント ItemGotSelection イベント ト	ItemLostSelection イベント ト
ItemGotFocusAccept イベント	ItemGotSelectionAccept イベント	ItemLostSelectionAccept イベント
ItemChangedAccept イベント	ItemLostFocusAccept イベント	ItemChanged イベント
ItemHintText イベント	ViewDoubleClick イベント	

ItemGotFocus イベント

ItemGotFocusは項目にフォーカスが当たった時に起動します。

ItemGotFocusイベントは項目がフォーカスを得た時に起動されます。これは通常、ユーザーが項目でクリックした時に起きます。リスト・タイプのコンポーネントの場合、このイベントは実行時プロパティのFocusItemを設定します。

ItemGotFocusAccept イベント

ItemGotFocusAcceptは項目にフォーカスが当たった時に起動します。

ItemGotFocusAcceptイベントは項目がフォーカスを得た時に起動されま
す。これは通常、ユーザーが項目でクリックした時に起きます。

ItemGotFocusAcceptイベントがItemGotFocusイベントと違うのは、この
イベントを続行するか否かの制御ができることです。このイベントの
AcceptパラメータをFalseに設定すると、イベントは実行されません。

例えば、グリッド・セルにフォーカスがあり、フォーカスを別のセルに
移動すると、次のようなイベントが起動します。

- 最初のセルのItem LostFocusAcceptとItemLostSelectionAccept イベント
新しいセルが別の列にある場合は、最初の列にLostFocusAccept と
LostSelectionAcceptイベント

- 2番目のセルのItem GotFocusAcceptとItemGotSelectionAccept イベント
新しいセルが別の列にある場合は、2番目のの列にGotFocusAccept と
GotSelectionAcceptイベント

これらのイベントのいずれかのイベントのAcceptにFalseが設定されてい
ると、操作はキャンセルされます。すべてのイベントのAcceptにTrueが
設定されている場合は、次のイベントが実行されます。

- 最初のセルのItemLostFocusとItemLostSelection、同時に最初の列の
LostFocusとLostSelection
- 2番目のセルのItemGotFocusとItemGotSelection、同時に2番目の列の
GotFocusとGotSelection

Accept処理の目的は、ユーザーが行うことに対して、プログラムによる
制御を更に大きくすることです。何かが変更されようとする時は、その
処理を続ける前に変更を受け入れるかどうかの制御を常にプログラム上
で行うことができます。

Accept パラメータ

Reason パラメータ

Continue パラメータ

Value パラメータ

ItemChangedAccept イベント

ItemChangedAccept イベントは項目が変更されようとするときに起動します。項目が変更されようとする時に、ItemChangedAccept イベントが起動します。

ItemChangedAccept イベントにより、プログラムで変えられようとしている変更を受け入れるかどうかを事前に制御することができます。この処理は、accept (受理) と commit (実行) という 2 つの部分に分けることができます。変更を受け入れられない場合、この変更は行われません。次のようなケースでこの機能がよく利用されます：

- ユーザーが無効なデータを入力
- ItemChangedAccept イベントが起動
- プログラムによりメッセージ・ボックスが表示され、入力データが有効かどうかユーザーに確認
- データが無効の場合、Continue に False が設定され、ユーザーがデータを再入力できるようフォーカスが編集場所に残る

ItemChangedAccept イベントの典型的な使用例としては、ユーザーがフィールド #EMPNO の編集可能な欄に、000000 の値を入力し、その後グリッド内の別のところでクリックして選択を変更します。そうすると、ItemChangedAccept イベントが起動し、入力された値を受理するかどうかの確認が行われます。その後の処理は以下の通りです：

- 値を受理 (Accept パラメータを True に設定)
- 値を受理しない (Accept パラメータを False に設定)
- 処理を引き続き実行 (Continue パラメータを True に設定) クリックされた項目が選択されます。
- 処理を停止 (Continue パラメータを False に設定) クリックされた項目は選択されず、フォーカスは現在の編集場所に残ります。

次のサンプル・アプリケーションは ItemChangedAccept 処理の使い方を示しています。このコードをコピーして貼り付けることもできます。この例を実行するには、デモンストレーション・ファイル PSLMST が必要です。

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('Accept Processing E:
* Grid Definition
Define_Com Class(#PRIM_GRID) Name(#LIST) Columnbuttonheight(33) Di
```

```

Define_Com Class(#PRIM_GDCL) Name(#COL1) Displayposition(1) Parent(
Define_Com Class(#PRIM_GDCL) Name(#COL2) Displayposition(2) Parent(
Define_Com Class(#PRIM_GDCL) Name(#COL3) Displayposition(3) Parent(
Define_Com Class(#PRIM_GDCL) Name(#COL4) Displayposition(4) Parent(
* The rest....
Define_Com Class(#PRIM_GPBX) Name(#GPBX_7) Caption('Edit Events') I
Define_Com Class(#PRIM_CKBX) Name(#CKBX_ACCEPT) Buttonstate(Cl
Define_Com Class(#STD_NUM) Name(#STD_NUM)
Define_Com Class(#PRIM_CKBX) Name(#CKBX_CONTINUE) Buttonstate
Define_Com Class(#PRIM_VAR) Name(#NEW_VALUE)
Define_Com Class(#PRIM_STBR) Name(#STBR_1) Displayposition(3) Heig
* =====
*
* Form Operations
*
* =====
EvtRoutine Handling(#COM_OWNER.Initialize) Options(*NOCLEARMESS/
* Populate grid
Select Fields(*ALL) From_File(PSLMST)
Add_Entry To_List(#LIST)
Endselect
Endroutine
* =====
*
* Grid Accept Event Testing
*
* =====
EvtRoutine Handling(#LIST.ItemChangedAccept) Accept(#ACCEPT) Continu
Set Com(#NEW_VALUE) Value(#NEW)
Use Builtin(BCONCAT) With_Args('Accept value of' #NEW_VALUE.STRIN
Message Msgid(DCM9993) Msgf(DC@M01) Msgdta(#STD_TEXT)
If Cond('#CKBX_ACCEPT.ButtonState = unChecked')
Set Com(#ACCEPT) Value(false)
Endif
If Cond('#CKBX_CONTINUE.ButtonState = unChecked')
Set Com(#CONTINUE) Value(false)
Endif
Endroutine
End_Com

```

パラメータ

Accept パラメータ

Reason パラメータ

Continue パラメータ

Value パラメータ

Accept パラメータ

AcceptパラメータをTrueに設定すると、イベントの処理が進められ、Falseにするとイベントはキャンセルされます。

Reason パラメータ

Reasonパラメータはイベントが起動された理由を示します。
理由は次のいずれかです：

- LeaveControl
- ResetContent
- ChangeSelection
- CommitPendingChanges
- ChangeChecked (リスト・ビューやツリービューのチェック・ボックス選択の変更)

Continue パラメータ

ContinueパラメータをTrueに設定すると、このイベントを起動した処理を続行します。

キャンセルできる処理は限られていることに注意してください。例えば、CommitPendingChangesや LeaveControlはキャンセルできませんが、ChangeSelectionはキャンセルできます。

Value パラメータ

Valueパラメータは変更されたばかりの値を返します。このパラメータを使用して、フィールドの元となる値が変更される前に、変更された値の妥当性確認を行います。

返される値はバリエーションです。variant関数を使用して、返された値のタイプを決定できます。

ItemHintText イベント

ItemHintTextはヒント用のテキストが取り出された時に起動します。

ItemHintTextイベントは項目のヒントが取り出された時に起動するイベントです。これはカーソルが項目の上に移動した時に、少しの遅れを取って起動されます。

Captionパラメータに値を割り当てて、ヒントのテキストを提供します。

次の例はリスト欄のテキストを連結し、項目のヒントに表示します。この例をコピーしてフォームに貼り付けることもできます。

```
Function Options(*DIRECT)
```

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Height(395) Left(402) Top(10
```

```
Define_Com Class(#PRIM_LTVW) Name(#LISTVIEW) Displayposition(1) F
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_1) Displayposition(1) Paret
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_2) Displayposition(2) Paret
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_3) Displayposition(3) Paret
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_4) Parent(#LISTVIEW) So
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_5) Parent(#LISTVIEW) So
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_6) Parent(#LISTVIEW) So
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_7) Parent(#LISTVIEW) So
```

```
Define_Com Class(#PRIM_LVCL) Name(#LVCL_8) Parent(#LISTVIEW) So
```

```
Evtroutine Handling(#COM_OWNER.Initialize) Options(*NOCLEARMESS/
```

```
Select Fields(*ALL) From_File(PSLMST)
```

```
Add_Entry To_List(#LISTVIEW)
```

```
Endselect
```

```
Endroutine
```

```
Evtroutine Handling(#LISTVIEW.ItemHintText) Options(*NOCLEARMESS/
```

```
Use Builtin(BCONCAT) With_Args('This employee is named' #GIVENAME #
```

```
Use Builtin(BCONCAT) With_Args(#SYSVAR$AV 'and lives at' #ADDRESS
```

```
Set Com(#UseCaption) Value(#SysVar$av)
```

```
Endroutine
```

```
End_Com
```

ItemLostFocus イベント

ItemLostFocusは他の項目にフォーカスが移った時に起動します。

ItemLostFocusイベントは項目がフォーカスを失った時に起動されます。
これは通常、ユーザーが別の項目でクリックした時に起きます。

ItemGotSelection イベント

項目が選択されるとItemGotSelectionイベントが起動します。

使用されているコンポーネントが複数項目選択をサポートしない場合、選択された項目がこのイベントの直後にCurrentItemになります。

複数項目選択をサポートするコンポーネントでは、このイベントは項目のSelectedプロパティをTrueに設定します。フォーカスを受ける項目がCurrentItemになります。

グラフ・コンポーネントでは、ユーザーがパイ・セグメントや棒グラフの棒などのデータ項目をクリックした時にこのイベントが起動されます。

ItemGotSelectionAccept イベント

ItemGotSelectionAcceptは項目が選択される時に起動します。

ItemGotSelectionAcceptイベントは、項目が選択されると起動されます。

ItemGotSelectionAcceptイベントがItemGotSelectionイベントと違うのは、このイベントを続行するか否かの制御ができることです。このイベントのAcceptパラメータをFalseに設定すると、イベントは実行されません。

例えば、グリッド・セルにフォーカスがあり、フォーカスを別のセルに移動すると、次のようなイベントが起動します。

- 最初のセルのItemLostFocusAcceptとItemLostSelectionAccept イベント
新しいセルが別の列にある場合は、最初の列にLostFocusAccept とLostSelectionAcceptイベント
- 2番目のセルのItem GotFocusAcceptとItemGotSelectionAccept イベント
新しいセルが別の列にある場合は、2番目のの列にGotFocusAccept とGotSelectionAcceptイベント

これらのイベントのいずれかのイベントのAcceptにFalseが設定されていると、操作はキャンセルされます。すべてのイベントのAcceptにTrueが設定されている場合は、次のイベントが実行されます。

- 最初のセルのItemLostFocusとItemLostSelection、同時に最初の列のLostFocusとLostSelection
- 2番目のセルのItemGotFocusとItemGotSelection、同時に2番目の列のGotFocusとGotSelection

Accept処理の目的は、ユーザーが行うことに対して、プログラムによる制御を更に大きくすることです。何かを変更されようとする時は、その処理を続ける前に変更を受け入れるかどうかの制御を常にプログラム上で行うことができます。

ItemLostFocusAccept イベント

ItemLostFocusAcceptは項目がフォーカスを失う時に起動します。

ItemLostFocusAcceptイベントは項目がフォーカスを失うと起動されま
す。

ItemLostFocusAcceptイベントがItemLostFocusイベントと違うのは、この
イベントを続行するか否かの制御ができることです。このイベントの
AcceptパラメータをFalseに設定すると、イベントは実行されません。

例えば、グリッド・セルにフォーカスがあり、フォーカスを別のセルに
移動すると、次のようなイベントが起動します。

- 最初のセルのItemLostFocusAcceptとItemLostSelectionAccept イベント
新しいセルが別の列にある場合は、最初の列にLostFocusAccept と
LostSelectionAcceptイベント
- 2番目のセルのItem GotFocusAcceptとItemGotSelectionAccept イベント
新しいセルが別の列にある場合は、2番目のの列にGotFocusAccept と
GotSelectionAcceptイベント

これらのイベントのいずれかのイベントのAcceptにFalseが設定されてい
ると、操作はキャンセルされます。すべてのイベントのAcceptにTrueが
設定されている場合は、次のイベントが実行されます。

- 最初のセルのItemLostFocusとItemLostSelection、同時に最初の列の
LostFocusとLostSelection
- 2番目のセルのItemGotFocusとItemGotSelection、同時に2番目の列の
GotFocusとGotSelection

Accept処理の目的は、ユーザーが行うことに対して、プログラムによる
制御を更に大きくすることです。何かに変更されようとする時は、その
処理を続ける前に変更を受け入れるかどうかの制御を常にプログラム上
で行うことができます。

ItemLostSelection イベント

ItemLostFocusは別の項目が選択された時に起動します。

ある項目がもう選択されていない状態の時にItemLostSelectionイベントが起動します。

ItemLostSelectionAccept イベント

ItemLostSelectionAcceptは項目が選択を失う時に起動します。

ItemLostSelectionAcceptイベントは項目が選択を失うと起動されます。

ItemLostSelectionAcceptイベントがItemLostSelectionイベントと違うのは、このイベントを続行するか否かの制御ができることです。このイベントのAcceptパラメータをFalseに設定すると、イベントは実行されません。

例えば、グリッド・セルにフォーカスがあり、フォーカスを別のセルに移動すると、次のようなイベントが起動します。

- 最初のセルのItemLostFocusAcceptとItemLostSelectionAccept イベント
新しいセルが別の列にある場合は、最初の列にLostFocusAccept とLostSelectionAcceptイベント
- 2番目のセルのItem GotFocusAcceptとItemGotSelectionAccept イベント
新しいセルが別の列にある場合は、2番目のの列にGotFocusAccept とGotSelectionAcceptイベント

これらのイベントのいずれかのイベントのAcceptにFalseが設定されていると、操作はキャンセルされます。すべてのイベントのAcceptにTrueが設定されている場合は、次のイベントが実行されます。

- 最初のセルのItemLostFocusとItemLostSelection、同時に最初の列のLostFocusとLostSelection
- 2番目のセルのItemGotFocusとItemGotSelection、同時に2番目の列のGotFocusとGotSelection

Accept処理の目的は、ユーザーが行うことに対して、プログラムによる制御を更に大きくすることです。何かを変更されようとする時は、その処理を続ける前に変更を受け入れるかどうかの制御を常にプログラム上で行うことができます。

ItemChanged イベント

ItemChangedは項目が変更された時に起動します。

ItemChangedイベントはツリー・ビュー、リスト・ビューやグリッドの項目が修正されると起動します。

ViewDoubleClick イベント

ViewDoubleClickは、ユーザーが表示エリアでダブル・クリックすると起動されます。

ユーザーがコントロールの表示エリアでダブル・クリックするとViewDoubleClickが起動します。

表示エリアとは、リスト項目に使用されていない"白いスペース"のことです。リスト・ビュー・コントロールの場合は一番右端の列の右側のカラム・ヘッダーも表示エリアで、グレーのカラム・ヘッダー・ボタンとして表示されます。

ViewDoubleClickイベントが起動する場所ではDoubleClickイベントは起動しません。

Explorerコンポーネントでは、表示スタイルがFileListBoxまたはDirectoryListBoxの場合、ViewDoubleClickは適用されません。

Explorerでの異なるクリック・イベントを見るには、次の例をコピーして貼り付けてください：

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(539) Clientwidth
Define_Com Class(#PRIM_DCBX) Name(#DCBX_1) Displayposition(1) File
Define_Com Class(#PRIM_DCBX) Name(#DCBX_2) Displayposition(2) Dis
Define_Com Class(#STD_TEXT.Visual) Name(#FILENAME) Caption('FileN
Define_Com Class(#STD_TEXT.Visual) Name(#PATH) Caption('Path') Displa
Define_Com Class(#STD_TEXT.Visual) Name(#PATHTYPE) Caption('Pathty
Define_Com Class(#STD_TEXT.Visual) Name(#FILENAME1) Caption('Filef
Define_Com Class(#STD_TEXT.Visual) Name(#PATH1) Caption('Path Before
Define_Com Class(#STD_TEXT.Visual) Name(#PATHTYPE1) Caption('Patht
Define_Com Class(#PRIM_STBR) Name(#STBR_1) Displayposition(9) Heig
Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
Endroutine
Evtroutine Handling(#DCBX_2.ItemGotSelection) Path(#w_path) Name(#w_f
Set Com(#FILENAME) Value(#W_FILENAME)
Set Com(#PATH) Value(#W_PATH)
Set Com(#PATHTYPE) Value(#W_PATHTYPE)
Message Msgtxt('Item clicked in Right Explorer Component')
Endroutine
Evtroutine Handling(#DCBX_1.ItemGotSelection) Path(#w_path) Name(#w_f
```

```
Set Com(#FILENAME) Value(#W_FILENAME)
Set Com(#PATH) Value(#W_PATH)
Set Com(#PATHTYPE) Value(#W_PATHTYPE)
Message Msgtxt('Item clicked in Left Explorer Component')
Endroutine
Evroutine Handling(#DCBX_2.ViewDoubleClick)
Set Com(#FILENAME1) Value(#DCBX_2.Filename)
Set Com(#PATH1) Value(#DCBX_2.Path)
Set Com(#PATHTYPE1) Value(#DCBX_2.PathType)
Message Msgtxt('View double clicked Right Explorer Component')
Endroutine
Evroutine Handling(#DCBX_1.ViewDoubleClick)
Set Com(#FILENAME1) Value(#DCBX_1.Filename)
Set Com(#PATH1) Value(#DCBX_1.Path)
Set Com(#PATHTYPE1) Value(#DCBX_1.PathType)
Message Msgtxt('View double clicked Left Explorer Component')
Endroutine
Evroutine Handling(#DCBX_2.ItemDoubleClick)
Set Com(#FILENAME1) Value(#DCBX_2.Filename)
Set Com(#PATH1) Value(#DCBX_2.Path)
Set Com(#PATHTYPE1) Value(#DCBX_2.PathType)
Message Msgtxt('Double clicked Right Explorer Component')
Endroutine
Evroutine Handling(#DCBX_1.ItemDoubleClick)
Set Com(#FILENAME1) Value(#DCBX_1.Filename)
Set Com(#PATH1) Value(#DCBX_1.Path)
Set Com(#PATHTYPE1) Value(#DCBX_1.PathType)
Message Msgtxt('Double clicked Left Explorer Component')
Endroutine
End_Com
```

基本リストのメソッド

StartEdit メソッド

SizeToContents メソッド

FindItem メソッド

StartEdit メソッド

StartEditメソッドはプログラム上で現在の項目の編集を開始する方法です。

StartEditを使って、コンポーネント内の現在の項目の編集をプログラムにより開始します。

SizeToContents メソッド

SizeToContentsメソッドを利用して、カラムの幅をコンテンツの最大幅に変更します。（カラム・ヘッダー上でダブル・クリックした時と同じです）

このメソッドはUnlevelledのツリー・ビューにのみ適用されます。

FindItem メソッド

高度なメソッドです。

FindItemメソッドを使って、その参照からツリー内の項目を検索します。ツリー・ビュー項目を効果的に取り出すのが目的です。

また以下も参照してください。

[RelatedReference プロパティ](#) (ツリー項目の場合)

[Result パラメータ](#)

[RelatedReference パラメータ](#)

[StartItem パラメータ](#)

Result パラメータ

ResultパラメータはFindItemメソッドの結果を返します。

RelatedReference パラメータ

RelatedReferenceパラメータを使って、項目の参照を指定します。このパラメータは必須です。

StartItem パラメータ

StartItemパラメータを使用して、任意で検索する最初の項目を指定します。

StartEdit メソッド

StartEditメソッドはプログラム上で現在の項目の編集を開始する方法です。

StartEditを使って、コンポーネント内の現在の項目の編集をプログラムにより開始します。

基本編集

基本編集プロパティ

AutoSelect プロパティ

AutoSelect プロパティ AutoTab プロパティ

HideSelection プロパティ

HideSelection プロパティ

AutoSelect プロパティ

AutoSelectはテキストの自動選択を制御します。

AutoSelectは、ユーザーがタブを押した時に編集ボックスのテキストを選択できるかどうかを制御します。このプロパティにTrueを設定すると、エディット・ボックスのテキストが選択されます。

AutoTab プロパティ

AutoTabはカーソルを次のコントロールに移します。

入力フィールドが埋められ、カーソルを次の入力フィールド（次の TabPosition）に移動させたい時に、AutoTabプロパティをTrueに設定します。

AutoTabの使用方法を確認するには、次のソースをコピーしてフォームに貼り付け、フォームをコンパイルして実行してください。

```
FUNCTION options(*DIRECT)
BEGIN_COM role(*EXTENDS #PRIM_FORM) CAPTION('AutoSelectedItem
DEFINE_COM class(#PRIM_CKBX) name(#CKBX_AUTOTAB) CAPTION
DEFINE_COM class(#EMPNO.Visual) name(#EMPNO) AUTOSELECT(False)
DEFINE_COM class(#PRIM_EDIT) name(#EDIT_1) AUTOSELECT(False)
DEFINE_COM class(#SALARY.Visual) name(#SALARY) AUTOSELECT(False)
DEFINE_COM class(#PRIM_SPDT) name(#SPDT_1) DISPLAYPOSITION(
DEFINE_COM class(#PRIM_LABL) name(#LABL_1) CAPTION('Edit Box:'
DEFINE_COM class(#PRIM_LABL) name(#LABL_2) CAPTION('Spin Edit
DEFINE_COM class(#PRIM_LABL) name(#LABL_3) CAPTION('When Aut
EVTROUTINE handling(#CKBX_AUTOTAB.Click)
IF cond('#CKBX_AUTOTAB.ButtonState = Checked')
SET com(#EMPNO) AUTOTAB(TRUE)
SET com(#EDIT_1) AUTOTAB(TRUE)
SET com(#SALARY) AUTOTAB(TRUE)
SET com(#spdt_1) AUTOTAB(TRUE)
ELSE
SET com(#EMPNO) AUTOTAB(false)
SET com(#EDIT_1) AUTOTAB(false)
SET com(#SALARY) AUTOTAB(false)
SET com(#spdt_1) AUTOTAB(false)
ENDIF
ENDROUTINE
END_COM
```

HideSelection プロパティ

HideSelectionを利用して、フォーカスがない時に、編集ボックスに選択を表示するかどうかを指定します。

このプロパティにTrueを設定すると、編集ボックスが選択されていない時に選択を非表示にします。

MaxLength プロパティ

MaxLengthは編集ボックスに入力できる最大文字数を設定します。

ビジュアル・ピックリスト

ピックリストは、そこから項目を選択できるリストです。

[ビジュアル・ピックリストのプロパティ](#)

ビジュアル・ピックリストのプロパティ

Appearance プロパティ

Columns プロパティ

ComponentVersion プロパティ

Orientation プロパティ

Appearance プロパティ

Appearanceプロパティを使用して、ピックリストをどのように表示するか制御します。これは、次の値を取ることができます：

ボタン・セット

ピックリストは1セットのラジオ・ボタンとして表示されます。

Appearanceの省略値です。

以下はフィールド#SEXのコンポーネント定義で、これはFまたはMの値をとることができます。

```
Begin_Com Role(*EXTENDS #PRIM_OBJT)
Begin_Com Role(*Visual #PRIM_EVPL) Name(#PICKLIST) Defaultvisual(T
End_Com
Begin_Com Role(*picklist) Name(#LIST)
Define_Com Class(#PRIM_PKIT) Name(#ITEM1) Value('M') Caption('Male')
Define_Com Class(#PRIM_PKIT) Name(#ITEM2) Value('F') Caption('Female')
End_Com
End_Com
```

CheckBox

1つのチェックボックスが、ピックリスト項目のキャプションなしに表示されます。 AppearanceがCheckBoxの時は、ピックリスト項目は2つしか使うことができません。 実行時にチェックボックスがチェックされていないと、最初のピックリストの値がフィールドに割り当てられ、チェックボックスがチェックされている時は2番目のピックリストの項目がフィールドに割り当てられます。

次はフィールド#NOTIFYのコンポーネント定義で、これはYまたはNの値をとることができます。

```
Begin_Com Role(*EXTENDS #PRIM_OBJT)
Begin_Com Role(*Visual #PRIM_EVPL) Name(#PICKLIST) Appearance(Ch
End_Com
Begin_Com Role(*picklist) Name(#LIST)
Define_Com Class(#PRIM_PKIT) Name(#ITEM1) Value('Y') Parent(#LIST) I
Define_Com Class(#PRIM_PKIT) Name(#ITEM2) Value('N') Parent(#LIST)
End_Com
End_Com
```

DropDownList

ピックリストがドロップダウンとして表示されます。

Listbox

ピックリストがリスト・ボックスとして表示されます。

Image

ピックリストがイメージとして表示されます。

ImageAndText

ピックリストがテキストのキャプション付きのイメージとして表示されます。

Columns プロパティ

Columns プロパティを使って、ピックリストのカラム数を指定します。

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 11.0 以前のバージョンです。

1 はLANSA 11.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

ComponentVersionが1 の時、ShowSelectionとShowSelectionHilight プロパティの値はAppearanceがListboxのものに適用されます。

Orientation プロパティ

Orientationはピックリスト内の項目の方向を制御します。

AppearanceがButtonSetに設定されており、リストに複数のカラムがある場合に、Orientationプロパティを使用して、ピックリストの項目の方向を制御します。

ピックリストの方向設定の効果を示すには、カラム数が1以上に設定されている必要があります。

リスト・ボックス

リスト・ボックスは、基本的なリストです。

リスト・ボックスを使用して、ユーザーが1つまたは複数の項目を選択できる項目リストを表示します。項目数が表示できる範囲を超えると、スクロールバーがリストボックスに自動的に追加されます。

リスト・ボックスは最も基本的なリスト・コントロールです。異なる表示スタイル、アイコン、ソート、カラム・ヘッダーなどの機能を利用するには、リスト・ビュー・コントロールを選択します。リスト内の階層データを表示するには、ツリー・ビューを使用します。

リストに表示するデータを定義するには、リポジトリ・ブラウザのフィールドまたはファイルのタブから必要なフィールドをドラッグすることから始めます。リストにドラッグされたフィールドは全てリスト・カラムを作成します。このリスト・カラムには独自のプロパティがあり、修正することもできます。

リストを定義した後、取り出すデータを指定し、SELECTやADD_ENTRYステートメントを使ってリストに追加します。

リストボックスのプロパティ

リストボックスのプロパティ

Cell プロパティ

ComponentVersion プロパティ

Cell プロパティ

Cellプロパティを使用して、リストボックスのセルの処理を行います。
個別のセルやカラムにイメージを割り当てることができます。

パラメータ

Row パラメータ

Column パラメータ

Row パラメータ

セルのRowプロパティは、セルの行の識別をします。

Column パラメータ

セルのColumnプロパティは、セルの列の識別をします。

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 10.0 以前のバージョンです。

1 はLANSA 10.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

ComponentVersionが1の時、DLT_ENTRY コマンドはItemGotFocus、ItemLostFocus、ItemGotSelection と ItemLostSelection イベントを起動します。

リストボックス・セル

リストボックス・セルを使って、リストボックスの個別のセルの処理を行います。

リストボックス・セルのプロパティ

リストボックス・セルのプロパティ

Column プロパティ

Item プロパティ

ListBox プロパティ

IntegralHeight プロパティ

Column プロパティ

セルのColumnプロパティは、セルの列の識別をします。

Item プロパティ

Itemプロパティはセルに対応するリストボックスの項目コンポーネントを返します。

ListBox プロパティ

ListBoxプロパティはこのセルが属しているリストボックスの名前を返します。

IntegralHeight プロパティ

IntegralHeightはリストの最後の項目がどのように表示されるかを指定します。

リストに項目の一部分を表示するかどうかをIntegralHeightにより制御します。このプロパティがTrueに設定されていると、行の一部分が表示されることはありません。

リスト・ビュー

ユーザーにさまざまな方法でリストを表示するには、リスト・ビューを使用します。カラム・ヘディング付きまたはなしで列に項目を整列したり、アイコンやテキストと共に表示することもできます。

リストに表示するデータを定義するには、リボジトリ・ブラウザのフィールドまたはファイルのタブから必要なフィールドをドラッグすることから始めます。リストにドラッグされたフィールドは全てリスト・カラムを作成します。このリスト・カラムには独自のプロパティがあり、修正することもできます。

リストを定義した後、取り出すデータを指定し、SELECTやADD_ENTRYステートメントを使ってリストに追加します。

リスト・ビューにはViewStyleプロパティがあり、リストをどのように表示するかを指定します。リスト・ビュー項目のアイコンは、setコマンドによって割り当てられます。

```
set com(#ltvw_1.currentitem) image(#myicon)
```

リスト・ビューのエントリーをどのように位置づけるかの例については、SET例 162を参照してください：[リスト・ビュー内のエントリーの位置付け](#)

[リスト・ビューのプロパティ](#)

[リスト・ビューのメソッド](#)

リスト・ビューのプロパティ

AutoArrange プロパティ

Checkboxes プロパティ

ColumnHeaderPress プロパティ

ComponentVersion プロパティ

IconAlignment プロパティ

ValueAt

AutoArrange プロパティ

AutoArrangeは自動的にアイコンを整理します。

リスト・ビューのIconやSmallIconビューのアイコンを自動で整理するかどうかを決定します。 IconビューとSmallIconビューにのみ有効です。

Checkboxes プロパティ

Checkboxesはリスト・ビューの項目に対して表示することができます。Checkboxesプロパティを使用して、リスト・ビューの項目にチェック・ボックスを表示します。このプロパティがTrueに設定されていると、チェック・ボックスが表示されます。

ユーザーがクリックする、またはその項目のCheckedプロパティがプログラム上で設定されると、チェック・ボックスが選択されます。リスト・ビューの項目が選択されているかどうかを調べるには、項目のCheckedプロパティを使います。

CheckboxesプロパティがTrueで、ComponentVersionが0に設定されている時は、項目のImageStateプロパティを使用して指定されたイメージは無視されることに注意してください。

チェックボックスをクリックすることで、リスト・ビューのItemChangedイベントが起動されます。

このプロパティがどのように使用されるかを確認するには、次のコードをフォームにコピー・貼り付けてください。最初のリストの選択されたチェック・ボックスの項目は2番目のリストに追加されます。チェック・ボックスがクリアされると、項目は2番目のリストから取り除かれます。

```
Function Options(*DIRECT)
BEGIN_COM HEIGHT(331) LEFT(302) TOP(108) WIDTH(535)
DEFINE_COM CLASS(#PRIM_LTVW) NAME(#LTVW_1) CHECKBOXES
DEFINE_COM CLASS(#PRIM_LTVW) NAME(#LTVW_2) DISPLAYPOSITION
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_1) DISPLAYPOSITION
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_2) DISPLAYPOSITION
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_3) DISPLAYPOSITION
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_4) DISPLAYPOSITION
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_5) DISPLAYPOSITION
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_6) DISPLAYPOSITION
Evtroutine Handling(#com_owner.CreateInstance)
Select Fields(#LTVW_1) From_File(PSLMST)
Add_Entry To_List(#LTVW_1)
Endselect
Endroutine
Evtroutine Handling(#LTVW_1.ItemChanged) Options(*NOCLEARMESSAG
If Cond('#ltvw_1.currentitem.checked = true')
```

```
Add_Entry To_List(#LTVW_2)
Else
Invoke Method(#com_owner.remove_item) Employee(#empno)
Endif
Endroutine
Mthroutine Name(Remove_item)
Define_Map For(*input) Class(#empno) Name(#employee)
Selectlist Named(#LTVW_2)
Continue If('#empno *ne #employee.value')
Dlt_Entry From_List(#LTVW_2)
Leave
Endselect
Endroutine
End_Com
```

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 10.0 以前のバージョンです。

1 はLANSA 10.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

2 はLANSA 11.0 以降のバージョンです。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

ComponentVersionが1 の場合、VisualStyleで定義されたリスト・ビューのAlternateの色が表示され、同時にState ImagesやCheckBoxesも表示することができます。

ComponentVersionが2 の時、リスト・ビューのカラム・ヘディングのキャプションが複数行として表示されます。

IconAlignment プロパティ

IconAlignmentはアイコンがどのように整列されるかを指定します。

IconAlignmentプロパティを使用して、アイコンをどのように整理するかを指定します。

ValueAt

リスト・ビューの項目にRow（行）とColumn（列）の指標を使ってアクセスします。

RESULT

リスト要素のバリエーション値です。

ROW

アクセスする行の整数インデックス

COLUMN

アクセスする列の整数インデックス

例

以下は指定された場所の値を返します。

```
#ListView ValueAt<#RowIndex #ColIdx>
```

ColumnHeaderPress プロパティ

ColumnHeaderPress プロパティを使って、カラム・ヘッダーがクリックに
応答するかどうかを制御します。

リスト・ビューのメソッド

SetValue メソッド

SetValue メソッド

リスト・ビューの指定の行と列にある値を設定します。

RESULT

設定が成功(TRUE)したかどうかを示すブール値です。

ROW

設定する行の整数インデックス

COLUMN

設定する列の整数インデックス

VALUE

設定する値

例

```
#ListView SetValue<#RowIndex #ColIdx "newValue">
```

項目ヒント・テキストのパラメータ

[ItemHintText](#) イベント [Caption](#) パラメータ

[ItemHintText](#) イベント [Cell](#) パラメータ

ItemHintTextイベント Caption パラメータ

ItemHintTextイベント Caption パラメータ

ItemHintTextイベントのCaptionパラメータを使用して、ヒントのテキストを割り当てます。

コンポーネントのShowItemHintプロパティがTrueに設定され、ヒントが表示されることを確認してください。

次の例は、サンプルPSLMTファイルが基本になっています。フォームにはファーストネームを表示する1列のリストビューがあります。項目のヒントは苗字で、これは非表示の列としてリストに含まれています。

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM HEIGHT(292) LEFT(372) TOP(130) WIDTH(216)
DEFINE_COM CLASS(#PRIM_LTVW) NAME(#LTVW_1) DISPLAYPOSITION(1)
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_1) DISPLAYPOSITION(1)
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_2) PARENT(#LTVW_1)
EVTROUTINE handling(#com_owner.Initialize)
select fields(#ltvw_1) from_file(pslmst)
add_entry #ltvw_1
endselect
ENDROUTINE
EVTROUTINE HANDLING(#LTVW_1.ItemHintText) Caption(#thename)
set #thename value(#surname)
ENDROUTINE
END_COM
```

ItemHintText イベント Cell パラメータ

ItemHintText イベントの Cell パラメータを使って、ヒントが表示されるグリッド・セルを指定します。

次の例は、サンプル PSLMT ファイルが基本になっています。グリッド項目に表示されるヒントのタイプはグリッド・セルの列によって決定されます。

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Busyupdates(Immediate) Client
Define_Com Class(#PRIM_GRID) Name(#GRID) Captionnoblanklines(True)
Define_Com Class(#PRIM_GDCL) Name(#GDCL_1) Displayposition(1) Parent
Define_Com Class(#PRIM_GDCL) Name(#GDCL_2) Displayposition(2) Parent
Define_Com Class(#PRIM_CKBX) Name(#CKBX_HINTS) Buttonstate(Checked)
EvtRoutine Handling(#com_owner.Initialize)
Change Field(#STD_TEXTS) To('Col 1')
Change Field(#STD_TEXT) To('Col 2')
Select Fields(#GRID) From_File(PSLMST)
Add_Entry To_List(#GRID)
Endselect
Endroutine
EvtRoutine Handling(#GRID.ItemHintText) Options(*NOCLEARMESSAGES)
If_Ref Com(#CELL.Column) Is(*EQUAL_TO #GDCL_1)
Set Com(#thename) Value(#EMPNO)
Else
If_Ref Com(#CELL.Column) Is(*EQUAL_TO #GDCL_2)
Set Com(#thename) Value(#address1)
Else
Set Com(#thename) Value('Unknown')
Endif
Endif
Endroutine
EvtRoutine Handling(#CKBX_HINTS.Click)
If Cond('#CKBX_HINTS.buttonstate *eq unchecked')
Set Com(#GRID) Showitemhint(false)
Else
Set Com(#GRID) Showitemhint(true)
Endif
Endroutine
```

End_Com

グリッド

グリッドはデータをテーブル形式で表示します。ユーザーは任意でグリッドの情報を編集できます。

グリッドに表示するデータを定義するには、リポジトリ・ブラウザのフィールドまたはファイルのタブから必要なフィールドをドラッグすることから始めます。グリッドにドラッグされたフィールドは全てグリッド・カラムを作成します。このグリッド・カラムには独自のプロパティがあり、これを修正できます。

グリッドを定義した後、取り出すデータを指定し、SELECTやADD_ENTRYステートメントを使ってグリッドに追加します。

グリッド・カラムのソート方法の例については、SET例35を参照してください。 [グリッド列をソートする](#)

特定のセルにフォーカスを設定する方法についての例はSET例37を参照してください。 [グリッド内の特定のセルにフォーカスを設定する](#)

SET例199は再利用可能なポップアップ・メニューにより、ユーザーが実行時にグリッドの表示をカスタマイズできます。詳細は次を確認してください。 [動的にグリッドをフォーマットする](#)

[グリッドのプロパティ](#)

[グリッドのイベント](#)

[グリッドのメソッド](#)

グリッドのプロパティ

AnchorCell プロパティ	DragColumns プロパティ	RowSizing プロパティ
CaptionNoBlankLines プロパティ	EnterKeyStyle プロパティ	SelectedColumnCount プロパティ
Cell プロパティ	EscapeKeyStyle プロパティ	SelectedRowCount プロパティ
ColumnButtons プロパティ	FocusCell プロパティ	ShowButtonSelection プロパティ
ColumnButtonHeight プロパティ	FrozenColumns プロパティ	ShowLines プロパティ
ColumnButtonPress プロパティ	RowButtons プロパティ	TabbingStyle プロパティ
ColumnEllipses プロパティ	RowButtonWidth プロパティ	SplitStyle プロパティ
ColumnResize プロパティ	RowHeight プロパティ	ValueAt プロパティ
ComponentVersion プロパティ	RowResize プロパティ	

AnchorCell プロパティ

AnchorCellは選択が始まるセルです。

AnchorCellとはグリッド内のセルで、ここに選択のアンカーが置かれます。(選択の開始位置)

AnchorCellはSelectionStyleがMultipleまたはExtendedの時に使用されます。いったん選択の操作が開始されると、アンカーのセルは変更できません。

FocusCellとAnchorCellがどのようなものかを確認するには、まずSelectionStyleがExtendedで10個の項目を持つグリッドを作成してください。セル(1,1)をクリックし、Shiftキーを押しながらセル(5,5)をクリックします。この時、FocusCellはセル(5,5)、AnchorCellはセル(1,1)です。

ここでセル(7,7)をクリックすると、これがFocusCellになりますが、AnchorCellはセル(1,1)のままです。

[Cell プロパティ](#)の例も参照してください。

CaptionNoBlankLines プロパティ

CaptionNoBlankLinesは、カラム・ヘッダーの空白行の表示を抑制します。

Trueに設定されると、CaptionNoBlankLinesはグリッドのカラム・ヘッダーの無意味な空白行を抑制します。

Cell プロパティ

Cellプロパティはグリッドのセルです。このプロパティにはRow(行)とColumn(列)の2つのキーがあり、これによりセルが識別されます。

セルのプロパティはグリッド・セル・コンポーネントのインスタンスに相当します。セル・コンポーネントにどのようなプロパティがあるかを確認するには、左側のCellプロパティでダブルクリックしてください。

グリッド内の特定のセルをどのように処理するかを見るには、次の例をコピーしてフォームに貼り付け、そのフォームをコンパイルして実行してください。

```
function options(*DIRECT)
```

```
begin_com role(*EXTENDS #PRIM_FORM) clientheight(389) clientwidth(42)
define_com class(#PRIM_GRID) name(#GRID_1) columnbuttonheight(18) cc
define_com class(#PRIM_GDCL) name(#GDCL_1) captionalign(Left) display
define_com class(#PRIM_GDCL) name(#GDCL_2) captionalign(Left) display
define_com class(#PRIM_GPBX) name(#GPBX_13) caption('Style') displayp
define_com class(#PRIM_GPBX) name(#GPBX_12) displayposition(1) heigh
define_com class(#PRIM_RDBN) name(#RDBN_8) caption('Normal') display
define_com class(#PRIM_RDBN) name(#RDBN_9) caption('Warning') displa
define_com class(#PRIM_RDBN) name(#RDBN_14) caption('Large M') displ
define_com class(#PRIM_RDBN) name(#RDBN_10) caption('Large') display
define_com class(#PRIM_RDBN) name(#RDBN_13) caption('Emph') display
define_com class(#PRIM_RDBN) name(#RDBN_11) caption('Smth') displayp
define_com class(#PRIM_RDBN) name(#RDBN_12) caption('Title') displayp
define_com class(#PRIM_SPDT) name(#SPDT_ROW) displayposition(1) heig
define_com class(#PRIM_SPDT) name(#SPDT_COL) displayposition(2) heig
define_com class(#PRIM_LABL) name(#LABL_1) caption('Row') displaypos
define_com class(#PRIM_LABL) name(#LABL_2) caption('Column') display
define_com class(#PRIM_RDBN) name(#RDBN_FOCUS) buttonchecked(Tru
define_com class(#PRIM_RDBN) name(#RDBN_ANCHOR) caption('Anchor
define_com class(#PRIM_RDBN) name(#RDBN_CELL) caption('Cell') displa
define_com class(#PRIM_RDBN) name(#RDBN_CITEM) caption('Current It
define_com class(#STD_NUM) name(#STD_NUM)
evtroutine handling(#COM_OWNER.Initialize) options(*NOCLEARMESSA(
select fields(#GRID_1) from_file(PSLMST)
add_entry to_list(#GRID_1)
endselect
```

```

endroutine
evtroutine handling(#RDBN_8.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#vs_norm)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#vs_norm)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#vs_norm)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_9.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#vs_WARN)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#vs_WARN)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#vs_WARN)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_10.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#vs_Large)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#VS_LARGE)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#vs_Large)
endif

```

```
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_11.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#VS_SMTH)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#vs_smth)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#vs_smth)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_12.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#VS_TITLE)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#VS_TITLE)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#VS_TITLE)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_13.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#VS_EMPH)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#VS_EMPH)
endif
```

```
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#VS_EMPH)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_14.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#VS_larem)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#VS_larem)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#VS_larem)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine

end_com
```

また以下も参照してください。

[CellのRow プロパティ](#)

[CellのCollumn プロパティ](#)

CellのRow プロパティ

セルのRowプロパティは、セルの行の識別をします。

CellのColumn プロパティ

セルのColumnプロパティは、セルの列の識別をします。

ColumnButtons プロパティ

ColumnButtonsはカラム・ヘッダー・ボタンを表示/非表示にします。このプロパティにFalseを設定すると、カラム・ヘッダー・ボタンなしのグリッドまたはプロパティ・シートが作成されます。

ColumnButtonHeight プロパティ

ColumnButtonHeightはカラム・ヘッダー・ボタンの高さを設定します。

ColumnButtonPress プロパティ

ColumnButtonPressプロパティを使用して、カラム・ヘッダー・ボタンを有効・無効にします。このプロパティがTrueに設定されると、カラム・ヘッダー・ボタンは無効になります。

ColumnEllipses プロパティ

ColumnEllipsesはカラム・ヘッダーに省略記号を追加します。

ColumnEllipsesプロパティを使用して、サイズ調整されたためにカラム・ヘッダーが一部しか表示されない場合に、省略記号（3つの点）を追加するかどうかを指定します。

ColumnResize プロパティ

ColumnResizeはカラムがサイズ変更可能かどうかを制御します。このプロパティにTrueを設定すると、カラムのサイズ変更ができるようになります。

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 10.0 以前のバージョンです。

1 はLANSA 10.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

ComponentVersionが1の時、DLT_ENTRY コマンドはItemGotFocus、ItemLostFocus、ItemGotSelection と ItemLostSelection イベントを起動します。

DragColumns プロパティ

DragColumnsはカラムのドラッグ・アンド・ドロップを制御します。

DragColumnsはドラッグ・アンド・ドロップによってカラムを再編成できるかどうかを制御します。

EnterKeyStyle プロパティ

EnterKeyStyleは実行時のプロパティで、グリッドでEnterキーが押されるとどうなるかを決定します。

EnterKeyStyleを使って、グリッド内でEnterキーが押された時に何が起きるかを決定します。

省略値(Down)では、グリッドにフォーカスがある時にEnterを押すと、行が下に移動します。

代わりにEnterキーに次のような値を設定することもできます：

- フォーカスを次の行の最初の列に移動します。(NextRow)
- 現在行のセルに沿ってフォーカスを移動後、次の行に下り、一番最後のセルまで行くと、再び最初の行に戻ります。(AroundGrid)
- 現在行のセルに沿ってフォーカスを移動後、次の行に下り、一番最後のセルまで行くと、グリッドから出ます。(ThroughGrid)
- フォームの省略値のボタンを起動します。省略値のボタンがない場合、グリッド内の行に移動します。(DialogDown)
- フォームの省略値のボタンを起動します。省略値のボタンがない場合、フォーカスは次の行の最初の行に移動します。
- フォームの省略値のボタンを起動します。省略値のボタンがない場合、フォーカスはAroundGridで定義されたように移動します。
- フォームの省略値のボタンを起動します。省略値のボタンがない場合、フォーカスはThroughGridで定義されたように移動します。

EscapeKeyStyle プロパティ

EscapeKeyStyleは、グリッドでEscキーが押された時にどうするかを決定します。

実行時のプロパティです。

EscapeKeyStyleを使って、グリッド内でEscキーが押された時に何が起きるかを決定します。

省略値では、グリッド・セルにフォーカスがある時にEscキーを押すと、値は割り当てられた値に戻されます。(CancelEdit)

または、ButtonCancelにTrueが設定されたフォーム上のボタンを起動するよう、Escキーに設定することもできます。(DialogCancelEdit) この値の場合、ButtonCancelにTrueが設定されたボタンが存在しない時にEscキーを押すと、セルには割り当てられた値がリセットされます。

FocusCell プロパティ

FocusCellは選択が始まるセルです。

FocusCellはフォーカスのあるグリッド内のセルです。

FocusCellとAnchorCellがどのようなものかを確認するには、まず SelectionStyleがExtendedで10個の項目を持つグリッドを作成してください。セル(1, 1)をクリックし、Shiftキーを押しながらセル(5, 5)をクリックします。この時、FocusCellはセル(5, 5)、AnchorCellはセル(1, 1)です。

ここでセル(7,7)をクリックすると、これがFocusCellになりますが、AnchorCellはセル(1, 1)のままです。

[Cell プロパティ](#)の例も参照してください。

FrozenColumns プロパティ

FrozenColumnsは動かさない列の数を設定します。

固定カラム(Frozen column)は、グリッド内の別の列や行がスクロールされても動きません。

固定カラムは、通常スプレッドシート・アプリケーション内で常に表示すべき欄に使用されます。

特に一番左のカラムを水平スクロールの一部に含みたくない場合は、そのカラム数をこのプロパティに指定します。

RowButtons プロパティ

RowButtonsは行の一番最初の列をグリッドのボタンにします。

RowButtonWidth プロパティ

RowButtonWidthは行ボタンの幅をグリッドに設定します。

RowHeight プロパティ

RowHeightは行の高さをピクセルで設定します。

RowResize プロパティ

RowResizeはグリッドやプロパティ・シートの行がサイズ変更できるかどうかを制御します。このプロパティにTrueを設定すると、行のサイズ変更ができるようになります。

RowSizing プロパティ

RowSizing プロパティを使用して、行がどのようにサイズ調整されるかを指定できます。このプロパティは、RowResize プロパティが True に設定されている時のみ有効になります。

ひとまとまりの行のサイズ変更を可能にするには、Group に設定します。

行のサイズ変更を各個別の行の高さに適用させるには、Individual に設定します。

コンテンツのサイズを調整するには、ContentHeight を使用します。

このプロパティの使用方法を確認するには、次のソースをコピーしてフォームに貼り付け、フォームをコンパイルして実行してください。

```
FUNCTION options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CLIENTHEIGHT(226) C
DEFINE_COM CLASS(#PRIM_GRID) NAME(#GRID_1) COLUMNBUTTC
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_1) CAPTIONALIG
DEFINE_COM CLASS(#PRIM_GDCL) NAME(#GDCL_2) CAPTIONALIG
DEFINE_COM CLASS(#PRIM_GPBX) NAME(#GPBX_1) CAPTION('Row
DEFINE_COM CLASS(#PRIM_RDBN) NAME(#RDBN_2) BUTTONCHEC
DEFINE_COM CLASS(#PRIM_RDBN) NAME(#RDBN_1) CAPTION('Indi
DEFINE_COM CLASS(#PRIM_RDBN) NAME(#RDBN_3) CAPTION('Con
EVTROUTINE handling(#COM_OWNER.Initialize) options(*NOCLEARME
SELECT fields(#GRID_1) from_file(PSLMST)
ADD_ENTRY to_list(#GRID_1)
ENDSELECT
ENDROUTINE
EVTROUTINE handling(#RDBN_1.Click)
IF cond('#RDBN_1.buttonchecked *eq true')
SET com(#grid_1) ROWSIZING(Individual)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_2.Click)
IF cond('#RDBN_2.buttonchecked *eq true')
SET com(#grid_1) ROWSIZING(group)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_3.Click)
```

```
IF cond('#RDBN_3.buttonchecked *eq true')
SET com(#grid_1) ROWSIZING(ContentHeight)
ENDIF
ENDROUTINE
END_COM
```

SelectedColumnCount プロパティ

SelectedColumnCountは選択された列の数を示します。

SelectedColumnCountプロパティを使って、現在選択されている列の数を取り出します。

このプロパティの使用方法を確認するには、次のソースをコピーしてフォームに貼り付け、フォームをコンパイルして実行してください。

```
FUNCTION options(*DIRECT)
BEGIN_COM role(*EXTENDS #PRIM_FORM) HEIGHT(330) LEFT(285) T
DEFINE_COM class(#PRIM_GRID) name(#GRID_1) COLUMNBUTTONH
DEFINE_COM class(#PRIM_GDCL) name(#GDCL_1) CAPTIONALIGN(Lc
DEFINE_COM class(#PRIM_GDCL) name(#GDCL_2) CAPTIONALIGN(Lc
DEFINE_COM class(#STD_NUM.Visual) name(#STD_NUM) CAPTION('Se
DEFINE_COM class(#STD_NUM.Visual) name(#STD_NUM_1) CAPTION(
DEFINE_COM class(#PRIM_GPBX) name(#GPBX_1) CAPTION('Selection
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_1) BUTTONCHECKED
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_2) CAPTION('Multiple'
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_3) CAPTION('Extended
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_4) CAPTION('Whole R
EVTROUTINE handling(#COM_OWNER.Initialize) options(*NOCLEARME
SELECT fields(#GRID_1) from_file(PSLMST)
ADD_ENTRY to_list(#GRID_1)
ENDSELECT
ENDROUTINE
EVTROUTINE handling(#GRID_1.Changed) options(*NOCLEARMESSAGI
CHANGE field(#STD_NUM) to('#GRID_1.SelectedROWCount')
CHANGE field(#STD_NUM_1) to('#GRID_1.SelectedColumnCount')
ENDROUTINE
EVTROUTINE handling(#RDBN_1.Click)
IF cond('#RDBN_1.buttonchecked *eq true')
SET com(#grid_1) SELECTIONSTYLE(single)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_2.Click)
IF cond('#RDBN_2.buttonchecked *eq true')
SET com(#grid_1) SELECTIONSTYLE(multiple)
ENDIF
ENDROUTINE
```

```
EVTROUTINE handling(#RDBN_3.Click)
IF cond('#RDBN_3.buttonchecked *eq true')
SET com(#grid_1) SELECTIONSTYLE(extended)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_4.Click)
IF cond('#RDBN_4.buttonchecked *eq true')
SET com(#grid_1) SELECTIONSTYLE(WholeRow)
ENDIF
ENDROUTINE
END_COM
```

SelectedRowCount プロパティ

SelectedRowCountは選択された行の数を示します。

SelectedColumnCount プロパティを使って、現在選択されている行の数を取り出します。

このプロパティの使用方法を確認するには、次のソースをコピーしてフォームに貼り付け、フォームをコンパイルして実行してください。

```
FUNCTION options(*DIRECT)
BEGIN_COM role(*EXTENDS #PRIM_FORM) HEIGHT(330) LEFT(285) T
DEFINE_COM class(#PRIM_GRID) name(#GRID_1) COLUMNBUTTONH
DEFINE_COM class(#PRIM_GDCL) name(#GDCL_1) CAPTIONALIGN(Lc
DEFINE_COM class(#PRIM_GDCL) name(#GDCL_2) CAPTIONALIGN(Lc
DEFINE_COM class(#STD_NUM.Visual) name(#STD_NUM) CAPTION('Se
DEFINE_COM class(#PRIM_GPBX) name(#GPBX_1) CAPTION('Selection
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_1) BUTTONCHECKED
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_2) CAPTION('Multiple'
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_3) CAPTION('Extended
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_4) CAPTION('Whole R
EVTROUTINE handling(#COM_OWNER.Initialize) options(*NOCLEARME
SELECT fields(#GRID_1) from_file(PSLMST)
ADD_ENTRY to_list(#GRID_1)
ENDSELECT
ENDROUTINE
EVTROUTINE handling(#GRID_1.Changed) options(*NOCLEARMESSAGI
CHANGE field(#STD_NUM) to('#GRID_1.SelectedRowCount')
ENDROUTINE
EVTROUTINE handling(#RDBN_1.Click)
IF cond('#RDBN_1.buttonchecked *eq true')
SET com(#grid_1) SELECTIONSTYLE(single)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_2.Click)
IF cond('#RDBN_2.buttonchecked *eq true')
SET com(#grid_1) SELECTIONSTYLE(multiple)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_3.Click)
IF cond('#RDBN_3.buttonchecked *eq true')
```

```
SET com(#grid_1) SELECTIONSTYLE(extended)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_4.Click)
IF cond('#RDBN_4.buttonchecked *eq true')
SET com(#grid_1) SELECTIONSTYLE(WholeRow)
ENDIF
ENDROUTINE
END_COM
```

ShowButtonSelection プロパティ

ShowButtonSelectionはカラム・ヘッダー・ボタンの外観を制御します。カラム内の項目が選択された時に、そのカラム・ヘッダー・ボタンのキャプションが強調表示されるようにするには、ShowButtonSelectionをTrueに設定します。

このプロパティの使用方法を確認するには、次のソースをコピーしてフォームに貼り付け、フォームをコンパイルして実行してください。

```
FUNCTION options(*DIRECT)
BEGIN_COM role(*EXTENDS #PRIM_FORM) HEIGHT(258) LEFT(336) T
DEFINE_COM class(#PRIM_GRID) name(#GRID_1) COLUMNBUTTONH
DEFINE_COM class(#PRIM_GDCL) name(#GDCL_1) CAPTIONALIGN(Lc
DEFINE_COM class(#PRIM_GDCL) name(#GDCL_2) CAPTIONALIGN(Lc
DEFINE_COM class(#PRIM_GPBX) name(#GPBX_1) CAPTION('ShowButt
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_2) BUTTONCHECKED
DEFINE_COM class(#PRIM_RDBN) name(#RDBN_1) CAPTION('True') DI
EVTROUTINE handling(#COM_OWNER.Initialize) options(*NOCLEARME
SELECT fields(#GRID_1) from_file(PSLMST)
ADD_ENTRY to_list(#GRID_1)
ENDSELECT
ENDROUTINE
EVTROUTINE handling(#RDBN_1.Click)
IF cond('#RDBN_1.buttonchecked *eq true')
SET com(#grid_1) SHOWBUTTONSELECTION(True)
ENDIF
ENDROUTINE
EVTROUTINE handling(#RDBN_2.Click)
IF cond('#RDBN_2.buttonchecked *eq true')
SET com(#grid_1) SHOWBUTTONSELECTION(False)
ENDIF
ENDROUTINE
END_COM
```

ShowLines プロパティ

ShowLinesはグリッド線を表示するかどうかを決定します。このプロパティをFalseに設定すると、グリッド線が表示されません。

TabbingStyle プロパティ

TabbingStyleがAlongRowの時にTabキーが押されると、カーソルはあるセルから同じ行の別のセルに移動します。

AroundGridの時にTabキーが押されると、カーソルはあるセルから同じ行の別のセルに移動し、次の行に移動します。

LeaveGridの時にTabキーが押されると、カーソルはグリッドから移動します。

ThroughGridの時は、右下または左上のセルにフォーカスがある時以外は、AroundGridと同様にカーソルが移動します。右下のセルにフォーカスがある時にTabキーが押される、もしくは左上のセルにフォーカスがある時にバックTabキーが押されると、ThroughGridはLeaveGridと同様の動きをします。

SplitStyle プロパティ

SplitStyleプロパティを使用して、個別のペインがスクロールできるようにします（Excelと似た表示方法）こうすることで、グリッドのユーザーはグリッドの異なる部分を一度に見ることができます。

通常分割ウィンドウは、縦の分割にはユーザーがグリッドの右上の分割ハンドルを、横の分割には左下の分割ハンドルを使って開始します。

SplitStyleでは、グリッドを2つの横並びのペインに分割したり、2つの縦並びのペインや4つのペイン（両方）に分割することができます。

グリッドが縦に分割された場合、ユーザーはペインの下部のデータにスクロールする際、特定数の行を常にペインの上に表示することができます。

ValueAt プロパティ

指定した行と列の値を取得します。

RESULT

リスト要素のバリエーション値です。

ROW

アクセスする行の整数インデックス

COLUMN

アクセスする列の整数インデックス

例

```
Begin_Loop Using(#RowIndex) To(3)
  Begin_Loop Using(#ColIdx) To(#Grid1.columns.itemcount)
    #StringVal := #Grid1.ValueAt<#RowIndex #ColIdx>
    * do something useful with #StringVal ...
  End_Loop
End_Loop
```

グリッドのイベント

EditorChanged イベント

EditorChanged イベント

EditorChangedイベントはカラムのエディターで変更された時に起動します。例えば、編集ボックスのエディターでは、各キー入力ごとにこのイベントが起動します。

グリッドのメソッド

SetValue メソッド

StartSplit メソッド

SetValue メソッド

指定された場所にグリッド項目の値を設定します。

RESULT

設定が成功(TRUE)したかどうかを示すブール値です。

ROW

設定する行の整数インデックス

COLUMN

設定する列の整数インデックス

VALUE

設定する値

例

```
#Grid1.SetValue ( 1 1 "11" )
```

StartSplit メソッド

StartSplitは個別のペインのスクロールを可能にします。

コンポーネントは複数のペインとして表示することができます。

(Excelと似た表示方法) こうすることで、コンポーネントのユーザーはコンポーネントの異なる部分を一度に見ることができます。

コンポーネントが縦に分割された場合、ユーザーはペインの下部のデータにスクロールする際、特定数の行を常にペインの上に表示することができます。

SplitStyleプロパティはコンポーネントで使用可能な分割動作のタイプを管理します。

通常分割ウィンドウは、縦の分割にはユーザーがコンポーネントの右上の分割ハンドルを、横の分割には左下の分割ハンドルを使って開始します。StartSplitはプログラム上でこの効果をシミュレーションする方法です。ユーザーからの入力なしに分割操作を開始する場合に使用します。

グリッド・セル

グリッド・セルのコンポーネントを使って、グリッド内の個別のセルの処理を行います。

グリッド・セルのプロパティ

グリッド・セルのメソッド

グリッド・セルのプロパティ

Column プロパティ

Item プロパティ

Grid プロパティ

CellReadOnly プロパティ

Column プロパティ

Columnプロパティを使って、このセルが属するグリッド欄を返します。

Item プロパティ

Itemプロパティを使用して、セルに対応するグリッド項目コンポーネントを返します。

Grid プロパティ

Gridプロパティを使って、このセルが属するグリッドを返します。

CellReadOnly プロパティ

CellReadOnly プロパティは下位互換性を保つためだけのものです。代わりにセル・コンポーネントのReadOnly プロパティを使用するようにしてください。

グリッド・セルのメソッド

SetValue メソッド

SetValue メソッド

このメソッドを使用して、グリッド・セルの値を設定します。

RESULT

設定が成功(TRUE)したかどうかを示すブール値です。

VALUE

設定する値

例

```
#Grid1.Cell<1 1>.SetValue ( "8" )
```

```
#Grid1.Cell<1 2>.SetValue ( "9" )
```

```
#Grid1.Cell<2 1>.SetValue ( "1" )
```

```
#Grid1.Cell<2 2>.SetValue ( "2" )
```

GridItem

[GridItemのプロパティ](#)

[GridItemのメソッド](#)

GridItemのプロパティ

ValueAt プロパティ

Cell プロパティ

Column パラメータ

ValueAt プロパティ

グリッド・セルの値にアクセスします。

RESULT

リスト要素のバリエーション値です。

ROW

アクセスする行の整数インデックス

例

```
#Value := #Grid1.ValueAt<#RowIndex #ColIdx>
```

Cell プロパティ

Cell プロパティはグリッドのセルです。

グリッド内の特定のセルをどのように処理するかを見るには、次の例をコピーしてフォームに貼り付け、そのフォームをコンパイルして実行してください。

```
function options(*DIRECT)
```

```
begin_com role(*EXTENDS #PRIM_FORM) clientheight(389) clientwidth(42)
define_com class(#PRIM_GRID) name(#GRID_1) columnbuttonheight(18) columnbuttonwidth(18)
define_com class(#PRIM_GDCL) name(#GDCL_1) captionalign(Left) displayposition(1) height(18) width(18)
define_com class(#PRIM_GDCL) name(#GDCL_2) captionalign(Left) displayposition(1) height(18) width(18)
define_com class(#PRIM_GPBX) name(#GPBX_13) caption('Style') displayposition(1) height(18) width(18)
define_com class(#PRIM_GPBX) name(#GPBX_12) displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_8) caption('Normal') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_9) caption('Warning') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_14) caption('Large M') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_10) caption('Large') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_13) caption('Emph') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_11) caption('Smth') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_12) caption('Title') displayposition(1) height(18) width(18)
define_com class(#PRIM_SPDT) name(#SPDT_ROW) displayposition(1) height(18) width(18)
define_com class(#PRIM_SPDT) name(#SPDT_COL) displayposition(2) height(18) width(18)
define_com class(#PRIM_LABL) name(#LABL_1) caption('Row') displayposition(1) height(18) width(18)
define_com class(#PRIM_LABL) name(#LABL_2) caption('Column') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_FOCUS) buttonchecked(True) displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_ANCHOR) caption('Anchor') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_CELL) caption('Cell') displayposition(1) height(18) width(18)
define_com class(#PRIM_RDBN) name(#RDBN_CITEM) caption('Current Item') displayposition(1) height(18) width(18)
define_com class(#STD_NUM) name(#STD_NUM)
evtroutine handling(#COM_OWNER.Initialize) options(*NOCLEARMESSAGE)
select fields(#GRID_1) from_file(PSLMST)
add_entry to_list(#GRID_1)
endselect
endroutine
evtroutine handling(#RDBN_8.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#vs_norm)
```

```

endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#vs_norm)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#vs_norm)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_9.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#vs_WARN)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#vs_WARN)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#vs_WARN)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_10.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#vs_Large)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#VS_LARGE)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#vs_Large)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine

```

```

evtroutine handling(#RDBN_11.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#VS_SMTH)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#vs_smth)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#vs_smth)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_12.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#VS_TITLE)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#VS_TITLE)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#VS_TITLE)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_13.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#VS_EMPH)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#VS_EMPH)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#VS_EMPH)
endif
if cond('#RDBN_cell.Buttonchecked = true')

```

```
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine
evtroutine handling(#RDBN_14.Click)
if cond('#RDBN_CITEM.Buttonchecked = true')
set com(#GRID_1.currentitem) visualstyle(#VS_larem)
endif
if cond('#RDBN_FOCUS.Buttonchecked = true')
set com(#GRID_1.focuscell) visualstyle(#VS_larem)
endif
if cond('#RDBN_anchor.Buttonchecked = true')
set com(#GRID_1.anchorcell) visualstyle(#VS_larem)
endif
if cond('#RDBN_cell.Buttonchecked = true')
set com(#GRID_1.cell<#SPDT_ROW.Value #SPDT_COL.Value>) visualstyle
endif
endroutine

end_com
```

Column パラメータ

Columnパラメータを使って、セルのグリッド欄を指定します。

GridItemのメソッド

SetValue メソッド

SetValue メソッド

グリッド・セルの値を設定します。

RESULT

設定が成功(TRUE)したかどうかを示すブール値です。

ROW

設定する行の整数インデックス

VALUE

設定する値

例

```
#Grid1.Items<2>.SetValue ( 1 "21" )
```

コンボ・ボックス

コンボ・ボックスは狭いスペースに収まるリストを提供し、ユーザーはそこにエントリーを追加することができます。

コンボ・ボックスは、編集ボックスとリスト・ボックスの機能を結合したものです。ユーザーは編集エリアに情報を入力したり、リストから項目を選択したりできます。ユーザーがリストからエントリーを選択すると、そのエントリーが編集エリアに表示されます。

ComboBoxStyleプロパティを使用して、コンボ・ボックスのスタイル（ユーザーがエントリーを追加できるかどうかやリストがドロップ・ダウンか通常のリストなのか）を設定します。

コンボ・ボックスに表示するデータを定義するには、リポジトリ・ブラウザのフィールドまたはファイルのタブから必要なフィールドをドラッグすることから始めます。コンボ・ボックスにドラッグされたフィールドは全てコンボ・ボックス・カラムを作成します。このカラムには独自のプロパティがあり、これを修正できます。

コンボ・ボックスを定義した後、取り出すデータを指定し、SELECTやADD_ENTRYステートメントを使ってリストに追加します。

コンボ・ボックスに複数のカラムが存在できる場合でも、編集エリアに表示されるのは1つのカラムだけであることに注意してください。これはDisplayPositionが1に設定されたカラムです。

実行時リスト部分でエントリーが選択されていない場合、コンボ・ボックスの編集エリアは空です。フォームが最初に表示された時にコンボ・ボックスの編集エリアにエントリーを表示する場合は、コンボ・ボックスにフォーカスが当たっていることを確認してください。これはTabPositionを1に設定するか、次のようなコードをフォームの初期化イベントに追加します。

```
GET_ENTRY NUMBER(1) FROM_LIST(#CMBX_1)
SET #CMBX_1.CURRENTITEM FOCUS(TRUE)
```

コンボ・ボックスを動的に作成する方法の例については、SET例67を参照してください。 [動的にドロップダウン・リストを作成する](#)

また以下も参照してください。

[コンボ・ボックスのセル](#)

[コンボ・ボックスのプロパティ](#)

[コンボ・ボックスのイベント](#)

コンボ・ボックスのメソッド

コンボ・ボックスのプロパティ

AutoSelectedItem プロパティ

Cell プロパティ

ComboBoxStyle プロパティ

ComponentVersion プロパティ

DropDownCount プロパティ

FixedHeight プロパティ

IntegralHeight プロパティ

AutoSelectedItem プロパティ

AutoSelectedItemは自動的に最初の項目を設定します。

コンボ・ボックスにフォーカスが当たった時にリストの最初の項目でドロップダウンのコンボ・ボックスの編集部分を埋めたくない場合は、AutoSelectedItemプロパティをFalseを設定します。

AutoTabの使用方法を確認するには、次のソースをコピーしてフォームに貼り付け、フォームをコンパイルして実行してください。

```
FUNCTION options(*DIRECT)
BEGIN_COM role(*EXTENDS #PRIM_FORM) CAPTION('AutoSelectedItem
DEFINE_COM class(#PRIM_CKBX) name(#CKBX_AUTOSEL) CAPTION(
DEFINE_COM class(#PRIM_CMBX) name(#CMBX_1) AUTOSELECT(False)
DEFINE_COM class(#PRIM_CBCL) name(#CBCL_1) DISPLAYPOSITION(
DEFINE_COM class(#PRIM_PHBN) name(#PHBN_1) CAPTION('Reload') I
EVTROUTINE handling(#com_owner.Initialize)
SET com(#com_owner) CAPTION(*component_desc)
SELECT fields(#CMBX_1) from_file(PSLMST)
ADD_ENTRY to_list(#CMBX_1)
ENDSELECT
ENDROUTINE
EVTROUTINE handling(#PHBN_1.Click)
CLR_LIST named(#CMBX_1)
SELECT fields(#CMBX_1) from_file(PSLMST)
ADD_ENTRY to_list(#CMBX_1)
ENDSELECT
ENDROUTINE
EVTROUTINE handling(#CKBX_AUTOSEL.Click)
IF cond('#CKBX_AUTOSEL.ButtonState = Checked')
SET com(#CMBX_1) AUTOSELECTITEM(TRUE)
ELSE
SET com(#CMBX_1) AUTOSELECTITEM(false)
ENDIF
ENDROUTINE
END_COM
```

Cell プロパティ

Cellプロパティを使用して、コンボ・ボックスのセルの処理を行います。

個別のセルやカラムにイメージを割り当てることができます。

また以下も参照してください。

[Column プロパティ](#)

[Row プロパティ](#)

Column プロパティ

セルのColumnプロパティは、セルの列の識別をします。

Row プロパティ

セルのRowプロパティは、セルの行の識別をします。

ComboBoxStyle プロパティ

ComboBoxStyleはコンボ・ボックスのタイプとそのリスト部分の動作を設定します。

コンボ・ボックスのスタイルは以下のとおりです：

Simple: 常に表示される編集エリアとリスト部分を含みます。ユーザーはリストから選択でき、編集エリアでタイプもできます。

Drop-Down: ユーザーが編集ボックスのスピン・ボタンをクリックすると、編集ボックスに付いているリストのドロップダウンが開きます。ユーザーはリストから選択でき、編集エリアで新しいエントリーをタイプすることもできます。

Drop-Down List: Drop-downと同様ですが、ユーザーが編集エリアでタイプできません。

drop-downやdrop-down listコンボ・ボックスが最初に表示される時、リスト部分は表示されないため、編集エリアにエントリーが含まれていることを必ず確認するようにしてください。これはコンボ・ボックスにフォーカスを設定することで可能になります。

```
INVOKE #CMBX_1.SETFOCUS
```

これにより、リストの最初のエントリーが表示されます。もしくは、コンボ・ボックスの特定の項目にフォーカスを設定することもできます。

```
GET_ENTRY NUMBER(3) FROM_LIST(#PICK)  
SET #CMBX_1.CURRENTITEM FOCUS(TRUE)
```

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 10.0 以前のバージョンです。

1 はLANSA 10.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

コンボ・ボックスのComponentVersionが1に設定されると、ShowSelection プロパティはDropDownList コンボ・スタイルの処理をします。

またComponentVersionが1に設定されると、DLT_ENTRY コマンドはItemGotFocus とItemLostFocus イベントを起動します。

DropDownCount プロパティ

DropDownCountは表示する項目数を設定します。

このプロパティには、Drop-DownまたはDrop-Down Listのコンボ・ボックスに表示されるエントリーの数指定します。（HeightプロパティはComboBoxStyleがSimpleのコンボ・ボックスにのみ適用されます。）

リストに多くのエントリーがある場合は、スクロール・バーが自動的に表示されます。

FixedHeight プロパティ

FixedHeight プロパティを使って、コンボ・ボックスの高さが調整できるかどうかの制御を行います。

省略値では、FixedHeight プロパティはTrueに設定されます。Falseに設定されている場合、コンボ・ボックスの垂直方向のサイズ変更は可能です。

IntegralHeight プロパティ

IntegralHeightはリストの最後の項目がどのように表示されるかを指定します。

IntegralHeightにより、リストに項目の一部分を表示するかどうかを制御します。このプロパティがTrueに設定されていると、行の一部分が表示されることはありません。

コンボ・ボックスのイベント

CloseUp イベント

DropDown イベント

CloseUp イベント

CloseUpイベントは、コンボ・ボックスのリスト・ボックス部分が非表示の時に起動します。

DropDown イベント

このイベントはコンボ・ボックスのリスト部分がドロップダウンした時に起動されます。

ComboBoxStyleがSimpleに設定されている時はこのイベントは起動されません。

[Continue](#) パラメータ

Continue パラメータ

ContinueパラメータをTrueに設定すると、イベントの処理が進められ、Falseにするとイベントはキャンセルされます。

このパラメータは現在コンボ・ボックスのDropDownにのみ使用されていますが、将来は他の用途にも広げられる予定です。

コンボ・ボックスのメソッド

[OpenDropDown](#) メソッド

[CloseDropDown](#) メソッド

OpenDropDown メソッド

OpenDropDownメソッドはDropDownまたはDropDownListスタイルのコンボ・ボックスのリスト・ボックス部分を強制的に表示します。

コントロールにドロップダウンのリスト・ボックスがあり、そのリスト・ボックスがドロップダウンされていない時に、OpenDropDownはTrueを返し、それ以外はFalseを返します。

CloseDropDown メソッド

CloseDropDownメソッドはDropDownまたはDropDownListスタイルのコンボ・ボックスのリスト・ボックス部分を強制的に閉じます。

コントロールにドロップダウンのリスト・ボックスがあり、そのリスト・ボックスがドロップダウンされている時に、CloseDropDownはTrueを返し、それ以外はFalseを返します。

コンボ・ボックスのセル

コンボ・ボックスのセルを使って、コンボ・ボックスの個別のセルの処理を行います。

コンボ・ボックスのセルのプロパティ

コンボ・ボックスのセルのプロパティ

Column プロパティ

Item プロパティ

ComboBox プロパティ

Column プロパティ

セルのColumnプロパティは、セルの列の識別をします。

Item プロパティ

Itemプロパティを使用して、セルに対応するコンボ・ボックスの項目コンポーネントを返します。

ComboBox プロパティ

ComboBoxプロパティはこのセルが属しているコンボ・ボックスの名前を返します。

エクスプローラ

ファイルとフォルダを表示します。

エクスプローラ・コンポーネントを使用して、ローカル・ハードディスクやネットワーク上のファイルやディレクトリを表示させます。

エクスプローラ・コンポーネントのイベントはファイル名、パス、そしてパス・タイプをパラメータとして引き渡します。次のソース例では、エクスプローラ・コンポーネントで項目がフォーカスを受けた時に、上記の情報をLANSAフィールドに割り当てます。

```
Evtroutine Handling(#ExplorerLeft.ItemGotFocus  
#ExplorerRight.ItemGotFocus) Path(#Path) Pathtype(#Type) Name(#File)
```

```
#FileName := #File  
#PathName := #Path  
#PathType := #Type
```

```
Endroutine
```

2つのエクスプローラ・コンポーネント、つまり1つはディレクトリとパスの表示用にそしてもう1つはその中のファイルを表示用に（Windowsエクスプローラと同様に）使用するには、最初のエクスプローラ・コンポーネントのNotifyComponentプロパティを使ってこの2つのコンポーネント間の通信を導入します。

エクスプローラの使用法の例については、SET例44を参照するか、以下のコードをコピーしてフォームに貼り付けます。（上記のイベント・ルーチン・ハンドラーも含めてください）

```
Function Options(*DIRECT)  
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('Explorer Sample  
Application') Clientheight(432) Clientwidth(767) Height(470) Left(132)  
Top(212) Width(783)  
Define_Com Class(#PRIM_DCBX) Name(#ExplorerLeft) Displayposition(1)  
Fileincludemask('*.*') Filename('Desktop') Height(329) Left(8)  
Notifycomponent(#ExplorerRight) Parent(#COM_OWNER) Tabposition(1)  
Tabstop(False) Top(24) Width(345)  
Define_Com Class(#PRIM_DCBX) Name(#ExplorerRight)  
Displayposition(2) Displaystyle(GeneralListView) Fileincludemask('*.*')  
Filename('Desktop') Height(329) Left(360) Parent(#COM_OWNER)
```

```
Tabposition(2) Tabstop(False) Top(24) Width(393)
Define_Com Class(#PRIM_LABL) Name(#LABL_2) Caption('Explorer 1')
Displayposition(3) Height(17) Left(8) Parent(#COM_OWNER)
Tabposition(3) Tabstop(False) Top(8) Width(98)
Define_Com Class(#PRIM_LABL) Name(#LABL_3) Caption('Explorer 2')
Displayposition(4) Height(17) Left(360) Parent(#COM_OWNER)
Tabposition(4) Tabstop(False) Top(8) Width(98)
Define_Com Class(#STD_QSEL.Visual) Name(#FileName)
Caption('Filename') Displayposition(5) Height(20) Labeltype(Caption) Left(8)
Marginleft(120) Parent(#COM_OWNER) Tabposition(5) Top(408)
Usepicklist(False) Width(500)
Define_Com Class(#STD_QSEL.Visual) Name(#PathName) Caption('Path')
Displayposition(6) Height(20) Labeltype(Caption) Left(8) Marginleft(120)
Parent(#COM_OWNER) Tabposition(6) Top(384) Usepicklist(False)
Width(500)
Define_Com Class(#STD_QSEL.Visual) Name(#PathType) Caption('Path
type') Displayposition(7) Height(20) Labeltype(Caption) Left(8)
Marginleft(120) Parent(#COM_OWNER) Tabposition(7) Top(360)
Usepicklist(False) Width(500)
```

```
Evtroutine Handling(#ExplorerLeft.ItemGotFocus
#ExplorerRight.ItemGotFocus) Path(#Path) Pathtype(#Type) Name(#File)
```

```
#FileName := #File
#PathName := #Path
#PathType := #Type
```

```
Endroutine
```

```
End_Com
```

エクスプローラのプロパティ

AppendPathSeparator プロパティ	DriveSpaceFree プロパティ	NotifyComponent プロパティ
ApplySecurity プロパティ	DriveSpaceTotal プロパティ	Path プロパティ
ArrangeBy プロパティ	FileName プロパティ	PathType プロパティ
AutoRefresh プロパティ	FileTypeMask プロパティ	RootPath プロパティ
DisplayStyle プロパティ	IncludeMask プロパティ	VisiblePath プロパティ

ArrangeBy プロパティ

項目が表示される順番を指定します。

ArrangeByプロパティを利用して、FileListBoxビューのソートの順番を制御します。

ファイル名や日付、サイズやファイル拡張子によってソートできます。昇順だけがサポートされます。

DisplayStyle プロパティ

複数の表示スタイルから1つ選択します。

DisplayStyleプロパティを使用して、エクスプローラ・コンポーネントで使用可能な表示スタイルの1つを選択します。 次のいずれかのタイプを選択することができます：

Directory list box

Directory list view

Directory tree view

Drive combo box

File list box

File list view

General list view.

DriveSpaceFree プロパティ

現在選択されたパスで使用可能な空きディスク・スペースの量を返します。

現在のドライブの使用可能なディスクの空きスペースの量を調べるには、DriveSpaceFreeを使用します。

値はメガバイトで返されます。

DriveSpaceTotal プロパティ

現在選択されているディスク・ドライブのサイズをメガバイトで示します。

DriveSpaceTotal プロパティを使用して、現在選択されているディスク・ドライブのサイズ (メガバイト) を調べます。

返された値は使用されているスペースに関係なく、全体の容量を示しています。

Path プロパティ

ネットワークまたはローカル・ドライブの現在の場所を指定します。

Path プロパティを使用して、ネットワークもしくはローカル・ドライブ内の現在の場所を指定します。

PathType プロパティ

PathType プロパティを使って、コンポーネント表示内の選択された項目を識別します。

タイプは次の通りです：

Desktop	Local Drive
My Computer	Network Drive
Network Neighborhood	Mapped Network Drive
Global Network	Folder
	File

例えば、ユーザーがツリー・ビュー表示モードの時にデスクトップのアイコンをクリックすると、PathType プロパティはDesktopに設定されます。

IncludeMask プロパティ

ファイル名のフィルター・メカニズムです。

エクスプローラ・コンポーネントで検索時にファイル名をフィルターし、一致するものだけを表示できるようにします。

FileIncludeMaskではアクタリスク '*' と疑問符 '?' がサポートされます。

'*' はどんな文字列にも一致することを示します。

'?' はどんな文字にも一致することを示します。

例：

. - 全てのファイルが検索に一致します。

*.ico - ファイル拡張子が ".ico" の全てのファイルが検索に一致します。

?? .ico - ファイル拡張子 ".ico" の前に2文字あるファイルが全て検索に一致します。

t*.ico - 't' という文字で始まる、".ico" ファイルが全て検索に一致します。

FileTypeMask プロパティ

特定の属性のファイルの表示を制御します。

FileTypeMaskを使用して、どのファイルを表示するか制御します。ファイル・フィルタリングは次のようなファイル属性に基づいて行われます：

All

Archive

Directory

Hidden

Normal

ReadOnly

System

AutoRefresh プロパティ

ファイル・システムの変更を自動的に更新します。

このプロパティは現在使用できません。

Trueに設定されると、エクスプローラ・コンポーネントがファイル・システムに対して行われる変更を監視し、変更されると自動的に表示を更新します。

省略値はFalseです。

免責事項

この値がTrueに設定されると、AutoRefreshはSystem Internals Corporationの"FILEMON.SYS and FILEMON.VXD"という2つのサードパーティ・デバイス・ドライバを使用します。

System Internals Corporationはこのソフトウェアを現状のまま提供し、特殊な用途に対する適合性および適切性に関しては何の保証もしません。

このソフトウェアには欠陥が含まれている可能性もあります。ですからこのツールは各自の責任において使用してください。

詳しい情報に関しては、www.sysinternals.comを参照してください。

この使用により障害が発生した場合、それが故意でなかったとしても、Aspect Computing と販売代理店はいかなる責任も負わないものとし、一切の賠償は行わないものとします。

配布に関する考慮事項

エクスプローラ・コンポーネントのAutoRefreshが有効なアプリケーションを配布する時は、必ず "Filemon.sys" と "Filemon.vxd" もともに配布する必要があります。この2つはX_WIN95\X_LANSA\EXECUTEまたはシステム・パスを含むディレクトリにインストールされなければなりません。

エクスプローラ・コンポーネントがこれらのドライバを見つけられないと、AutoRefreshは無効になります。

AppendPathSeparator プロパティ

パス名がどのように構成されるかを制御します。

AppendPathSeparatorを使用して、パス名をどのように作成するかを制御します。

このプロパティの値がTRUEの時、自動的にバックスラッシュ文字が追加されます。

FileName プロパティ

現在選択されているファイルです。

Filename プロパティを使用して、現在選択されているファイル名を調べます。

NotifyComponent プロパティ

このコンポーネントの変更に対応するエクスプローラ・コンポーネントを識別します。

NotifyComponent プロパティを使用して、2つのエクスプローラ・コンポーネント間の通信を構築します。Windowsエクスプローラのようなアプリケーションを構築する際に便利です。これは以下の手順で行います：

1つのフォームに2つのエクスプローラ・コンポーネントを横並びに定義します。

設計時は左側のコンポーネント（仮にDCBX_1とします）のDisplayStyle プロパティをDirectoryTreeViewに設定し、もう1つのコンポーネント（DCBX_2）のDisplayStyleにはGeneralListViewを設定します。

DirectoryTreeView (DCBX_1) を選択し、"NotifyComponent" プロパティを *NULL から DCBX_2 に変更します。

これで両方のコンポーネントが自動的に互いの変更に対応します。

この通信を無効にするには、プロパティを省略値*NULLに戻します。

ApplySecurity プロパティ

ApplySecurity プロパティを使って、ユーザーがどのリソースを見ることができるかを制御します。

このプロパティがTrueの時、Windowsのセキュリティ設定でユーザーに閲覧権限のあるリソースのみがエクスプローラに表示されます。

RootPath プロパティ

RootPathパラメータを使って、ルート・パスの名前を指定します。

エクスプローラのVisiblePathがRootpathに設定されている時、ユーザーのアクセスはこのプロパティに指定されたパス以下のオブジェクトに制限されます。

VisiblePath プロパティ

エクスプローラに表示されるパスを制御するには、VisiblePathプロパティを使用します。

この値には次のようなものがあります：

- LocalDrivesOnly：ユーザーのアクセスはローカル・マシンのリソースに限られます。
- RootPath：ユーザーのアクセスはRootPathプロパティに指定されたパスより下のリソースに限られます。
- Unrestricted：ユーザーは全てのリソースにアクセスできます。
(ApplySecurityプロパティによる制限を除く)

エクスプローラのイベント

[ItemDoubleClick イベント](#)

[PathChanged イベント](#)

ItemDoubleClick イベント

ユーザーが項目上で二度クリックすると起動します。

PathChanged イベント

パス名が変更されると、PathChangedイベントが起動されます。

エクスプローラのメソッド

Refresh メソッド

Refresh メソッド

Refreshメソッドを使用して、エクスプローラの内容を更新します。

ツリー・ビュー

データの階層表示です。

ツリー・ビューを使って、ラベルや任意のビットマップから成る項目の階層的なリストを表示します。ツリー・ビューは通常ドキュメントの見出し、インデックスのエントリー、ディスク上のファイルやディレクトリなど階層的な表示に適した情報を表示する際に使用されます。

まずはツリー・ビューのコントロールをフォームにドラッグしてツリー・ビューを作成するところから始めます。そして、リポジトリから表示するフィールドをツリー・ビューにドラッグします。ツリー・ビューに追加するフィールドは全てツリー・ビュー内にカラムを作成します。省略値ではこのツリー・ビューのカラムはTVCL_1、TVCL_2などのような名前が付けられます。

ツリー・ビュー・カラムの階層を設定するには、Levelプロパティを使用します。Level 1が一番上のレベルです。

Level内でDisplayPositionプロパティを利用して、フィールドのコンテンツをどのように表示するかを指定します。0はコンテンツを非表示にします。1はコンテンツを表示します。

ツリー・ビュー・カラムのKeyPositionプロパティを使って、そのフィールドがツリー・ビューのキーのどの部分に位置するかを指定します。0はフィールドをキーの一部として使用しません。ツリー・ビューのすべてのレベルには少なくとも1つのキーがなければなりません。

例えば、第1レベルが社員番号と苗字からなるツリー・ビューを作成するには、この両方のカラムのLevelプロパティに1を設定します。社員番号は表示したくないので、DisplayPositionを0に設定します。ただしツリー・ビューのキーとして使用するので、エントリーが一意であることを確認し、KeyPositionを1に設定します。（ツリー・ビューのキーに複数のフィールドを使用する場合は、KeyPositionでその相対的順序を割り当てます。）

苗字はキーの一部ではないので、KeyPositionを0に設定しますが、これを表示するにはDisplayPositionを1に設定します。

ツリー・ビューを作成後、SELECTとADD_ENTRYコマンドを使ってビューを埋めていきます。

ツリー・ビューにどのようなプロパティの項目があるかを確認するには、「[CurrentItem プロパティ](#)」を参照してください。

ツリー・ビュー項目の前のイメージはツリー・ビュー・カラムまたは個

別のツリー・ビュー項目のimageプロパティで割り当てることができません。（詳細は「[CurrentItem プロパティ](#)」を参照してください。）

[ツリー・ビューのプロパティ](#)

[ツリー・ビューのメソッド](#)

[ツリー・ビューのイベント](#)

ツリー・ビューのプロパティ

Checkboxes プロパティ	LinesAtRoot プロパティ
ComponentVersion プロパティ	ManageChildren プロパティ
EntriesAtRoot プロパティ	MultipleSelectStyle プロパティ
HasButtons プロパティ	Level プロパティ
HasLines プロパティ	ValueAt プロパティ

Checkboxes プロパティ

Use the CheckBoxes property to add checkboxes to the tree view.

このプロパティがTrueに設定されていると、チェック・ボックスが表示されます。

ユーザーがクリックする、またはその項目のCheckedプロパティがプログラム上で設定されると、チェック・ボックスが選択されます。ツリー・ビューの項目が選択されているかどうかを調べるには、項目のCheckedプロパティを使います。

チェックボックスをクリックすることで、ツリー・ビューのItemChangedイベントが起動されます。

CheckEnabledを使って、項目のチェックボックスを無効にすることもできます。

ComponentVersion プロパティ

ComponentVersion プロパティを使って、どのバージョンのLANSAのコンポーネントを使用するかを指定します。

0 はLANSA 10.0 以前のバージョンです。

1 はLANSA 10.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

2 はLANSA 11.0 以降のバージョンです。この値を選択すると、以前のバージョンのLANSAではコンポーネントが機能しない可能性があります。

ComponentVersion プロパティは、機能が拡張されたコンポーネントで使用可能で、LANSAの以前のバージョンと互換性がない場合があります。

ComponentVersionが1 の時、VisualStyleで定義されたAlternate color、ItemBackColorおよびItemForeColorはツリー・ビューに表示されます。

またComponentVersionが1に設定されると、DLT_ENTRYコマンドはItemLostFocusイベントを起動します。

ComponentVersionが2 の時、VisualStyleで定義されたItemBackColorおよびItemForeColorがツリー・ビューに表示されます。

EntriesAtRoot プロパティ

EntriesAtRootはツリー・ビューにいくつの項目が含まれているかを示します。

EntriesAtRootを使用して、ツリー・ビューのルートにあるエントリーの数を確認します。これは読み取り専用のプロパティです。

ツリー・ビューの特定のノードのエントリーの数を確認したい時は、その特定の項目（ノード）のHasChildreプロパティを使用します。

HasButtons プロパティ

HasButtonsは + や - の記号を表示・非表示します。

HasButtonsプロパティを使用して、ツリー・ビューに他の項目が含まれているかどうかを示す + と - のアイコンを表示したり、非表示にしたりします。

HasLines プロパティ

HasLinesはツリー・ビューの線を表示・非表示します。

HasLinesプロパティを使用して、ツリー・ビューに表示する線を表示したり、非表示にしたりします。

ManageChildren プロパティ

ManageChildrenはツリーが折りたたまれた時の動作を制御します。

ManageChildrenプロパティはツリー項目が折りたたまれた時に子の項目を自動的に削除するかどうかを制御します。通常これはItemExpandingイベントの最中に子がロードされた時に使用されます。TrueまたはFalseに設定できます。

Level プロパティ

Levelはリストの格納方法を制御します。

Levelプロパティを使用して、フィールドをツリー・ビューのどのレベルに表示するかを指定します。

Level 1が一番上のレベルです。同じレベルに複数のフィールドを格納できます。

ValueAt プロパティ

Unlevelledのツリー・ビュー項目の値にアクセスします。

RESULT

リスト要素のバリエーション値です。

ROW

アクセスする行の整数インデックス

COLUMN

アクセスする列の整数インデックス

例

```
#Value := #TreeV1.ValueAt<#RowIdx #ColIdx>
```

LinesAtRoot プロパティ

LinesAtRootはツリーのルート・レベルの線を表示・非表示します。

LinesAtRootプロパティを使用して、ツリー・ビューのルート・レベルに表示する線を表示したり、非表示にしたりします。

MultipleSelectStyle プロパティ

LinesAtRootは複数選択の有無を設定します。

MultipleSelectStyleプロパティを使用して、複数のツリー項目が選択できるかどうか制御します。

このプロパティには次の値が設定できます：

- All: どの項目にも複数選択が可能です。
- SameParent: フォーカスの項目と同じ親の項目のみ選択できます。
- SameLevel. フォーカスの項目と同じレベルの項目のみ選択できます。

ツリー・ビューのメソッド

[CollapseAll](#) メソッド

[SetValue](#) メソッド

[StartLabelEdit](#) メソッド

CollapseAll メソッド

CollapseAllメソッドを使用して、ツリー内の展開された全てのレベルを折りたたみます。

次の例は#trvw_1内の全ての展開されたレベルを折りたたみます：

```
Invoke Method(#trvw_1.Collapseall)
```

SetValue メソッド

Unlevelledのツリー・ビュー項目に値を設定します。

StartLabelEdit メソッド

StartLabelEditメソッドを使って、ツリー・ビュー項目のラベル・テキストの編集を開始します。

ツリー・ビューのイベント

ExpandItem パラメータ

ItemExpanding イベント

ItemCollapsed イベント

ExpandItem パラメータ

ExpandItemをFalseに設定すると、ダブルクリックで展開以外のアクションを実行します。

例えば、LANSAエディターでファイル上でダブルクリックされるとそのファイルを開きます。このパラメータはTreeItemも展開されてしまわないようにするためのものです。

ItemExpanding イベント

ItemExpandingはツリー・ビューのイベントで、次のレベルのエントリーを表示します。

ItemExpandingイベントはツリー・ビューが展開されて次のレベルのエントリーが表示される時に起動します。通常はこのレベルのデータをロードする際に使用されます。

次の例ではLevelプロパティと共にこのイベントをどのように使用するかを示しています。

```
DEFINE FIELD(#LEVEL) TYPE(*DEC) LENGTH(7) DECIMALS(0) DESC
EVTROUTINE HANDLING( #TRVW_1.ItemExpanding )
change #level #Trvw_1.CurrentItem.Level
case #level
when '= 1'
select (#section #secdesc) from_file(sectab) with_key(#deptment)
add_entry #trvw_1
endselect
when '= 2'
select (#empno #surname #givename) from_file(pslmst1) with_key(#deptment)
add_entry #trvw_1
endselect
endcase
ENDROUTINE
```

ItemCollapsed イベント

ItemCollapsedイベントは、ツリー・ビューのあるレベルのエントリーを閉じます。

ItemCollapsedイベントは、ツリー・ビューが1つのレベルのエントリーを隠して閉じられた時に起動します。通常このイベントは変更されたまたは新しいエントリーを保存する際に使用されます。

RESULT

設定が成功(TRUE)したかどうかを示すブール値です。

ROW

設定する行の整数インデックス

COLUMN

設定する列の整数インデックス

VALUE

設定する値

例

```
#TreeV1.SetValue( 1 1 "tv1" )
```

ツリー項目

ツリー項目のプロパティ

ツリー項目のメソッド

ツリー項目のプロパティ

Bold プロパティ	Expanded プロパティ	MarginTop プロパティ
Checked プロパティ	HasChildren プロパティ	ParentItem プロパティ
CheckEnabled プロパティ	HasLines プロパティ	Position プロパティ
ChildrenCount プロパティ	Level プロパティ	Tree プロパティ
CheckReadOnly プロパティ	MarginBottom プロパティ	ValueAt プロパティ

Bold プロパティ

Boldプロパティはツリー・ビュー項目を太字テキストで表示するかどうかを制御します。

次のコードは、ツリー・ビュー内の現在の項目を太字に設定します。

```
Set #trvw_1.CurrentItem Bold(True)
```

ChildrenCount プロパティ

読み取り専用のプロパティです。

ChildrenCount プロパティを使用して、ツリー・ビューのエントリーに
従属するエントリーの数を確認します。

HasChildren プロパティ

HasChildrenプロパティを使用して、ツリー・ビューのエントリーが従属するエントリーを持てるかどうかを指定します。このプロパティの値は、Unknown、NoまたはYesになります。

HasLines プロパティ

HasLines プロパティを使用して、ツリー・ビュー項目に表示する線を表示したり、非表示にしたりします。

ParentItem プロパティ

実行時のプロパティです。

ParentItemプロパティを使用して、ツリー・ビュー項目の親を返す、もしくは親を設定します。

Level プロパティ

Levelはツリ・ビュー項目のレベルを示します。

読み取り専用のプロパティです。

Levelプロパティを使用して、ツリー・ビュー項目のレベルを確認します。次のコードは現在の項目のレベルによって異なったアクションを実行します：

```
if '#treeview.currentitem.level *eq 1'  
  execute new_dept  
else  
if '#treeview.currentitem.level *eq 2'  
  execute new_sect  
else  
if '#treeview.currentitem.level *eq 3'  
  execute new_empl  
endif  
endif  
endif
```

Expanded プロパティ

Expandedプロパティを使用して、ツリー・ビュー項目が展開されているか、折りたたまれているかを制御または確認します。このプロパティの値は、TrueまたはFalseになります。

CheckEnabled プロパティ

CheckEnabled プロパティを使用してツリー項目のチェックボックスが有効かどうかを制御します。

次のコードはツリー内の現在の項目のチェックボックスを無効にします。

```
Set Com(#trvw_1.CurrentItem) CheckEnabled(False)
```

CheckReadOnly プロパティ

チェックボックス内の項目を書き込み禁止にします。

このプロパティを使って、項目のチェックボックスの編集を有効・無効にします。

Checked プロパティ

Checkedプロパティを使用してツリー項目のチェックボックスがチェックされるかどうかを制御します。

このプロパティの値は、True、FalseまたはGrayedになります。

次のコードはツリー内の現在の項目のチェックボックスを選択します。

```
Set Com(#trvw_1.CurrentItem) Checked(True)
```

Tree プロパティ

Tree プロパティはこの項目が属するツリー・ビューです。
次のコードは項目のツリー・ビュー名を返します。

```
Set Com(#std_descs) Value(#trvw_1.CurrentItem.Tree.name)
```

MarginBottom プロパティ

MarginBottomはツリー項目の下により多くの間隔をあけます。

MarginBottomプロパティを使用して、ツリー・ビュー項目の下の下部余白を増やします。マージンはピクセルで指定します。

通常はツリー・ビューにより提供される自動間隔を変更する必要はありませんが、場合によっては余白を調整することでより見栄えがよくなることもあります。

次のコードをコピー・貼り付けして、ツリー・ビューの上部と下部余白が変更される方法を確認してください。

```
FUNCTION options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(313) Clientwidth
Define_Com Class(#PRIM_TRVW) Name(#TRVW_1) Displayposition(1) Height
Define_Com Class(#PRIM_TVCL) Name(#TVCL_1) Keyposition(1) Level(1)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_3) Keyposition(1) Level(2)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_2) Displayposition(1) Image
Define_Com Class(#PRIM_TVCL) Name(#TVCL_4) Displayposition(1) Image
Define_Com Class(#PRIM_TVCL) Name(#TVCL_5) Keyposition(1) Level(3)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_6) Displayposition(1) Image
Define_Com Class(#STD_NUM.Visual) Name(#STD_NUM) Caption('Bottom')
Define_Com Class(#STD_NUM.Visual) Name(#STD_NUM_1) Caption('Top')
EVTROUTINE handling(#TRVW_1.Initialize) options(*NOCLEARMESSAGES)
SELECT fields(#EMPNO #SURNAME #DEPARTMENT #SECTION) from_file
FETCH fields(#DEPTDESC) from_file(DEPTAB) with_key(#DEPARTMENT)
FETCH fields(#SECDESC) from_file(SECTAB) with_key(#DEPARTMENT #SECTION)
ADD_ENTRY to_list(#TRVW_1)
ENDSELECT
ENDROUTINE
EVTROUTINE HANDLING(#STD_NUM_1.Changed) OPTIONS(*NOCLEARMESSAGES)
Set Com(#TRVW_1.CurrentItem) Margintop(#std_num_1.value)
ENDROUTINE
EVTROUTINE HANDLING(#STD_NUM.Changed) OPTIONS(*NOCLEARMESSAGES)
Set Com(#TRVW_1.CurrentItem) MarginBottom(#std_num.value)
ENDROUTINE
END_COM
```

MarginTop プロパティ

MarginTopはツリー項目の上により多くの間隔をあけます。

MarginBottomプロパティを使用して、ツリー・ビュー項目の上の上部余白を増やします。マージンはピクセルで指定します。

通常はツリー・ビューにより提供される自動間隔を変更する必要はありませんが、場合によっては余白を調整することでより見栄えがよくなることもあります。

Position プロパティ

Positionはツリー項目の再配置を許可します。

Positionプロパティを使用して、ツリー項目をそのレベル内での位置を変更します。

ツリー項目の位置は、ViewStyle(Unlevelled)のツリー・ビューで、ツリー・ビューにソート欄がない時のみ変更が可能です。

ValueAt プロパティ

Unlevelledのツリー・ビュー項目の値にアクセスします。

RESULT

リスト要素のバリエーション値です。

COLUMN

アクセスする列の整数インデックス

例

```
#treeItem1 := #Tree1.Items<1>.ValueAt(2)
```

ツリー項目のメソッド

リスト項目のCollapse メソッド

リスト項目のExpand メソッド

SetValue メソッド

リスト項目のCollapse メソッド

Collapseメソッドを使用して、展開されていたツリー項目をプログラム上で折りたたみます。

リスト項目のExpand メソッド

Expandメソッドを使用して、ツリーの項目をプログラム上で展開します。

次のコードをコピー・貼り付けして、ExpandとCollapseメソッドがどのように作動するか確認してください。

```
FUNCTION options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(316) Clientwidth
Define_Com Class(#PRIM_TRVW) Name(#TRVW_1) Displayposition(1) Height(316)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_1) Keyposition(1) Level(1)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_3) Keyposition(1) Level(2)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_2) Displayposition(1) Image
Define_Com Class(#PRIM_TVCL) Name(#TVCL_4) Displayposition(1) Image
Define_Com Class(#PRIM_TVCL) Name(#TVCL_5) Keyposition(1) Level(3)
Define_Com Class(#PRIM_TVCL) Name(#TVCL_6) Displayposition(1) Image
Define_Com Class(#PRIM_PHBN) Name(#PHBN_1) Caption('Expand') Displayposition(1)
Define_Com Class(#PRIM_PHBN) Name(#PHBN_2) Caption('Collapse') Displayposition(1)
EVTROUTINE handling(#TRVW_1.Initialize) options(*NOCLEARMESSAGES)
SELECT fields(#EMPNO #SURNAME #DEPARTMENT #SECTION) from_file
FETCH fields(#DEPTDESC) from_file(DEPTAB) with_key(#DEPARTMENT)
FETCH fields(#SECDESC) from_file(SECTAB) with_key(#DEPARTMENT #SECTION)
ADD_ENTRY to_list(#TRVW_1)
ENDSELECT
ENDROUTINE
EVTROUTINE HANDLING(#PHBN_1.Click)
Invoke Method(#TRVW_1.CurrentItem.Expand)
ENDROUTINE
EVTROUTINE HANDLING(#PHBN_2.Click)
Invoke Method(#TRVW_1.CurrentItem.Collapse )
ENDROUTINE
END_COM
```

SetValue メソッド

Unlevelledのツリー・ビュー項目に値を設定します。

RESULT

設定が成功(TRUE)したかどうかを示すブール値です。

COLUMN

設定する列の整数インデックス

VALUE

設定する値

例

```
#TreeV1.Items<2>.SetValue ( 1 "tv1" )
```

リスト・タイプのコンポーネントのカラム

カラムとは、リスト、コンボ・ボックス、ツリー・ビュー、グリッド、グラフまたは複数行編集ボックス内の欄のことです。これはリポジトリのフィールドに相当します。

カラムの様々なプロパティを使って、位置や項目のソート順などの属性を設定できます。

[リスト・タイプ・コンポーネントのカラムのプロパティ](#)

[リスト・タイプのコンポーネントのイベント](#)

[リスト・タイプのコンポーネントのメソッド](#)

リスト・タイプ・コンポーネントのカラムのプロパティ

CaptionAlign プロパティ	EditAppearance プロパティ	SortAsColumn プロパティ
CaptionText プロパティ	Editor プロパティ	SortDirection プロパティ
CaptionType プロパティ	KeyPosition プロパティ	SortOnClick プロパティ
Checkboxes プロパティ	Level プロパティ	SortPosition プロパティ
ColumnAlign プロパティ	MinimumWidth プロパティ	SortType プロパティ
ColumnRole プロパティ	ReadOnly プロパティ	Source プロパティ
DisplayAppearance プロパティ	RowAlign メソッド	ValueAt プロパティ
DisplayPosition プロパティ	SelectOnClick プロパティ	WidthType プロパティ

RowAlign メソッド

RowAlignメソッドを使用してグリッド項目のコンテンツの垂直方向の整列を指定します。

この値にはTop、Center、Bottomが指定できます。

[SelectOnClick](#) プロパティ

[SortAsColumn](#) プロパティ

[SortDirection](#) プロパティ

[SortOnClick](#) プロパティ

[SortPosition](#) プロパティ

SelectOnClick プロパティ

SelectOnClickはカラムが選択可能かどうかを制御します。

SelectOnClickプロパティを使用して、ヘッダーをクリックすることでグリッド・カラム内の現在の項目を選択できるかどうかを指定します。このプロパティ値は、TrueまたはFalseになります。

SortAsColumn プロパティ

SortAsColumnはソートに使用するカラムを制御します。

SortAsColumnプロパティを使用して、このカラムのソートに使用される別のカラムを指定します。

あるカラムのソートを別のカラム（表示されている必要はありません）を使ってソートしたい場合もあります。例えば、あるカラムが日付をDDMMYYYY形式で表示している時、実際のソートはYYYYMMDD形式の日付カラムを利用して行うでしょう。

次の例をコピー・貼り付けして、SortAsColumnプロパティがどのように作動するかを確認することができます。

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(166) Clientwidth
Define_Com Class(#PRIM_LTVW) Name(#LTVW_1) Columnbuttonheight(2
Define_Com Class(#PRIM_LVCL) Name(#LVCL_1) Displayposition(1) Paren
Define_Com Class(#PRIM_LVCL) Name(#LVCL_2) Displayposition(2) Paren
Define_Com Class(#PRIM_LVCL) Name(#LVCL_3) Displayposition(3) Paren
Define_Com Class(#PRIM_CKBX) Name(#CKBX_1) Caption('Sort third col
EvtRoutine Handling(#com_owner.Initialize)
Change Field(#std_text) To(NSW)
Change Field(#ddmmyyyy) To(01012001)
Change Field(#yyyymmdd) To(20010101)
Add_Entry To_List(#LTVW_1)
Change Field(#std_text) To(VIC)
Change Field(#ddmmyyyy) To(03042004)
Change Field(#yyyymmdd) To(20040304)
Add_Entry To_List(#LTVW_1)
Change Field(#std_text) To(TAS)
Change Field(#ddmmyyyy) To(02022005)
Change Field(#yyyymmdd) To(20050202)
Add_Entry To_List(#LTVW_1)
Change Field(#std_text) To(WA)
Change Field(#STD_NUM) To(4)
Change Field(#ddmmyyyy) To(06072000)
Change Field(#yyyymmdd) To(20000606)
Add_Entry To_List(#LTVW_1)
Endroutine
```

```
EVTROUTINE HANDLING(#CKBX_1.Click)
if cond('#ckbx_1.buttonstate *eq Checked')
Set Com(#LVCL_3) Sortascolumn(#LVCL_1)
else
Set Com(#LVCL_3) Sortascolumn(*null)
endif
ENDROUTINE
End_Com
```

SortDirection プロパティ

SortDirectionはリストのソート方法を制御します。

SortDirectionプロパティを使用して、リストを昇順か降順でソートするかを指定します。

SortOnClick プロパティ

SortOnClickはカラムがソートされるかどうかを制御します。

SortOnClickプロパティを使用して、ヘッダーをクリックすることでカラムをソートできるかどうかを指定します。このプロパティ値は、TrueまたはFalseになります。

SortOnClickがTrueの場合、カラム・ヘッダーをクリックすると、このカラムのSortPositionは1に設定され、主要ソート・カラムになります。カラム・ヘッダーの2度目のクリックは、現在のソート順序を逆にします。つまり、当初カラムが昇順でソートされていると、クリックによりソートは降順になります。逆の場合も同じです。

SortPosition プロパティ

SortPosition プロパティは、ソートにおけるこのカラムの位置付けを決定します。

SortPosition プロパティを使って、リストやグリッドのソート階層のカラムの場所を制御します。

グリッドはまず SortPosition が 1 のカラムによりソートされ、次に SortPosition が 2 のカラムでソートなどのようになります。ソートからカラムを外すには SortPosition を 0 に設定します。

ユーザーが SortOnClick(True) のカラムをクリックしてこのカラムの SortOnClick(True) を 1 にすると、SortPosition が実行時に変更されます。またプログラムで SortPosition を変更することもできます。

Source プロパティ

Source プロパティはこのカラムを作成するのに使用されたフィールドを表示します。

ReadOnly プロパティ

ReadOnlyにより項目が書き込み禁止になります。

ReadOnlyプロパティを使って、項目の編集を有効・無効にします。

項目のReadOnlyプロパティを使って個別の項目のカラムのReadOnly設定を上書きできることに注意してください。

またこのプロパティはリスト・ビューの最初のカラムにのみ適用され、ReadOnlyプロパティの値に関係なく、リスト・ビューの他のカラムは編集できないことに注意してください。

WidthType プロパティ

WidthTypeプロパティを利用して、カラムの幅がどのように指定されるかを決定します。このプロパティは、次の値を取ることができます：

Characters: 標準の文字サイズに基づいて、カラムの幅は文字数として指定されます。

Fixed: カラムの幅はグリッドやリスト・ビュー全体のパーセントでサイズ調整され、その幅は固定されます。グリッドやリスト・ビュー全体がサイズ変更されてもカラムの幅は変わりません。この値は固定のサイズのフィールドに使用してください。FixedのWidthTypeはカラム幅を固定しているわけではありません。ユーザーはカラムをサイズ変更することは可能です。

Pixels: カラムの幅はピクセルで指定され、正確なサイズ調整がされます。

Scaleable: カラムの幅はグリッドまたはリスト・ビュー全体のパーセントでサイズ調整されます。リストやグリッド全体がサイズ変更されると、それに伴いカラムもサイズ変更されます。

Remainder: カラムはグリッドやリスト・ビューで使用されていない残りのスペースを使用します。グリッドやリスト・ビューで一番大きなカラムをRemainderで指定してください。これは左側の最初のカラムに使用するように設計されていませんので注意してください。

MinRemainder: カラムは使用されていない残りのスペースを使用し、このカラムの最低限の幅はWidthパラメータにパーセントで指定されます。

推奨の方法としては、全ての固定サイズのフィールドをWidthTypeのFixedまたはPixelsのカラムで指定し、一番大きなフィールドはWidthTypeがMinRemainderのカラムで、そして残りのカラムはScaleableで指定するようにしてください。

SortType プロパティ

SortType プロパティを利用して、カラムのコンテンツがどのようにソートされるかを指定します。

このプロパティは、次の値を取ることができます：

Word: Wordソートでは、ハイフンとアポストロフィを除く全ての句読点とその他の英数字以外の文字は英数字の前に来ます。ハイフンとアポストロフィは、その他の英数字以外の記号と異なった扱いをされます。これは"coop"と"co-op" などのような単語がソートされたリストで近くなるようにするためです。

String: Stringソートでは、ハイフンとアポストロフィはその他の英数字以外の文字と同様に扱われ、英数字の前に来ます

最もよく使用されるソートのタイプはWordです。

[Level プロパティ](#)

CaptionAlign プロパティ

CaptionAlignプロパティを利用して、キャプションの位置を設定します。キャプションは左や右に揃えたり、中央に揃えることもできます。

CaptionType プロパティ

CaptionType プロパティを使用して、カラムのヘッダーにどの種類のキャプションが使用されるかを指定します。これには次のタイプがあります：

Description: フィールド定義で指定された記述が使用されます。

Caption: カラムのCaption プロパティに入力されたテキストが使用されます。

CaptionText プロパティ

CaptionTypeプロパティを発行するのは、以下に記すツリー・ビュー、グリッド、リスト・ビューおよびプロパティ・シートのコンポーネントです。

#PRIM_TVCL#PRIM_GRID#PRIM_LTVW#PRIM_PRCL

CaptionTextプロパティはこれらのコンポーネントで発行され、キャプションにアクセスします。

例

次の例では"社員番号"を取り出します。

```
Define_Com Class(#PRIM_TBESH) Name(#listSheet) Caption('List') Displayp  
Define_Com Class(#PRIM_LTVW) Name(#ListV1) Componentversion(2) Dis  
Define_Com Class(#PRIM_LVCL) Name(#listCol1) Displayposition(1) Paren  
#listCol1.CaptionText
```

Checkboxes プロパティ

Checkboxes プロパティを使用して、このカラムにある全ての項目のチェックボックスの表示のオン・オフを切り替えます。

項目のCheckboxesを使って、個別の項目のチェックボックスの表示を制御することもできます。

次のサンプル・アプリケーションをコピー・貼り付けして、ツリー・ビューのカラムや項目のチェックボックスをどのように指定できるかを確認できます。

```
Function Options(*DIRECT)
```

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('Check Box Tester')
```

```
Clientheight(585) Clientwidth(413) Height(619) Left(300) Top(108)
```

```
Width(421)
```

```
* Tree & Column Definition
```

```
Define_Com Class(#PRIM_TRVW) Name(#TREE) Displayposition(4)
```

```
Height(241) Left(8) Parent(#COM_OWNER) Tabposition(4) Top(64)
```

```
Width(393)
```

```
Define_Com Class(#PRIM_TVCL) Name(#COL1) Displayposition(1)
```

```
Image(#VI_DEPTCL) Imageexpanded(#VI_DEPTOP) Keyposition(1)
```

```
Level(1) Parent(#TREE) Sortposition(1) Source(#DEPARTMENT)
```

```
Define_Com Class(#PRIM_TVCL) Name(#COL2) Displayposition(1)
```

```
Image(#VI_SECTCL) Imageexpanded(#VI_SECTOP) Keyposition(1)
```

```
Level(2) Parent(#TREE) Sortposition(1) Source(#SECTION)
```

```
Define_Com Class(#PRIM_TVCL) Name(#COL3) Displayposition(1)
```

```
Image(#VI_EMPLOY) Keyposition(1) Level(3) Parent(#TREE)
```

```
Sortposition(1) Source(#EMPNO)
```

```
* Tracing Definition
```

```
Define_Com Class(#PRIM_PHBN) Name(#CLEAR) Buttondefault(True)
```

```
Caption('Clear') Displayposition(2) Left(8) Parent(#COM_OWNER)
```

```
Tabposition(3) Top(4) Width(393)
```

```
Define_Com Class(#PRIM_PHBN) Name(#LOAD) Caption('Load')
```

```
Displayposition(1) Left(8) Parent(#COM_OWNER) Tabposition(2) Top(32)
```

```
Width(393)
```

```
* Sheet & Page Definition
```

```
Define_Com Class(#PRIM_TAB) Name(#SHEET) Displayposition(3)
```

```
Height(265) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(312)
```

```
Width(393)
```

```
Define_Com Class(#PRIM_TBESH) Name(#PAGE_COL) Caption('Columns')
```

Displayposition(1) Height(239) Left(4) Parent(#SHEET) Tabposition(1)
Tabstop(False) Top(22) Width(385)

* Columns Tab

Define_Com Class(#PRIM_GPBX) Name(#GPBX_DEPT)

Caption('Department') Displayposition(3) Height(58) Left(8)

Parent(#PAGE_COL) Tabposition(3) Tabstop(False) Top(32) Width(361)

Define_Com Class(#PRIM_GPBX) Name(#GPBX_SECT) Caption('Section')

Displayposition(2) Height(58) Left(8) Parent(#PAGE_COL) Tabposition(2)

Tabstop(False) Top(96) Width(361)

Define_Com Class(#PRIM_GPBX) Name(#GPBX_EMPNO)

Caption('Employee') Displayposition(1) Height(58) Left(8)

Parent(#PAGE_COL) Tabposition(1) Tabstop(False) Top(160) Width(361)

Define_Com Class(#PRIM_CKBX) Name(#CKBX_DEPT_CHK)

Caption('Check boxes') Displayposition(1) Left(16) Parent(#GPBX_DEPT)

Tabposition(1) Top(16) Width(89)

Define_Com Class(#PRIM_CKBX) Name(#CKBX_SECT_CHK)

Caption('Check boxes') Displayposition(1) Left(16) Parent(#GPBX_SECT)

Tabposition(1) Top(16) Width(97)

Define_Com Class(#PRIM_CKBX) Name(#CKBX_EMPNO_CHK)

Caption('Check boxes') Displayposition(1) Left(16) Parent(#GPBX_EMPNO)

Tabposition(1) Top(16) Width(97)

* Items Tab

Define_Com Class(#PRIM_TBESH) Name(#PAGE_ITEMS) Caption('Items')

Displayposition(2) Height(239) Left(4) Parent(#SHEET) Tabposition(2)

Tabstop(False) Top(22) Width(385)

Define_Com Class(#PRIM_CKBX) Name(#CKBX_ITEM_CHECKBOX)

Caption('Check boxes') Displayposition(3) Left(16) Parent(#PAGE_ITEMS)

Tabposition(3) Top(32) Width(97)

Define_Com Class(#PRIM_CKBX) Name(#CKBX_ITEM_CHECKED)

Caption('Checked') Displayposition(2) Left(16) Parent(#PAGE_ITEMS)

Tabposition(2) Top(56) Width(97)

Define_Com Class(#PRIM_CKBX) Name(#CKBX_ITEM_CHKENABLED)

Caption('Check box enabled') Displayposition(1) Left(16)

Parent(#PAGE_ITEMS) Tabposition(1) Top(80) Width(201)

Define_Com Class(#PRIM_LABL) Name(#LABL_1) Caption('Clear and

reload the tree to set check boxes for columns.') Displayposition(4) Height(15)

Left(8) Parent(#PAGE_COL) Tabposition(4) Tabstop(False) Top(8)

Width(361)

Define_Com Class(#PRIM_LABL) Name(#LABL_2) Caption('Set check box

options for individual items in the tree.}') Displayposition(4) Height(15)
Left(16) Parent(#PAGE_ITEMS) Tabposition(4) Tabstop(False) Top(16)
Width(329)

Evroutine Handling(#CLEAR.Click)

Clr_List Named(#TREE)

Endroutine

Evroutine Handling(#LOAD.Click)

* Load the Tree View

Select Fields(#TREE) From_File(PSLMST)

Add_Entry To_List(#TREE)

Endselect

Endroutine

*

=====

*

* Form Initialization

*

*

=====

Evroutine Handling(#com_owner.CreateInstance)

Set Com(#com_owner) Caption(*component_desc)

* Load the Tree View

Select Fields(#TREE) From_File(PSLMST)

Add_Entry To_List(#TREE)

Endselect

Endroutine

*

=====

*

* Handlers

*

*

=====

Evroutine Handling(#CKBX_DEPT_CHK.Click)

If Cond(#CKBX_DEPT_CHK.buttonstate = Checked')

Set Com(#COL1) Checkboxes(True)

Else

Set Com(#COL1) Checkboxes(False)

Endif

```

Endroutine
Evroutine Handling(#CKBX_SECT_CHK.Click)
If Cond('#CKBX_SECT_CHK.buttonstate = Checked')
Set Com(#COL2) Checkboxes(True)
Else
Set Com(#COL2) Checkboxes(False)
Endif
Endroutine
Evroutine Handling(#CKBX_EMPNO_CHK.Click)
If Cond('#CKBX_EMPNO_CHK.buttonstate = Checked')
Set Com(#COL3) Checkboxes(True)
Else
Set Com(#COL3) Checkboxes(False)
Endif
Endroutine
Evroutine Handling(#CKBX_ITEM_CHECKBOX.Click)
If_Ref Com(#TREE.CurrentItem) Is_Not(*NULL)
If Cond('#CKBX_ITEM_CHECKBOX.buttonstate = Checked')
Set Com(#TREE.currentitem) Checkboxes(True)
Else
Set Com(#TREE.CURRENTITEM) Checkboxes(False)
Endif
Endif
Endroutine
Evroutine Handling(#CKBX_ITEM_CHKENABLED.Click)
If_Ref Com(#TREE.CurrentItem) Is_Not(*NULL)
If Cond('#CKBX_ITEM_CHKENABLED.buttonstate = Checked')
Set Com(#TREE.currentitem) Checkenabled(True)
Else
Set Com(#TREE.CURRENTITEM) Checkenabled(False)
Endif
Endif
Endroutine
Evroutine Handling(#CKBX_ITEM_CHECKED.Click)
If_Ref Com(#TREE.CurrentItem) Is_Not(*NULL)
If Cond('#CKBX_ITEM_CHECKED.buttonstate = Checked')
Set Com(#TREE.currentitem) Checked(True)
Else
Set Com(#TREE.CURRENTITEM) Checked(False)

```

Endif
Endif
Endroutine
End_Com

ColumnAlign プロパティ

ColumnAlignプロパティを利用して、カラムのコンテンツを左、右、中央揃えのいずれにするかを指定します。

このプロパティは、常に左に揃えられるリスト・ビューの最初のカラムには影響を与えませんので注意してください。

ColumnRole プロパティ

ColumnRole プロパティを使用して、カラムの役割を指定します。

グラフではカラムはラベルまたはデータ・カラムとして使用できます。テキスト・フィールドではラベルとして、数値フィールドではグラフ化されるデータとして使用されます。

円グラフはラベルとデータの2カラムのデータしか表示できません。棒グラフと折れ線グラフには複数のデータ・カラムが存在可能ですが、ラベル・カラムは1つだけです。

複数行編集ボックスでは、カラムはデータ、または行番号や行継続番号を含むカラムとして使用できます。

フィールドが行より短いために行を分割しなければならない時に、行継続番号として定義されたカラムを使用して行を作成する文字列の順序を記録します。

データ・カラム以外のカラムは非表示にする、もしくはデータ・カラムの後ろに置いて、ユーザーがデータ・カラム以外のカラムにテキストを入力しないようにします。

DisplayAppearance プロパティ

DisplayAppearanceはカラムがどのように表示されるかを制御します。

DisplayAppearance プロパティを利用して、カラム内の項目をどのように表示するかを決定します。

このプロパティには次のいずれかの値を設定できます：

Checkboxes：カラム内の項目の前にチェックボックスが付きます。

Edit：カラム内の項目は編集ボックスとして表示されます。

MultilineEdit：カラム内の項目は複数行編集ボックスとして表示されま

Image：カラムの項目はイメージになります。

ImageAndText：カラム内の項目はイメージとキャプションになります。

Default：この項目のDisplayAppearanceは、このカラムの基本となっているフィールドのコンポーネント定義内のビジュアルライゼーションによって自動的に決定されます。

フィールドのビジュアルライゼーションとの関係は、例えば次のようになります：

- Appearance(CheckBox)のピックアップリストの場合、DefaultはCheckboxesになります。
- Appearance(Images)のピックアップリストの場合、DefaultはImageになります。
- Appearance(ImagesAndText)のピックアップリストの場合、DefaultはImageAndTextになります。
- Appearance(DropDown)のピックアップリストの場合、DefaultはComboBoxになります。

DisplayAppearanceのCheckBoxes、Image、ImageAndTextは、カラムの基本となるフィールドのコンポーネント定義にピックアップリストが含まれる時のみ使用できることに注意してください。

リスト・ビューでは、ViewStyleがReportの時にだけDisplayAppearanceが適用されます。

DisplayAppearanceはツリー・ビューのカラムには適用されません。

DisplayPosition プロパティ

リスト、ツリー・ビュー、グリッド・カラムのDisplayPositionを使用して、その中のカラムの表示の順番を指定します。

VisibleプロパティがFalseに設定されているカラムは、DisplayPositionが0になることに注意してください。

EditAppearance プロパティ

EditAppearanceはカラムがどのように表示されるかを制御します。

EditAppearance プロパティを利用して、編集時にカラム内の項目がどのように表示されるかを決定します。

項目の編集を許可するには、カラムのReadOnly プロパティをFalseに設定しておかないといけないことに注意してください。

このプロパティには次のいずれかの値を設定できます：

Checkboxes：編集されるカラム内の項目の前にチェックボックスが付きます。

Default：この項目のEditAppearanceは、このカラムの基本となっているフィールドのコンポーネント定義内のビジュアルライゼーションによって自動的に決定されます。

Edit：編集されるカラム内の項目が編集ボックスとして表示されます。

Image：編集されるカラム内の項目がイメージとして表示されます。

ImageAndText：編集される項目にイメージとキャプションが付きます。

Multiline edit：編集されるカラム内の項目が編集ボックスとして表示されます。（グリッドでのみ使用可能）

ReusablePart：再利用可能パーツが編集用の項目に表示されます。

EditorPart プロパティで再利用可能パーツ名を指定します。

SpinEdit：編集されるカラム内の項目がスピン編集ボックスとして表示されます。

フィールドのビジュアルライゼーションとの関係は、例えば次のようになります：

- SpinEditの場合、DefaultはSpinEditになります。
- Appearance(CheckBox)のピックリストの場合、DefaultはCheckboxesになります。
- Appearance(Images)のピックリストの場合、DefaultはImageになります。
- Appearance(ImagesAndText)のピックリストの場合、DefaultはImageAndTextになります。
- Appearance(DropDown)のピックリストの場合、DefaultはComboBoxになります。

DisplayAppearanceのCheckBoxes、Image、ImageAndTextは、カラムの基

本となるフィールドのコンポーネント定義にピックリストが含まれる時
のみ使用できることに注意してください。

リスト・ビューでは、ViewStyleがReportの時にだけEditAppearanceが適
用されます。

KeyPosition プロパティ

KeyPositionはリストの格納方法を制御します。

KeyPositionプロパティを利用して、このカラムがツリー・ビューのキーのどの部分に当たるかを指定します。ツリー・ビュー・キーはツリー・ビューの特定のレベルに固有のもので、一番低い階層のレベルでない限り、各レベルに少なくとも1つのキーがなければなりません。

ツリー・ビュー・キーはツリー・ビュー内のエントリーが一意であることを確実にするために使用されます。典型的なフィールドとしては、社員番号、製品番号、受注番号などがキーとして使用されます。

フィールドをツリー・ビュー・キーの一部にしない場合は、0を指定します。

キーとして使用する唯一のフィールドの場合は1を指定します。

複数のフィールドをキーとして使用する時は、その相対的な順序を指定します。

Editor プロパティ

EditorによりEditorManagerコンポーネントにアクセスできるようになります。

Editorプロパティを使って、カラムの編集を管理するEditorManagerコンポーネントにアクセスできるようになります。

設計時にカラムの編集プロパティ（DisplayAppearanceやEditAppearanceなど）を直接変更することができますが、実行時はこのプロパティを変更するにはEditorManagerを使用しなければなりません。

カラムに設定されたプロパティにより、実行時にEditorManagerの省略値が提供されます。

EditorManagerのヘルプを表示させるには、左側のツリーのEditorプロパティでダブル・クリックします。

MinimumWidth プロパティ

MinimumWidthプロパティを使用して、カラムの最小幅を設定します。このプロパティを0に設定すると、ユーザーはカラムのサイズ調整をして見えないようにすることができます。

WidthTypeがCharacterの時、この値は文字数になります。その他のWidthTypesでは、この値はパーセントで表示されます。

ValueAt プロパティ

ListBox、GridView、ComboBox、Graphsなどのカラム・ビューやUnleveledのTreeViewにはValueAtプロパティがあり、取り出す文字列を表示することができます。

行を参照することで、カラム・ビューから値を読みだすことができます。

RESULT

リスト要素のバリエーション値です。

ROW

アクセスする行の整数インデックス

例

```
#Value := #TreeV1.columns<1>.ValueAt( 3 )
```

リスト・タイプのコンポーネントのイベント

GotFocus イベント

GotFocusAccept イベント

GotSelection イベント

GotSelectionAccept イベント

LostFocus イベント

LostFocusAccept イベント

LostSelection イベント

LostSelectionAccept イベント

GotFocus イベント

GotFocus イベントは、このカラム内の項目にフォーカスが移動した時に起動します。カラム内の項目間でのフォーカスの移動の場合、このイベントは起動しません。

GotFocusAccept イベント

GotFocusAcceptイベントはこのカラム内の項目がフォーカスを得た時に起動されます。

GotFocusAcceptイベントがGotFocusイベントと違うのは、このイベントを続行するか否かの制御ができることです。このイベントのAcceptパラメータをFalseに設定すると、イベントは実行されません。

例えば、グリッド・セルにフォーカスがあり、フォーカスを別のセルに移動すると、次のようなイベントが起動します。

- 最初のセルのItemLostFocusAcceptとItemLostSelectionAccept イベント
新しいセルが別の列にある場合は、最初の列にLostFocusAccept とLostSelectionAcceptイベント
- 2番目のセルのItem GotFocusAcceptとItemGotSelectionAccept イベント
新しいセルが別の列にある場合は、2番目のの列にGotFocusAccept とGotSelectionAcceptイベント

これらのイベントのいずれかのイベントのAcceptにFalseが設定されていると、操作はキャンセルされます。すべてのイベントのAcceptにTrueが設定されている場合は、次のイベントが実行されます。

- 最初のセルのItemLostFocusとItemLostSelection、同時に最初の列のLostFocusとLostSelection
- 2番目のセルのItemGotFocusとItemGotSelection、同時に2番目の列のGotFocusとGotSelection

Accept処理の目的は、ユーザーが行うことに対して、プログラマーによる制御を更に大きくすることです。何かが変更されようとする時は、その処理を続ける前に変更を受け入れるかどうか常にプログラマーに確認されます。

GotSelection イベント

GotSelectionイベントは、このカラム内の項目に選択が移動した時に起動します。カラム内の項目間での選択の移動ではこのイベントは起動されません。

GotSelectionAccept イベント

GotSelectionAcceptイベントはこのカラム内の項目が選択された時に起動されます。

GotSelectionAcceptイベントがGotSelectionイベントと違うのは、このイベントを続行するか否かの制御ができることです。このイベントのAcceptパラメータをFalseに設定すると、イベントは実行されません。

例えば、グリッド・セルにフォーカスがあり、フォーカスを別のセルに移動すると、次のようなイベントが起動します。

- 最初のセルのItemLostFocusAcceptとItemLostSelectionAccept イベント
新しいセルが別の列にある場合は、最初の列にLostFocusAccept とLostSelectionAcceptイベント
- 2番目のセルのItem GotFocusAcceptとItemGotSelectionAccept イベント
新しいセルが別の列にある場合は、2番目のの列にGotFocusAccept とGotSelectionAcceptイベント

これらのイベントのいずれかのイベントのAcceptにFalseが設定されていると、操作はキャンセルされます。すべてのイベントのAcceptにTrueが設定されている場合は、次のイベントが実行されます。

- 最初のセルのItemLostFocusとItemLostSelection、同時に最初の列のLostFocusとLostSelection
- 2番目のセルのItemGotFocusとItemGotSelection、同時に2番目の列のGotFocusとGotSelection

Accept処理の目的は、ユーザーが行うことに対して、プログラマーによる制御を更に大きくすることです。何かを変更されようとする時は、その処理を続ける前に変更を受け入れるかどうか常にプログラマーに確認されます。

LostFocus イベント

LostFocus イベントは、このカラム内の項目のフォーカスが失われた時に起動します。カラム内の項目間でのフォーカスの移動ではこのイベントは起動されません。

LostFocusAccept イベント

LostFocusAcceptイベントはこのカラム内の項目のフォーカスがなくなった時に起動されます。

LostFocusAcceptイベントがLostFocusイベントと違うのは、このイベントを続行するか否かの制御ができることです。このイベントのAcceptパラメータをFalseに設定すると、イベントは実行されません。

例えば、グリッド・セルにフォーカスがあり、フォーカスを別のセルに移動すると、次のようなイベントが起動します。

- 最初のセルのItemLostFocusAcceptとItemLostSelectionAccept イベント
新しいセルが別の列にある場合は、最初の列にLostFocusAccept とLostSelectionAcceptイベント
- 2番目のセルのItem GotFocusAcceptとItemGotSelectionAccept イベント
新しいセルが別の列にある場合は、2番目のの列にGotFocusAccept とGotSelectionAcceptイベント

これらのイベントのいずれかのイベントのAcceptにFalseが設定されていると、操作はキャンセルされます。すべてのイベントのAcceptにTrueが設定されている場合は、次のイベントが実行されます。

- 最初のセルのItemLostFocusとItemLostSelection、同時に最初の列のLostFocusとLostSelection
- 2番目のセルのItemGotFocusとItemGotSelection、同時に2番目の列のGotFocusとGotSelection

Accept処理の目的は、ユーザーが行うことに対して、プログラマーによる制御を更に大きくすることです。何かに変更されようとする時は、その処理を続ける前に変更を受け入れるかどうか常にプログラマーに確認されます。

LostSelection イベント

LostSelectionイベントは、このカラム内の項目の選択が失われた時に起動します。カラム内の項目間での選択の移動ではこのイベントは起動されません。

LostSelectionAccept イベント

LostSelectionAcceptは項目が選択を失う時に起動します。

LostFocusAcceptイベントはこのカラム内の項目のフォーカスがなくなった時に起動されます。

LostSelectionAcceptイベントがLostSelectionイベントと違うのは、このイベントを続行するか否かの制御ができることです。このイベントのAcceptパラメータをFalseに設定すると、イベントは実行されません。

例えば、グリッド・セルにフォーカスがあり、フォーカスを別のセルに移動すると、次のようなイベントが起動します。

- 最初のセルのItemLostFocusAcceptとItemLostSelectionAccept イベント
新しいセルが別の列にある場合は、最初の列にLostFocusAccept とLostSelectionAcceptイベント
- 2番目のセルのItemGotFocusAcceptとItemGotSelectionAccept イベント
新しいセルが別の列にある場合は、2番目の列にGotFocusAccept とGotSelectionAcceptイベント

これらのイベントのいずれかのイベントのAcceptにFalseが設定されていると、操作はキャンセルされます。すべてのイベントのAcceptにTrueが設定されている場合は、次のイベントが実行されます。

- 最初のセルのItemLostFocusとItemLostSelection、同時に最初の列のLostFocusとLostSelection
- 2番目のセルのItemGotFocusとItemGotSelection、同時に2番目の列のGotFocusとGotSelection

Accept処理の目的は、ユーザーが行うことに対して、プログラマーによる制御を更に大きくすることです。何かに変更されようとする時は、その処理を続ける前に変更を受け入れるかどうか常にプログラマーに確認されます。

リスト・タイプのコンポーネントのメソッド

SetValue メソッド

SetValue メソッド

ListBox、GridView、ComboBox、Graphsなどのカラム・ビューやUnleveledのTreeViewsにはValueAtプロパティがあり、取り出す文字列を表示することができます。

行と列を参照することで、カラム・ビューに値を挿入できます。

RESULT

設定が成功(TRUE)したかどうかを示すブール値です。

ROW

設定する行の整数インデックス

VALUE

設定する値

例

```
#TreeV1.columns<1>.SetValue ( 3 "tv1" )
```

```
#Grid1.columns<1>.SetValue ( 3 "31" )
```

EditorManager コンポーネント

EditorManagerは実行時のカラムの編集を管理します。

EditorManagerコンポーネントを使用して、実行時にカラムのプロパティを変更できます。

設計時にカラムの編集プロパティ（DisplayAppearanceやEditAppearanceなど）を直接変更することができますが、実行時はこのプロパティを変更するにはEditorManagerを使用しなければなりません。

設計時にカラムに設定されたプロパティにより、EditorManagerの省略値が提供されます。

[EditorManagerコンポーネントのプロパティ](#)

[EditorManagerコンポーネントのメソッド](#)

EditorManagerコンポーネントのプロパティ

EditorPart プロパティ

ShowDropDown プロパティ

ShowPrompt プロパティ

EditorPart プロパティ

EditorPartはカラムの編集に使用される再利用可能パーツです。

EditorPart プロパティを使ってカラムの編集に使用する再利用可能パーツの名前を指定します。

通常グリッド・カラムに値を編集する適切な機能がない場合にエディター部分を作成します。

例えば、別のカラムやコンポーネントの値によって変更するコンテンツの値のドロップ・ダウンの場合、ピックリストの内容は動的に変更できないのでフィールド・ビジュアライゼーションを使うことができません。このような場合にコンボ・ボックスとその内容を動的に変更できるメソッド・ルーチンを含む再利用可能パーツを作成します。

EditAppearance プロパティをReusablePartに設定して、編集部分を表示します。

ShowDropDown プロパティ

ShowDropDownはドロップダウン・リストを表示するかどうかを制御します。

ShowDropDownを使って、カラムのビジュアルイゼーションの省略値を上書きします。

LANSALはカラムのビジュアルイゼーションの省略値を元のフィールドのコンポーネント定義に存在するビジュアルイゼーションに基づいて決定します。

ShowDropDownプロパティにTrueを設定することで、このビジュアルイゼーションをドロップダウン・リストに変更できます。

ShowPrompt プロパティ

ShowPromptはプロンプトを表示するかどうかを制御します。

ShowPromptを使って、カラムのビジュアルライゼーションの省略値を上書きします。

LANSAsはカラムのビジュアルライゼーションの省略値を元のフィールドのコンポーネント定義に存在するビジュアルライゼーションに基づいて決定します。

ShowPromptプロパティにTrueを設定することで、このビジュアルライゼーションをプロンプトに変更できます。

EditorManagerコンポーネントのメソッド

Prompt メソッド

DisplayPosition プロパティ

MinimumWidth プロパティ

Prompt メソッド

Promptメソッドを使用して現在のセル・エディターのためのプロンプトを表示します。

このメソッドを起動させるのは、ユーザーがF4を押した時と同じです。

基本リスト項目

リスト項目はリスト・ボックス、リスト・ビュー、コンボ・ボックス、ツリー・ビューまたはグリッドのエントリーです。

リスト項目は実行時のコンポーネントです。リスト、コンボ・ボックス、ツリー・ビューやグリッドのCurrentItemもしくはFocusItemプロパティを使ってリスト項目を操作することができます。

CurrentItemはアプリケーションで現在処理されている項目です。ユーザーはリストの項目を選択することでCurrentItemを設定できます。またはGET_ENTRYやSELECTLISTを使用してプログラム上でCurrentItemを設定することも可能です。

FocusItemはデスクトップでフォーカスを持つ項目です。ユーザーはリスト項目のフォーカスを移動させることでFocusItemを設定できます。またプログラムでCurrentItemのFocusプロパティにTrueを設定することでFocusItemを設定することもできます。

FocusItemとCurrentItemはたいていの場合は同じですが、異なる場合もあります。例えばアプリケーションがデスクトップのFocusItemを変更せずに、バックグラウンドでリスト項目を処理している場合などです。

プロパティ

Checkboxes プロパティ	CurrentItemのPosition プロパティ	Entry プロパティ
Checked プロパティ	CurrentItemのSelected プロパティ	FirstVisible プロパティ
CheckEnabled プロパティ	EnsureVisible プロパティ	RelatedReference プロパティ
CurrentItemのFocus プロパティ		

EnsureVisible プロパティ

リスト・ビュー、リスト・ボックス、コンボ・ボックス、ツリービューやグリッド内でリスト項目が表示されているか、グリッド内でグリッド・カラムが表示されているかをEnsureVisibleプロパティを使って確認します。

次のコードでは、コンボ・ボックス内の現在の項目が表示されることを確実にします。

```
Set #Cmbx_1.CurrentItem EnsureVisible(True)
```

Checked プロパティ

Checkedプロパティを使って項目の前のチェックボックスを選択するかどうかを制御したり、選択の有無を確認したりします。このCheckedプロパティにはTrue、GrayedまたはFalseが設定できます。

Checkboxes プロパティ

Checkboxes プロパティを使用して、項目のチェックボックス表示のオン・オフを切り替えます。

またカラムのCheckboxes プロパティを使ってカラム内の全ての項目のチェックボックスの表示を制御することもできます。

次のイベント・ルーチン例では、項目が選択されていない時にチェックボックスが非表示になります。

```
Evtroutine Handling(#CKBX_ITEM_CHECKBOX.Click)
If_Ref Com(#TREE.CurrentItem) Is_Not(*NULL)
If Cond('#CKBX_ITEM_CHECKBOX.buttonstate = Checked')
Set Com(#TREE.currentitem) Checkboxes(True)
Else
Set Com(#TREE.CURRENTITEM) Checkboxes(False)
Endif
Endif
Endroutine
```

Entry プロパティ

読み取り専用プロパティです。

Entry プロパティを使用してリスト内のリスト項目のエントリー番号を探しだします。

例えば他の項目がリストに追加されたり、リストを保存する時などに、項目番号を記録して、経過を追うことが必要となります。次のコードではフィールドのリスト項目のエントリー番号を記録します。

```
CHANGE FIELD(#eno) TO(#Ltw_1.CurrentItem.Entry)
```

リストはロード時は特定の項目への選択を設定しません。特定の項目の選択を設定するには、GET_ENTRY コマンドを使ってエントリー番号を指定し、選択を設定します。

```
Get_Entry Number(3) From_List(#Ltw_1)  
Set #Ltw_1.CurrentItem Selected(True)
```

FirstVisible プロパティ

FirstVisibleプロパティを使用して、項目がグリッドの最初の項目に位置されることを確実にします。

次のコードではグリッドの現在の項目がグリッド内の最初の項目として位置づけられることを確実にします。

```
Set #Grid_1.CurrentItem FirstVisible(True)
```

CurrentItemのFocus プロパティ

Focusプロパティを使って、リスト、グリッド、コンボ・ボックスやツリービューの項目やグリッド・カラムにフォーカスがあるかどうかを検知したり、項目やカラムにフォーカスを設定したりします。一度に1つの項目やカラムにだけFocusプロパティをTrueに設定できます。

SELECTLISTコマンドを使用してリストの各アイテムのプロパティを調べることができます。例えば、以下のコードは、一度に1つのアイテムをCurrentItemにし、Focusプロパティがtrueに設定されているかどうかをテストします。

```
SELECTLIST NAMED(#LTVW_1)
IF '#ltyw_1.CurrentItem.Focus *eq True'
do something
ENDIF
ENDSELECT
```

一度に複数のアイテムを選択できないリスト(ツリー表示、コンボ・ボックス、リスト・ボックス)では、FocusプロパティとSelectedプロパティは常に同じ値になります。つまり、リスト項目のFocusプロパティにTrueを設定すると、自動的にSelectedプロパティもTrueに設定されます。

複数選択可能なリストでは、1つのアイテムだけにフォーカスが置かれますが(FocusプロパティがTrueに設定)、複数のアイテムを選択できます(CtrlキーとShiftキーを使用)。

FocusItemの場合、Focusプロパティは常にTrueに設定されます。

CurrentItemのPosition プロパティ

リストまたはグリッド項目のPositionプロパティを使って、リストまたはグリッドのどの位置が表示されているかを探しだします。

次のコードでは、フィールド#STD_NUMのグリッドの項目の位置を示します。このコードをコピーしてフォームに貼り付けることもできます。

```
FUNCTION options(*DIRECT)
BEGIN_COM role(*EXTENDS #PRIM_FORM) HEIGHT(289) LEFT(346) T
DEFINE_COM class(#PRIM_GRID) name(#GRID_1) COLUMNBUTTONH
DEFINE_COM class(#PRIM_GDCL) name(#GDCL_1) CAPTIONALIGN(Lc
DEFINE_COM class(#PRIM_GDCL) name(#GDCL_2) CAPTIONALIGN(Lc
DEFINE_COM class(#STD_NUM.Visual) name(#STD_NUM) DISPLAYPOS
EVTROUTINE handling(#COM_OWNER.Initialize) options(*NOCLEARME
SELECT fields(#GRID_1) from_file(PSLMST)
ADD_ENTRY to_list(#GRID_1)
ENDSELECT
ENDROUTINE
EVTROUTINE handling(#GRID_1.ItemGotFocus) options(*NOCLEARMES
CHANGE field(#STD_NUM) to('#GRID_1.CURRENTITEM.POSITION')
ENDROUTINE
END_COM
```

RelatedReference プロパティ

RelatedReference プロパティを使用してリストの項目に対するオブジェクトを保存します。これにより、リスト項目からユーザー定義のオブジェクトに移動することが可能になります。

次の例は社員情報を表示する再利用可能パーツとこのパーツを使ってグリッドの現在の項目の詳細を表示するフォームとから成り立っています。このコードをコピーして貼り付けることもできます。

Reusable part GRID_REF:

```
Function Options(*DIRECT)
```

```
Begin_Com Role(*EXTENDS #PRIM_PANL) Displayposition(1) Left(0) Tab
```

```
Define_Pty Name(p_Employee) Get(*auto #empno) Set(*auto #empno)
```

```
Define_Pty Name(p_Surname) Get(*auto #Surname) Set(*auto #Surname)
```

```
Define_Pty Name(p_Givename) Get(*auto #Givename) Set(*auto #Givename)
```

```
Mthroutine Name(Show_details)
```

```
Use Builtin(MESSAGE_BOX_ADD) With_Args(#EMPNO)
```

```
Use Builtin(MESSAGE_BOX_ADD) With_Args(#GIVENAME)
```

```
Use Builtin(MESSAGE_BOX_ADD) With_Args(#SURNAME)
```

```
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK INFORMATION)
```

```
Endroutine
```

```
End_Com
```

The form:

```
Function Options(*DIRECT)
```

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Height(422) Left(310) Top(10
```

```
* Grid Definition
```

```
Define_Com Class(#PRIM_GRID) Name(#GRID_1) Captionnoblanklines(Tru
```

```
Define_Com Class(#PRIM_GDCL) Name(#GDCL_1) Displayposition(1) Pare
```

```
Define_Com Class(#PRIM_GDCL) Name(#GDCL_2) Displayposition(2) Pare
```

```
Define_Com Class(#PRIM_GDCL) Name(#GDCL_3) Displayposition(3) Pare
```

```
Define_Com Class(#GRID_REF) Name(#temp_employee) Reference(*dynam
```

```
Define_Com Class(#PRIM_STBR) Name(#STBR_1) Displayposition(2) Heig
```

```
Evtoutine Handling(#com_owner.CreateInstance)
```

```
Set Com(#com_owner) Caption(*component_desc)
```

```
Select Fields(#GRID_1) From_File(PSLMST)
```

```
Set_Ref Com(#temp_employee) To(*create_as #GRID_REF)
```

```
Set Com(#temp_employee) P_Employee(#empno) P_Surname(#surname) P_C
```

```
Add_Entry To_List(#GRID_1)
```

```
Set Com(#grid_1.currentitem) Relatedreference(#temp_employee)
```

```
Set_Ref Com(#temp_employee) To(*null)
Endselect
Endroutine
Evtroutine Handling(#GRID_1.DoubleClick) Options(*NOCLEARMESSAGE
Set_Ref Com(#Temp_employee) To(*dynamic #grid_1.currentitem.relatedrefe
Invoke Method(#Temp_employee.Show_details)
Endroutine End_Com
```

CurrentItemのSelected プロパティ

Selectedプロパティを使って、リスト、グリッド、コンボ・ボックスやツリービューの項目やグリッド・カラムが選択されているかどうかを検知したり、項目やカラムに選択を設定したりします。

SELECTLISTコマンドを使用してリストの各アイテムのプロパティを調べることができます。例えば、以下のコードは、1度に1つのアイテムをCurrentItemにし、Selectedプロパティがtrueに設定されているかどうかをテストします。

```
SELECTLIST NAMED(#LTVW_1)
IF '#ltvw_1.CurrentItem.Selected *eq True'
do something
ENDIF
ENDSELECT
```

複数選択が可能なリストでは、プログラム上でまたはユーザーがCtrlとShiftキーを押すことで複数の項目の選択ができます。

次のコードではエントリー番号10から15までのリストビュー項目の選択を設定します。リストビューのSelectionModeはMultipleに設定されています。

```
SELECTLIST NAMED(#LTVW_1)
IF cond('(#ltvw_1.CurrentItem.entry *ge 10) *and (#ltvw_1.CurrentItem.entry
Set #Ltvw_1.CurrentItem Selected(True)
ENDIF
ENDSELECT
```

CheckEnabled プロパティ

CheckEnabled プロパティを使用してリスト項目のチェックボックスが有効かどうかを制御します。

次のコードはリストビューの現在の項目のチェックボックスを無効にします。

```
Set Com(#ltvw_1.CurrentItem) CheckEnabled(False)
```

基本バー

基本バーは表示されません。このコンポーネントはプログレス・バーとトラック・バーの共通のプロパティ/イベント/メソッドのための1つのソースを提供します。

[MinimumValue](#) プロパティ

[MaximumValue](#) プロパティ

[Value](#) プロパティ

MinimumValue プロパティ

MinimumValueはプログレス・バーとトラック・バーの目盛りの最初の値を設定します。 MaximumValueと共に使用して目盛りを設定します。

トラック・バーでは、MinimumValueプロパティが最初の目盛りを設定し、MaximumValueプロパティが最後の目盛りを設定します。

MinimumValueとMaximumValueは符号付きの2バイトの範囲内 (MIN=-32767 MAX=32767) でなければなりません。

プログレス・バーでは、MinimumValueとMaximumValueプロパティは処理全体の時間の範囲を設定します。

MaximumValue プロパティ

MaximumValueはプログレス・バーとトラック・バーの目盛りの最後の値を設定します。 MinimumValueと共に使用して目盛りを設定します。

トラック・バーでは、MinimumValueプロパティが最初の目盛りを設定し、MaximumValueプロパティが最後の目盛りを設定します。

MinimumValueとMaximumValueは符号付きの2バイトの範囲内 (MIN=-32767 MAX=32767) でなければなりません。

プログレス・バーでは、MinimumValueとMaximumValueプロパティは処理全体の長さの範囲を設定します。

Value プロパティ

Valueはトラック・バーのスライダーの位置を設定したり、戻り値として返したりします。

プログレス・バーでは、ある処理の完了までの進捗を示す値を設定したり、戻したりします。値を増やしてもその値の分だけプログレス・バーの表示が変更されるわけではないことに注意してください。

値は常にMinimum（最小値）とMaximum（最大値）の間の範囲内です。

プログレス・バー

プログレス・バーを使用して長い処理の進捗状況を表示します。プログレス・バーは処理が進行するにつれ、左側から右側に徐々に埋められていきます。

MinimumValueとMaximumValueプロパティを使って処理全体の長さの範囲を設定します。Valueプロパティはプログレス・バーの現在の値を示します。

例えば10個のファイル进行处理する場合、MinimumValueを0に、MaximumValueを10に設定します。そしてファイルが処理されるごとに値を1ずつ増やしていき、最後のファイルが処理されるとバーが埋められるようにします。

次のコードでは始めにプログレス・バーを表示し、最初のインジケータ位置を1に設定します。そして#loopcount変数を0に初期化します。ループでプログレス・バーの値はループ・カウンターの現在の値に設定されます。最後にプログレス・バーが非表示になります。

```
set #progbar value(1) visible(true)
change #loopcount 1
begin_loop using(#loopcount) to(10)
set #progbar value(#loopcount)
end_loop
set #progbar visible(false)
```

トラック・バー

トラック・バーはスライダーと目盛り（オプション）が付いたウィンドウです。離散値やある範囲内の連続値を選択したい場合に便利です。

ユーザーはスライダーをドラッグする、マウスでスライダーの片側をクリックする、またはキーボードを使って移動させることができます。

トラック・バーの値を10に設定するには、次のようにします。

```
set #trackbar value(10)
```

トラック・バーに異なる値を入れるコードを書くには、トラック・バー値を保存するフィールド #value を定義して、次のようなCASE命令文を書きます。

```
change #value #Tkbr_1.value
case #value
when '= 1'
do something
when '= 2'
do something else
when '= 3'
do something else
endcase
```

LineStyle プロパティ

Orientation プロパティ

PageSize プロパティ

TickFrequency プロパティ

TickMark プロパティ

TickStyle プロパティ

LineStyle プロパティ

LineStyleはキーボードの矢印キーが押された時に、トラック・バーのスライダーが移動する増分を指定します。

省略値のライン・サイズは1です。

Orientation プロパティ

Orientationプロパティはバー位置を垂直にするか水平にするかを制御します。

PageSize プロパティ

PageSizeはPage UpとPage Downキーが押される、またはトラック・バーがマウスでクリックされた時にトラック・バーのスライダーが移動する増分を指定します。

省略値のページ・サイズは5です。

TickFrequency プロパティ

TickFrequencyは、MinimumValueとMaximumValueの値の範囲に対するトラック・バーの目盛りの度数を制御します。例えば、範囲が0から10の時にこのプロパティが2に設定されたとすると、この範囲内で2ごとに1つの目盛りが表示されます。

TickMark プロパティ

TickMarkはトラック・バーのどのサイドに目盛りを付けるかを制御します。

このプロパティは、次の値を取ることができます：

BottomRight: 水平方向トラック・バーの場合は下に沿って、垂直方向の場合は右側に沿って目盛が付きます。

TopLeft: 水平方向トラック・バーの場合は上側に沿って、垂直方向の場合は左側に沿って目盛が付きます。

Both: トラック・バーの上下両方にメモリが付きます。

TickStyle プロパティ

TickStyleはトラック・バーの目盛りをどのように表示するかを制御します。

このプロパティは、次の値を取ることができます：

Auto: MinimumValue、MaximumValueおよびTickFrequencyプロパティに指定されたとおりにトラック・バーの目盛りが表示されます。

Manual: 現在は使用できません。

None: トラック・バーに目盛りが表示されません。

グループ・ボックス

グループ・ボックスにより、識別可能なコンポーネントのグループ化ができます。

グループ・ボックスを使って、例えばフィールドのグループを分けるなど、フォームを細かく分割して表示できます。また、例えば互いに排他的なラジオ・ボタンなどの機能にもこれを使用することができます。

イメージ

イメージ・コンポーネントを使用して画像ファイルを表示します。表示する画像を指定する方法は2通りあります。Imageプロパティを使用する方法とFilenameプロパティを使用する方法です。Imageプロパティはリポジトリに登録されたビットマップを指定する場合に使用し、Filenameプロパティは次のいずれかの形式の画像ファイルを指定する場合に使用します。

.bmp

.gif (also animated gif)

.jpeg

.tga

.pcx

.sgi

.tif

イメージ・コントロールを使って、会社のロゴや多くのアプリケーションで使用される標準画像などのめったに変更することのない画像を表示する時に、Imageプロパティを使用します。このような画像はリポジトリに保存して、後に検索して更新しやすいようにしておく方が便利です。

アプリケーションで画像がデータの一部として処理される場合は、FileNameプロパティを使用します。

[AutoSize プロパティ](#)

[Format プロパティ](#)

[FileName プロパティ](#)

AutoSize プロパティ

AutoSize プロパティを使って、イメージ・コントロールのサイズに画像を調整するかどうかを決定します。

ツールバー・ボタンにテキストと画像の両方を追加する場合はAutoSize をfalseに指定する必要があります。

Format プロパティ

Formatは画像で使用するファイル・タイプを指定します。このプロパティの値としては、GIF、JPEG、PCX、TGA、BMP、SGIが使用できません。

FileName プロパティ

FileName プロパティを使って、画像を含むファイルの名前を指定します。

指定する画像は設計時に表示されます。ただし、コンポーネントを保存する時はFileName プロパティが空白になり、画像はそれ以降表示されません。これはイメージ・コンポーネントはプログラムで使用して画像を表示するように設計されているからです。

例えば次のコードはEMPNOフィールドの値に応じて画像を設定します。

```
EvtRoutine Handling(#Form_12.Initialize)
Case Of_Field(#EMPNO)
When Value_Is(= A1234')
Set Com(#imge_1) Filename('C:\My Documents\pictA1234.bmp')
...
```

ImageHeight プロパティ

ImageHeight プロパティは画像の元の高さを示します。

ImageWidth プロパティ

ImageWidthプロパティは画像の元の幅を示します。

基本ビデオ

表示されません。

FileName プロパティ

ImageHeight プロパティ

ImageWidth プロパティ

FileName プロパティ

FileName プロパティを使って、.avi ファイルを含むファイルの名前を指定します。

ImageHeight プロパティ

ImageHeight プロパティは画像の元の高さを示します。

ImageWidth プロパティ

ImageWidthプロパティは画像の元の幅を示します。

ビデオ

ビデオを使って再生するマルチメディア・ファイルを指定します。

[VideoStopped イベント](#)

[ビデオ・パネル](#)

[FrameBackward メソッド](#)

[FrameForward メソッド](#)

[Play メソッド](#)

[Stop メソッド](#)

[CycleToStart プロパティ](#)

VideoStopped イベント

VideoStoppedイベントは、ビデオが停止されると起動されます。

ビデオ・パネル

ビデオ・パネルを使用して、ユーザーがマルチメディア・ファイルを巻き戻したり、再生したりできるようにします。概念としては、このコンポーネントはビデオ・レコーダーのボタンのように一連のボタンからなるコンポーネントです。

FrameBackward メソッド

FrameBackwardメソッドはビデオを1フレーム戻します。

FrameForward メソッド

FrameForwardメソッドはビデオを1フレーム進めます。

Play メソッド

Playメソッドはビデオを開始します。

Stop メソッド

Stopメソッドはビデオを停止します。

CycleToStart プロパティ

CycleToStartをTrueに設定すると、ビデオが終わると自動的に開始します。

サウンド

サウンド・コンポーネントを使用してサウンド・ファイルを再生します。

FileName プロパティ

FileName プロパティ

FileName プロパティを使って、サウンド・ファイルを含むファイルの名前を指定します。

ステータス・バー

ウィンドウの下の情報領域です。

ステータス・バーを使用してフォームの下のウィンドウを作成します。これを通してアプリケーションで様々なメッセージを表示することができます。

ステータス・バーには、情報や状態メッセージ（メッセージ・タイプが*INFOまたは*STATUSのもの）を表示できます。また、I/Oモジュールから発行される全ての検証メッセージがこのステータス・バーに自動的に表示されます。

情報メッセージは新しいイベント・ルーチンが起動されるまでステータス・バーに表示されます。複数の情報メッセージが発行された場合は、ユーザーはスクロールすることができます。

ステータス・メッセージは、処理が進行するにつれてステータス・バーに表示されます。メッセージを発行したイベント・ルーチンが終了すると、最後のメッセージは消去されます。

I/O検証メッセージは全て自動的にステータス・バーに表示されます。2次レベルのヘルプ・テキストを見るには、ステータス・バーに表示されたメッセージをクリックします。

ステータス・バーを使用する簡単な方法はVL_BBSTSBRTemplateを実行することです。（[編集]メニューで[ウィザード（テンプレート）の実行]オプションを選択します。）これにより、フォームにステータス・バーが置かれ、2つのサブ・ルーチンが作成されます。1つは情報メッセージ (infmessage)を送信し、もう1つはステータス・メッセージ (stsmmessage)を送信します。

単にドラッグ・アンド・ドロップするだけで、イメージやコンボ・ボックスなど他のコントロールをステータス・バーに取り込むことが可能です。例えば、アニメーション.gif 画像ファイルをステータス・バーに追加し、Visibleプロパティを使って制御することができます。

ステータス・バー内に別のコントロールがある場合、ステータス・バーのMessagePositionプロパティとその他のコントロールのDisplayPositionプロパティを使ってステータス・バー上に表示する順序を決定することができます。

例えば、ステータス・バーに2つのイメージ・コントロールを持つとします。これをImage_1、message area、Image_2の順に表示するには次のようになります。Image_1のDisplayPositionを1に、ステータス・バーの

MessagePositionを2、そしてImage_2のDisplayPositionを3に設定します。

LayoutStyle プロパティ

MessagePosition プロパティ

LayoutStyle プロパティ

LayoutStyleはステータス・バーのスタイルを設定します。

LayoutStyleプロパティを使ってステータス・バー上の複数の項目をどのように位置づけるかを制御します。LayoutStyleには以下のタイプがあります。

Attach - 項目はステータス・バーの両側を埋めるように位置付けられます。

Flow - 項目は右側から順に位置付けられます。

MessagePosition プロパティ

MessagePositionはメッセージがどの場所に表示されるかを決定します。ステータス・バー内に別のコントロールがある場合、ステータス・バーのMessagePositionプロパティとその他のコントロールのDisplayPositionプロパティを使ってステータス・バー上に表示する順序を決定することができます。

例えば、ステータス・バーに2つのイメージ・コントロールを持つとします。これをImage_1、message area、Image_2の順に表示するには次のようになります。Image_1のDisplayPositionを1に、ステータス・バーのMessagePositionを2、そしてImage_2のDisplayPositionを3に設定します。

タブ・フォルダー

タブ・フォルダーのプロパティ

タブ・フォルダーのイベント

複数のタブ・シートの付いたフォルダーです。

タブ・フォルダーを使って複数のシート付きのフォルダーを作成し、大量の情報を整理して表示する方法を提供します。まず始めにフォームにタブ・フォルダーを置き、そこに必要な数だけタブシート・コントロールをドラッグします。

タブ・フォルダーには外観を統制する基本プロパティとは別に、高度なドッキング、ドッキング解除、および自動非表示の機能があり、1つのタブ・フォルダーをウィンドウの様々なパーツに添付されたタブ・シート付きのタブ・ドッキング・アプリケーションにすることができます。このドッキング・アプリケーションのタブ・フォルダーを自動非表示 (autohide) にしたり、タブシートを画面のパーツに移動させたり、添付したりする (undock and dock) ことができます。

これらの高度な機能を使用する場合は、タブ・フォルダーをフォームの添付レイアウト・マネージャで管理される必要があります。タブ・フレームワークの構造は、各タブ・シートの DockPosition の設定で作成されます。

次のコードをコピー・貼り付けして、ドッキング・アプリケーションの基本を確認できます。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Sample IDE')
CLIENTHEIGHT(457) CLIENTWIDTH(652) HEIGHT(484)
LAYOUTMANAGER(#ATLM_1) LEFT(330) TOP(126) WIDTH(660)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
DISPLAYPOSITION(1) DRAGSTYLE(Automatic) DRAGTABS(True)
HEIGHT(457) LEFT(0) LEFTTABWIDTH(186) PARENT(#COM_OWNER)
TABPOSITION(1) TABSTOP(False) TOP(0) WIDTH(652)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_1)
CAPTION('Outliner') DISPLAYPOSITION(1)
DOCKALLOWEDPOSITIONS(Left+Bottom+Right)
DOCKALLOWUNDOCK(True) DOCKCLOSEBUTTON(True)
DOCKPOSITION(Left) HEIGHT(430) IMAGE(#STD_BTMAP) LEFT(4)
OPENED(True) PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False)
TOP(23) WIDTH(178)
```

```

DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_2)
CAPTION('Repository') DISPLAYPOSITION(2)
DOCKALLOWEDPOSITIONS(Left+Bottom+Right)
DOCKALLOWUNDOCK(True) DOCKCLOSEBUTTON(True)
DOCKPOSITION(Left) HEIGHT(430) IMAGE(#VB_LOCK)
LAYOUTMANAGER(#ATLM_1) LEFT(4) PARENT(#TAB_1)
TABPOSITION(2) TABSTOP(False) TOP(23) WIDTH(178)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_4)
CAPTION('Design') DISPLAYPOSITION(1) HEIGHT(431) LEFT(4)
OPENED(True) PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False)
TOP(22) WIDTH(453)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_3)
CAPTION('Source') DISPLAYPOSITION(2) HEIGHT(431)
LAYOUTMANAGER(#ATLM_1) LEFT(4) PARENT(#TAB_1)
TABPOSITION(2) TABSTOP(False) TOP(22) WIDTH(453)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_2)
DISPLAYPOSITION(1) HEIGHT(415) LEFT(0) PARENT(#TBESH_2)
TABLOCATION(Bottom) TABPOSITION(1) TABSTOP(False) TOP(15)
WIDTH(178)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_2) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_5)
CAPTION('Fields') DISPLAYPOSITION(1) HEIGHT(389)
LAYOUTMANAGER(#ATLM_1) LEFT(4) OPENED(True)
PARENT(#TAB_2) TABPOSITION(1) TABSTOP(False) TOP(4)
WIDTH(170)

DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_6)
CAPTION('Files') DISPLAYPOSITION(2) HEIGHT(389)
LAYOUTMANAGER(#ATLM_1) LEFT(4) PARENT(#TAB_2)
TABPOSITION(3) TABSTOP(False) TOP(4) WIDTH(170)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_7)
CAPTION('Forms') DISPLAYPOSITION(3) HEIGHT(389)
LAYOUTMANAGER(#ATLM_1) LEFT(4) PARENT(#TAB_2)
TABPOSITION(2) TABSTOP(False) TOP(4) WIDTH(170)
DEFINE_COM CLASS(#PRIM_LTVW) NAME(#FIELDS)
COMPONENTVERSION(2) DISPLAYPOSITION(1)
FULLROWSELECT(True) HEIGHT(389) LEFT(0) PARENT(#TBESH_5)

```

```
SHOWSORTARROW(True) TABPOSITION(1) TOP(0) WIDTH(170)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_2)
ATTACHMENT(Center) MANAGE(#FIELDS) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_LTVW) NAME(#FILES)
COMPONENTVERSION(2) DISPLAYPOSITION(1)
FULLROWSELECT(True) HEIGHT(389) LEFT(0) PARENT(#TBSH_6)
SHOWSORTARROW(True) TABPOSITION(1) TOP(0) WIDTH(170)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_3)
ATTACHMENT(Center) MANAGE(#FILES) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_LTVW) NAME(#FORMS)
COMPONENTVERSION(2) DISPLAYPOSITION(1)
FULLROWSELECT(True) HEIGHT(389) LEFT(0) PARENT(#TBSH_7)
SHOWSORTARROW(True) TABPOSITION(1) TOP(0) WIDTH(170)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_4)
ATTACHMENT(Center) MANAGE(#FORMS) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_1) CAPTION('Form
Name') CAPTIONTYPE(Caption) DISPLAYPOSITION(1)
PARENT(#FORMS) SOURCE(#STD_TEXT) WIDTH(99)
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_2) CAPTION('File
Name') CAPTIONTYPE(Caption) DISPLAYPOSITION(1)
PARENT(#FILES) SOURCE(#STD_TEXT) WIDTH(100)
DEFINE_COM CLASS(#PRIM_LVCL) NAME(#LVCL_3) CAPTION('Field
Name') CAPTIONTYPE(Caption) DISPLAYPOSITION(1)
PARENT(#FIELDS) SOURCE(#STD_TEXT) WIDTH(95)
DEFINE_COM CLASS(#prim_memo) NAME(#source) CURRENTLINE(1)
DISPLAYPOSITION(1) HEIGHT(431) LEFT(0)
MAXIMUMLINELENGTH(50) PARENT(#TBSH_3) TABPOSITION(1)
TOP(0) WIDTH(453)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_5)
ATTACHMENT(Center) MANAGE(#source) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_MECL) NAME(#MECL_1)
COLUMNROLE(Data) DISPLAYPOSITION(1) PARENT(#source)
SOURCE(#STD_TEXT)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_6)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
EvtRoutine Handling(#com_owner.Initialize)
#std_text := 'ADDRESS1'
Add_Entry_To_List(#FIELDS)
Set Com(#fields.CurrentItem) Image(#VI_FOLDOP)
```

```
#std_text := 'SALARY'  
Add_Entry To_List(#FIELDS)  
Set Com(#fields.CurrentItem) Image(#VI_FOLDOP)  
#std_text := 'STD_NUM'  
Add_Entry To_List(#FIELDS)  
Set Com(#fields.CurrentItem) Image(#VI_FOLDOP)  
#std_text := 'STD_TEXT'  
Add_Entry To_List(#FIELDS)  
Set Com(#fields.CurrentItem) Image(#VI_FOLDOP)  
#std_text := 'PSMLST'  
Add_Entry To_List(#files)  
Set Com(#files.CurrentItem) Image(#VI_EMPLOY)  
#std_text := 'DEPTMENT'  
Add_Entry To_List(#files)  
Set Com(#files.CurrentItem) Image(#VI_EMPLOY)  
#std_text := 'PSLSKL'  
Add_Entry To_List(#files)  
Set Com(#files.CurrentItem) Image(#VI_EMPLOY)  
#std_text := 'FORM1'  
Add_Entry To_List(#forms)  
Set Com(#forms.CurrentItem) Image(#STD_ICON)  
#std_text := 'FORM2'  
Add_Entry To_List(#forms)  
Set Com(#forms.CurrentItem) Image(#STD_ICON)  
#std_text := 'FORM3'  
Add_Entry To_List(#forms)  
Set Com(#forms.CurrentItem) Image(#STD_ICON)  
#std_text := 'FORM4'  
Add_Entry To_List(#forms)  
Set Com(#forms.CurrentItem) Image(#STD_ICON)  
#std_text := 'FORM5'  
Add_Entry To_List(#forms)  
Set Com(#forms.CurrentItem) Image(#STD_ICON)  
#std_text := 'FORM6'  
Add_Entry To_List(#forms)  
Set Com(#forms.CurrentItem) Image(#STD_ICON)  
#std_text := 'FORM7'  
Add_Entry To_List(#forms)  
Set Com(#forms.CurrentItem) Image(#STD_ICON)
```

```
#std_text := 'FORM8'
Add_Entry To_List(#forms)
Set Com(#forms.CurrentItem) Image(#STD_ICON)
#std_text := 'FORM9'
Add_Entry To_List(#forms)
Set Com(#forms.CurrentItem) Image(#STD_ICON)
#std_text := 'FORM10'
Add_Entry To_List(#forms)
Set Com(#forms.CurrentItem) Image(#STD_ICON)
#std_text := 'FORM11'
Add_Entry To_List(#forms)
Set Com(#forms.CurrentItem) Image(#STD_ICON)
#std_text := 'FORM12'
Add_Entry To_List(#forms)
Set Com(#forms.CurrentItem) Image(#STD_ICON)
#std_text := 'FORM13'
Add_Entry To_List(#forms)
Set Com(#forms.CurrentItem) Image(#STD_ICON)
#std_text := 'FORM14'
Add_Entry To_List(#forms)
Set Com(#forms.CurrentItem) Image(#STD_ICON)
#std_text := 'FORM15'
Add_Entry To_List(#forms)
Set Com(#forms.CurrentItem) Image(#STD_ICON)
#std_text := 'Function Options(*DIRECT)'
Add_Entry To_List(#source)
#std_text := '  Begin_Com Role(*EXTENDS #PRIM_FORM)
Clientheight(541) Clientwidth(825) Height(575) Left(340) Top(178)
Width(833)'
Add_Entry To_List(#source)
#std_text := '    Define_Com Class(#PRIM_TAB) Name(#TAB_1)
Displayposition(1) Docklefttabwidth(157) Height(489) Left(48)
Parent(#COM_OWNER) Tabposition(1) Tabstop(False) Top(16) Width(713)
\n End_Com'
Add_Entry To_List(#source)
#std_text := 'End_Com'
Add_Entry To_List(#source)
Endroutine
End_Com
```


タブ・フォルダーのプロパティ

BottomAllowAutoHide プロパティ	RightAutoHide プロパティ	TabStyle プロパティ
BottomAutoHide プロパティ	RightTabLocation プロパティ	MultiLine プロパティ
BottomTabHeight プロパティ	RightTabWidth プロパティ	RaggedRight プロパティ
BottomTabLocation プロパティ	RightViewStyle プロパティ	OpenPage プロパティ
BottomViewStyle プロパティ	TopAllowAutoHide プロパティ	TabHeight プロパティ
DragTabs プロパティ	TopAutoHide プロパティ	TabWidth プロパティ
LeftAllowAutoHide プロパティ	TopTabHeight プロパティ	ViewStyle プロパティ
LeftAutoHide プロパティ	TopTabLocation プロパティ	ShowPageTabs プロパティ
LeftTabLocation プロパティ	TopViewStyle プロパティ	LeftShowPageTabs プロパティ
LeftTabWidth プロパティ	FixedWidth プロパティ	TopShowPageTabs プロパティ
LeftViewStyle プロパティ	TabLocation プロパティ	BottomShowPageTabs プロパティ
RightAllowAutoHide プロパティ	TabLayout プロパティ	RightShowPageTabs プロパティ

BottomAllowAutoHide プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

BottomAllowAutoHideプロパティを使って、タブ・シートがウィンドウの下に添付された時（タブ・シートのDockPositionプロパティを参照）、タブ・フォルダーを自動非表示(autohide)にできるかどうかを指定します。

自動非表示を使用する場合、ユーザーがタブ・シートの右上の添付ボタン(ピン・アイコン)をクリックすると、タブ・フォルダーは自動的に非表示になります。タブ・フォルダーが非表示になると、タブ・シートのタブのみが表示されます。タブ・フォルダーは、ユーザーがタブをクリックすると再度表示されます。自動非表示を終了するには、再度添付ボタンをクリックします。

このプロパティをtrueに設定すると、添付ボタンがタブ・フォルダー内の全てのタブ・シートに表示され、ユーザーが自動非表示をアクティブにすることができるようになります。

自動非表示がどのようなものかを確認するには、LANSAエディターのリポジトリ・タブの添付ボタンをクリックする、もしくは次のコードをフォームにコピー・貼り付けします。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Docking
Framework Sample') CLIENTHEIGHT(378) CLIENTWIDTH(549)
HEIGHT(405) LAYOUTMANAGER(#ATLM_1) LEFT(474) TOP(136)
WIDTH(557)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
BOTTOMTABHEIGHT(141) DISPLAYPOSITION(1) HEIGHT(378)
LEFT(0) LEFTTABWIDTH(154) PARENT(#COM_OWNER)
TABPOSITION(1) TABSTOP(False) TOP(0) VISUALSTYLE(#VS_TAB)
WIDTH(549)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_1)
CAPTION('Page1') DISPLAYPOSITION(1) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Bottom) HEIGHT(112)
LEFT(4) PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
```

```
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_2)
CAPTION('Page2') DISPLAYPOSITION(2) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Bottom) HEIGHT(112)
LEFT(4) PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_3)
CAPTION('Page3') DISPLAYPOSITION(1) HEIGHT(203) LEFT(4)
OPENED(True) PARENT(#TAB_1) TABPOSITION(3) TABSTOP(False)
TOP(25) WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_4)
CAPTION('Page4') DISPLAYPOSITION(2) HEIGHT(203) LEFT(4)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_5)
CAPTION('Page5') DISPLAYPOSITION(3) HEIGHT(203) LEFT(4)
PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(541)
End_Com
```

BottomAutoHide プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

BottomAutohideプロパティを使用して、プログラムで画面の下にタブ・フォルダーを自動非表示にする、またはタブ・フォルダーが自動非表示になっているかどうかを検出します。

このプロパティの値は、TrueまたはFalseになります。

このプロパティは、自動非表示のタブ・フォルダーの設計を簡単にするために、設計時には効果はありません。

BottomTabHeight プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

BottomTabHeightプロパティを使用して、タブ・フォルダーが自動非表示モードではない時の分割線の位置の高さを指定します。

次のコードをコピー・貼り付けして、このプロパティをどのように使用するかを確認できます。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Docking
Framework Sample') CLIENTHEIGHT(378) CLIENTWIDTH(549)
HEIGHT(405) LAYOUTMANAGER(#ATLM_1) LEFT(474) TOP(136)
WIDTH(557)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
BOTTOMTABHEIGHT(141) DISPLAYPOSITION(1) HEIGHT(378)
LEFT(0) LEFTTABWIDTH(154) PARENT(#COM_OWNER)
TABPOSITION(1) TABSTOP(False) TOP(0) VISUALSTYLE(#VS_TAB)
WIDTH(549)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_1)
CAPTION('Page1') DISPLAYPOSITION(1) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Bottom) HEIGHT(112)
LEFT(4) PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_2)
CAPTION('Page2') DISPLAYPOSITION(2) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Bottom) HEIGHT(112)
LEFT(4) PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_3)
CAPTION('Page3') DISPLAYPOSITION(1) HEIGHT(203) LEFT(4)
PARENT(#TAB_1) TABPOSITION(3) TABSTOP(False) TOP(25)
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_4)
CAPTION('Page4') DISPLAYPOSITION(2) HEIGHT(203) LEFT(4)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
```

```
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_5)
CAPTION('Page5') DISPLAYPOSITION(3) HEIGHT(203) LEFT(4)
PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(541)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1)
CAPTION('Change BottomTabHeight') DISPLAYPOSITION(1) LEFT(152)
PARENT(#TBESH_3) TABPOSITION(1) TOP(64) WIDTH(209)
EVTROUTINE HANDLING(#PHBN_1.Click)
set #tab_1 BottomTabHeight(250)
ENDROUTINE
End_Com
```

BottomTabLocation プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

BottomTabLocationプロパティを使って、タブ・フォルダーがウィンドウの下に添付された時にタブをどのサイドに表示させるかを指定します。

タブ・シートはDockPositionプロパティを使って添付されます。

BottomViewStyle プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

BottomViewStyleプロパティを使って、ウィンドウの下に添付されたタブ・フォルダーのタブをどのように表示するかを制御します。表示方法はTabs（タブ）とStacked（重ねる）から選択できます。

DragTabs プロパティ

このプロパティを使って、マウスでタブシートの位置を変更できるかどうか指定します。

DragTabsプロパティをTrueに設定すると、マウスを押さえてドラッグすることでタブシートを移動させることができます。

LeftAllowAutoHide プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

LeftAllowAutoHideプロパティを使って、タブ・シートが左側に添付された時(タブ・シートのDockPositionプロパティを参照)、タブ・フォルダーを自動非表示(autohide)にできるかどうかを指定します。

自動非表示を使用する場合、ユーザーがタブ・シートの右上の添付ボタン(ピン・アイコン)をクリックすると、タブ・フォルダーは自動的に非表示になります。タブ・フォルダーが非表示になると、タブ・シートのタブのみが表示されます。タブ・フォルダーは、ユーザーがタブをクリックすると再度表示されます。自動非表示を終了するには、再度添付ボタンをクリックします。

このプロパティをtrueに設定すると、添付ボタンがタブ・フォルダー内の全てのタブ・シートに表示され、ユーザーが自動非表示をアクティブにすることができるようになります。

自動非表示がどのようなものかを確認するには、LANSAエディターのリポジトリ・タブの添付ボタンをクリックする、もしくは次のコードをフォームにコピー・貼り付けします。

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('Docking Framework
Sample') Clientheight(378) Clientwidth(549) Height(405)
Layoutmanager(#ATLM_1) Left(208) Top(151) Width(557)
Define_Com Class(#PRIM_TAB) Name(#TAB_1) Displayposition(1)
Height(378) Left(0) Lefttabwidth(154) Parent(#COM_OWNER)
Tabposition(1) Tabstop(False) Top(0) Visualstyle(#VS_TAB) Width(549)
Define_Com Class(#PRIM_ATLM) Name(#ATLM_1)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_1) Attachment(Center)
Manage(#TAB_1) Parent(#ATLM_1)
Define_Com Class(#PRIM_TBSH) Name(#TBSH_1) Caption('Page1')
Displayposition(1) Dockallowundock(True) Dockclosebutton(True)
Dockposition(Left) Height(349) Left(4) Opened(True) Parent(#TAB_1)
Tabposition(1) Tabstop(False) Top(25) Width(146)
Define_Com Class(#PRIM_TBSH) Name(#TBSH_2) Caption('Page2')
Displayposition(2) Dockallowundock(True) Dockclosebutton(True)
Dockposition(Left) Height(349) Left(4) Parent(#TAB_1) Tabposition(2)
Tabstop(False) Top(25) Width(146)
```

```
Define_Com Class(#PRIM_TBESH) Name(#TBESH_3) Caption('Page3')
Displayposition(1) Height(349) Left(4) Opened(True) Parent(#TAB_1)
Tabposition(3) Tabstop(False) Top(25) Width(382)
Define_Com Class(#PRIM_TBESH) Name(#TBESH_4) Caption('Page4')
Displayposition(2) Height(349) Left(4) Parent(#TAB_1) Tabposition(2)
Tabstop(False) Top(25) Width(382)
Define_Com Class(#PRIM_TBESH) Name(#TBESH_5) Caption('Page5')
Displayposition(3) Height(349) Left(4) Parent(#TAB_1) Tabposition(1)
Tabstop(False) Top(25) Width(382)
End_Com
```

LeftAutoHide プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

LeftAutohideプロパティを使用して、プログラムで画面の左側にタブ・フォルダーを自動非表示にする、またはタブ・フォルダーが自動非表示になっているかどうかを検出します。

このプロパティの値は、TrueまたはFalseになります。

このプロパティは、自動非表示のタブ・フォルダーの設計を簡単にするために、設計時には効果はありません。

LeftTabLocation プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

LeftTabLocation プロパティを使って、タブ・フォルダーがウィンドウの左側に添付された時にタブをどのサイドに表示させるかを指定します。

タブ・シートはDockPosition プロパティを使って添付されます。

LeftTabWidth プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

LeftTabWidthプロパティを使用して、タブ・フォルダーが自動非表示モードではない時の分割線の位置の幅を指定します。

LeftTabWidthの効果を確認するには以下のコードをコピー・貼り付けしてください。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Docking
Framework Sample') CLIENTHEIGHT(378) CLIENTWIDTH(549)
HEIGHT(405) LAYOUTMANAGER(#ATLM_1) LEFT(347) TOP(148)
WIDTH(557)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
DISPLAYPOSITION(1) HEIGHT(378) LEFT(0) LEFTTABWIDTH(154)
PARENT(#COM_OWNER) TABPOSITION(1) TABSTOP(False) TOP(0)
VISUALSTYLE(#VS_TAB) WIDTH(549)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_1)
CAPTION('Page1') DISPLAYPOSITION(1) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Left) HEIGHT(349)
LEFT(4) PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(146)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_2)
CAPTION('Page2') DISPLAYPOSITION(2) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Left) HEIGHT(349)
LEFT(4) PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(146)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_3)
CAPTION('Page3') DISPLAYPOSITION(1) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(3) TABSTOP(False) TOP(25)
WIDTH(382)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_4)
CAPTION('Page4') DISPLAYPOSITION(2) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(382)
```

```
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_5)
CAPTION('Page5') DISPLAYPOSITION(3) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(382)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1)
CAPTION('Change LeftTabWidth') DISPLAYPOSITION(1) LEFT(88)
PARENT(#TBESH_3) TABPOSITION(1) TOP(167) WIDTH(201)
EVTROUTINE HANDLING(#PHBN_1.Click)
set #tab_1 LeftTabWidth(300)
ENDROUTINE
End_Com
```

LeftViewStyle プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

LeftViewStyleプロパティを使って、左側に添付されたタブ・フォルダーのタブをどのように表示するかを制御します。表示方法はTabs（タブ）とStacked（重ねる）から選択できます。

タブ・シートはDockPositionプロパティを使って添付されます。

RightAllowAutoHide プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

RightAllowAutoHideプロパティを使って、タブ・シートが右側に添付された時（タブ・シートのDockPositionプロパティを参照）、タブ・フォルダーを自動非表示(autohide)にできるかどうかを指定します。

自動非表示を使用する場合、ユーザーがタブ・シートの右上の添付ボタン(ピン・アイコン)をクリックすると、タブ・フォルダーは自動的に非表示になります。タブ・フォルダーが非表示になると、タブ・シートのタブのみが表示されます。タブ・フォルダーは、ユーザーがタブをクリックすると再度表示されます。自動非表示を終了するには、再度添付ボタンをクリックします。

このプロパティをtrueに設定すると、添付ボタンがタブ・フォルダー内の全てのタブ・シートに表示され、ユーザーが自動非表示をアクティブにすることができるようになります。

自動非表示がどのようなものかを確認するには、LANSAエディターのリポジトリ・タブの添付ボタンをクリックする、もしくは次のコードをフォームにコピー・貼り付けします。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Docking
Framework Sample') CLIENTHEIGHT(378) CLIENTWIDTH(549)
HEIGHT(405) LAYOUTMANAGER(#ATLM_1) LEFT(426) TOP(147)
WIDTH(557)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
DISPLAYPOSITION(1) HEIGHT(378) LEFT(0) LEFTTABWIDTH(154)
PARENT(#COM_OWNER) RIGHTTABWIDTH(149) TABPOSITION(1)
TABSTOP(False) TOP(0) VISUALSTYLE(#VS_TAB) WIDTH(549)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_1)
CAPTION('Page1') DISPLAYPOSITION(2) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Right) HEIGHT(349)
LEFT(4) PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(141)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_2)
```

CAPTION('Page2') DISPLAYPOSITION(1) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Right) HEIGHT(349)
LEFT(4) PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(141)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_3)
CAPTION('Page3') DISPLAYPOSITION(1) HEIGHT(349) LEFT(4)
OPENED(True) PARENT(#TAB_1) TABPOSITION(3) TABSTOP(False)
TOP(25) WIDTH(387)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_4)
CAPTION('Page4') DISPLAYPOSITION(2) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(387)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_5)
CAPTION('Page5') DISPLAYPOSITION(3) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(387)
End_Com

RightAutoHide プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

RightAutohideプロパティを使用して、プログラムで画面の右側にタブ・フォルダーを自動非表示にする、またはタブ・フォルダーが自動非表示になっているかどうかを検出します。

このプロパティの値は、TrueまたはFalseになります。

このプロパティは、自動非表示のタブ・フォルダーの設計を簡単にするために、設計時には効果はありません。

RightTabLocation プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

RightTabLocationプロパティを使って、タブ・フォルダーがウィンドウの右側に添付された時にタブをどのサイドに表示させるかを指定します。

タブ・シートはDockPositionプロパティを使って添付されます。

RightTabWidth プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

RightTabWidthプロパティを使用して、タブ・フォルダーが自動非表示モードではない時の分割線の位置の幅を指定します。

次のコードをコピー・貼り付けして、RightTabWidthプロパティの効果を確認できます。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Docking
Framework Sample') CLIENTHEIGHT(378) CLIENTWIDTH(549)
HEIGHT(405) LAYOUTMANAGER(#ATLM_1) LEFT(426) TOP(147)
WIDTH(557)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
DISPLAYPOSITION(1) HEIGHT(378) LEFT(0) LEFTTABWIDTH(154)
PARENT(#COM_OWNER) RIGHTTABWIDTH(149) TABPOSITION(1)
TABSTOP(False) TOP(0) VISUALSTYLE(#VS_TAB) WIDTH(549)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_1)
CAPTION('Page1') DISPLAYPOSITION(2) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Right) HEIGHT(349)
LEFT(4) PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(141)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_2)
CAPTION('Page2') DISPLAYPOSITION(1) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Right) HEIGHT(349)
LEFT(4) OPENED(True) PARENT(#TAB_1) TABPOSITION(1)
TABSTOP(False) TOP(25) WIDTH(141)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_3)
CAPTION('Page3') DISPLAYPOSITION(1) HEIGHT(349) LEFT(4)
OPENED(True) PARENT(#TAB_1) TABPOSITION(3) TABSTOP(False)
TOP(25) WIDTH(387)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_4)
CAPTION('Page4') DISPLAYPOSITION(2) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(387)
```

```
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_5)
CAPTION('Page5') DISPLAYPOSITION(3) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(387)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1)
CAPTION('Change RightTabWidth') DISPLAYPOSITION(1) LEFT(89)
PARENT(#TBSH_3) TABPOSITION(1) TOP(154) WIDTH(216)
EVTROUTINE HANDLING(#PHBN_1.Click)
set #tab_1 RightTabWidth(300)
ENDROUTINE
End_Com
```

RightViewStyle プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

RightViewStyleプロパティを使って、右側に添付されたタブ・フォルダーのタブをどのように表示するかを制御します。表示方法はTabs（タブ）とStacked（重ねる）から選択できます。

タブ・シートはDockPositionプロパティを使って添付されます。

TopAllowAutoHide プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

TopAllowAutoHideプロパティを使って、タブ・シートがウィンドウの上に添付された時(タブ・シートのDockPositionプロパティを参照)、タブ・フォルダーを自動非表示(autohide)にできるかどうかを指定します。自動非表示を使用する場合、ユーザーがタブ・シートの右上の添付ボタン(ピン・アイコン)をクリックすると、タブ・フォルダーは自動的に非表示になります。タブ・フォルダーが非表示になると、タブ・シートのタブのみが表示されます。タブ・フォルダーは、ユーザーがタブをクリックすると再度表示されます。自動非表示を終了するには、再度添付ボタンをクリックします。

このプロパティをtrueに設定すると、添付ボタンがタブ・フォルダー内の全てのタブ・シートに表示され、ユーザーが自動非表示をアクティブにすることができるようになります。

自動非表示がどのようなものかを確認するには、LANSAエディターのリポジトリ・タブの添付ボタンをクリックする、もしくは次のコードをフォームにコピー・貼り付けします。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Docking
Framework Sample') CLIENTHEIGHT(378) CLIENTWIDTH(549)
HEIGHT(405) LAYOUTMANAGER(#ATLM_1) LEFT(474) TOP(136)
WIDTH(557)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
BOTTOMTABHEIGHT(141) DISPLAYPOSITION(1) HEIGHT(378)
LEFT(0) LEFTTABWIDTH(154) PARENT(#COM_OWNER)
TABPOSITION(1) TABSTOP(False) TOP(0) VISUALSTYLE(#VS_TAB)
WIDTH(549)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_1)
CAPTION('Page1') DISPLAYPOSITION(2) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Top) HEIGHT(71)
LEFT(4) PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(541)
```

```
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_2)
CAPTION('Page2') DISPLAYPOSITION(1) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Top) HEIGHT(71)
LEFT(4) OPENED(True) PARENT(#TAB_1) TABPOSITION(1)
TABSTOP(False) TOP(25) WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_3)
CAPTION('Page3') DISPLAYPOSITION(1) HEIGHT(244) LEFT(4)
OPENED(True) PARENT(#TAB_1) TABPOSITION(3) TABSTOP(False)
TOP(25) WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_4)
CAPTION('Page4') DISPLAYPOSITION(2) HEIGHT(244) LEFT(4)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_5)
CAPTION('Page5') DISPLAYPOSITION(3) HEIGHT(244) LEFT(4)
PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(541)
End_Com
```

TopAutoHide プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

TopAutohideプロパティを使用して、プログラムで画面の上にタブ・フォルダーを自動非表示にする、またはタブ・フォルダーが自動非表示になっているかどうかを検出します。

このプロパティの値は、TrueまたはFalseになります。

このプロパティは、自動非表示のタブ・フォルダーの設計を簡単にするために、設計時には効果はありません。

TopTabHeight プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

TopTabHeight プロパティを使用して、タブ・フォルダーが自動非表示モードではない時の分割線の位置の高さを指定します。

次のコードをコピー・貼り付けして、ドッキング・アプリケーションの基本を確認できます。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Docking
Framework Sample') CLIENTHEIGHT(378) CLIENTWIDTH(549)
HEIGHT(405) LAYOUTMANAGER(#ATLM_1) LEFT(474) TOP(136)
WIDTH(557)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
BOTTOMTABHEIGHT(141) DISPLAYPOSITION(1) HEIGHT(378)
LEFT(0) LEFTTABWIDTH(154) PARENT(#COM_OWNER)
TABPOSITION(1) TABSTOP(False) TOP(0) VISUALSTYLE(#VS_TAB)
WIDTH(549)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_1)
CAPTION('Page1') DISPLAYPOSITION(2) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Top) HEIGHT(71)
LEFT(4) PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_2)
CAPTION('Page2') DISPLAYPOSITION(1) DOCKALLOWUNDOCK(True)
DOCKCLOSEBUTTON(True) DOCKPOSITION(Top) HEIGHT(71)
LEFT(4) OPENED(True) PARENT(#TAB_1) TABPOSITION(1)
TABSTOP(False) TOP(25) WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_3)
CAPTION('Page3') DISPLAYPOSITION(1) HEIGHT(244) LEFT(4)
OPENED(True) PARENT(#TAB_1) TABPOSITION(3) TABSTOP(False)
TOP(25) WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_4)
CAPTION('Page4') DISPLAYPOSITION(2) HEIGHT(244) LEFT(4)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
```

```
WIDTH(541)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_5)
CAPTION('Page5') DISPLAYPOSITION(3) HEIGHT(244) LEFT(4)
PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(541) DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1)
CAPTION('Change TopTabHeight') DISPLAYPOSITION(1) LEFT(136)
PARENT(#TBESH_3) TABPOSITION(1) TOP(88) WIDTH(220)
EVTROUTINE HANDLING(#PHBN_1.Click)
set #tab_1 TopTabHeight(200)
ENDROUTINE
End_Com
```

TopTabLocation プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

TopTabLocationプロパティを使って、タブ・フォルダーがウィンドウの上に添付された時にタブをどのサイドに表示させるかを指定します。

タブ・シートはDockPositionプロパティを使って添付されます。

TopViewStyle プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

TopViewStyleプロパティを使って、上に添付されたタブ・フォルダーのタブをどのように表示するかを制御します。表示方法はTabs（タブ）とStacked（重ねる）から選択できます。

タブ・シートはDockPositionプロパティを使って添付されます。

FixedWidth プロパティ

FixedWidthは全てのタブが同じ幅になるよう指定します。

FixedWithプロパティをTrueに設定して、全てのタブが同じ幅になるようにします。

FixedWithプロパティに値を設定しても、全てのタブが同じ幅になりません。

TabLocation プロパティ

TabLocationプロパティを使ってタブフォルダのどのサイドにタブを表示するか指定します。省略値ではタブは上に設定されています。

TabLayout プロパティ

TabLayout プロパティを利用して、タブのレイアウトを設定します。

Centeredはタブでイメージとキャプションが中央に揃えられます。

IconLeftはアイコンがタブの左に揃えられ、キャプションは残りのスペースで中央に揃えられます。

CaptionLeftはアイコンとキャプションの両方が左に揃えられます。

IconLeftとCaptionLeftはタブが固定サイズの場合のみ適用されます。

TabStyle プロパティ

TabStyleプロパティを使用して、タブをどのように表示するかを指定します。 tabs、 buttonsまたはflat buttonsの中から選択できます。

MultiLine プロパティ

MultiLineプロパティにTrueを設定して、タブシートのキャプションに数行表示できるようにします。

RaggedRight プロパティ

RaggedRightプロパティを使用して、MultiLineプロパティにTrueが設定された時にタブシートをどのように表示するかを制御します。

MultiLineプロパティにFalseが設定されている場合、タブシートは全部のタブで埋められるように調整されます。 Trueに設定されている場合、タブシートの幅はそのキャプションにより決定されます。

OpenPage プロパティ

OpenPageはタブ・フォルダーの一番上のタブシートです。
実行時のみ、読み取り専用プロパティです。

OpenPageプロパティはタブの中で現在一番上にあるタブシートを示します。タブシートの[Opened プロパティ](#)も参照してください。

次のコードは#TAB_1で現在開いているタブシート名を編集ボックス#Edit_3に入れます。

```
set com(#edit_3) value(#tab_1.OpenPage.Name)
```

TabHeight プロパティ

TabHeight プロパティを使って、タブシートのキャプションが表示されているエリアの高さを設定します。

TabWidth プロパティ

TabWidthプロパティを使って、タブシートのキャプションが表示されているエリアの幅を設定します。

ViewStyle プロパティ

ViewStyle プロパティを使用して、タブ・フォルダーをどのように表示するか制御します。表示方法はTabs (タブ) とStacked (重ねる) から選択できます。

ShowPageTabs プロパティ

ShowPageTabsはタブのページ・タブのヘッダーを表示するかどうかを示します。

ShowPageTabsプロパティを使って、タブのタブ・ページのヘッダーを非表示にできます。これはドッキング機能を使用している時に DockPosition(Center) のタブ・ページのヘッダーを表示したくない場合などに便利です。

LeftShowPageTabs プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

LeftShowPageTabsプロパティを使って、全てのDockPosition(Left)タブ・ページのヘッダーを非表示にします。

TopShowPageTabs プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

TopShowPageTabs プロパティを使って、全てのDockPosition(Top)タブ・ページのヘッダーを非表示にします。

BottomShowPageTabs プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

BottomShowPageTabs プロパティを使って、全てのDockPosition(Bottom)タブ・ページのヘッダーを非表示にします。

RightShowPageTabs プロパティ

タブ・フレームワークのプロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

RightShowPageTabs プロパティを使って、全てのDockPosition(Right)タブ・ページのヘッダーを非表示にします。

タブ・フォルダーのイベント

TabChanging イベント

TabChanging イベント

TabChangingイベントは別のタブ・シートが表示される寸前に起動されます。

このイベントを使ってユーザーがタブをクリックした時に何が起きるかを決定することができます。 CanChangePageパラメータを使用して、タブの表示を停止することができます。 ただし、このイベントを使用する時はユーザーが予期しないタブ動作にならないよう注意してください。

次のコードをフォームにコピーして、コンパイルするとTabChangingイベントをどのように使用されるか確認できます。

```
BEGIN_COM CAPTION('TabChanging Event') HEIGHT(298) LEFT(372) TC
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1) DISPLAYPOSITION
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_1) CAPTION('Tab 1
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_2) CAPTION('Tab 2
DEFINE_COM CLASS(#PRIM_CKBX) NAME(#CKBX_1) CAPTION('Allo
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_3) CAPTION('Tab 3
DEFINE_COM CLASS(#PRIM_EDIT) NAME(#EDIT_1) DISPLAYPOSITIO
DEFINE_COM CLASS(#PRIM_EDIT) NAME(#EDIT_2) DISPLAYPOSITIO
DEFINE_COM CLASS(#PRIM_LABL) NAME(#LABL_1) CAPTION('Open
DEFINE_COM CLASS(#PRIM_LABL) NAME(#LABL_2) CAPTION('CanC
* note that the parameter names cannot be repository fields or fields defined in
EVTROUTINE HANDLING(#TAB_1.TabChanging) OpenPage(#TheTab) Ca
if cond('#TheTab.name *eq #tbsh_2.name')
if cond('#ckbx_1.ButtonState *eq Unchecked')
change #std_bool 'False'
else
change #std_bool 'True'
endif
set #CanChange value(#std_bool)
endif
set com(#edit_1) value(#TheTab.name)
set com(#edit_2) value(#CanChange);
ENDROUTINE
end_com
```

[OpenPage パラメータ](#)

[CanChangePage パラメータ](#)

OpenPage パラメータ

Open Pageパラメータを使って、タブの中で現在どのタブ・シートが開いているかを探し出します。

CanChangePage パラメータ

CanChangePageパラメータを使用して、タブ・シートの表示を止めたり、表示できるようにしたりします。

タブ・シート

タブ・シートを使用して、タブ・フォルダー上に置く1ページの情報を作成します。

[タブ・シートのプロパティ](#)

[タブ・シートのイベント](#)

[タブ・シートのメソッド](#)

タブ・シートのプロパティ

Opened プロパティ

Opened プロパティ

Openedプロパティを使用して、どれを一番上のタブ・シートにするかを制御します。一番上のタブ・シートのOpenedプロパティにはTrueを設定します。

このプロパティを使ってプログラムでタブ・シートを表示することができます。

```
set com(#TBSH_4) Opened(True)
```

タブ・シートのイベント

Opening イベント

Closing イベント

Opening イベント

Opening イベントはタブ・シートがタブの一番上に持ってこられた時に起動されます。このイベントを使ってタブ・シートに表示するデータをロードします。

Closing イベント

タブ・シートのClosingイベントは別のタブ・シートが開かれた時に起動されます。

タブ・シートのメソッド

Open メソッド

Open メソッド

Openメソッドを使用してタブシートをタブの一番上にします。

タブ・シートToolbar

Toolbarを使って様々なツールバー・ボタンを表示します。

Toolbarにはアプリケーション・メニュー項目に対応するボタンが含まれ、アプリケーションで最も良く使用される機能やコマンドにユーザーがアクセスできるグラフィック・インターフェイスを提供します。

ビジュアル・スタイル

ビジュアル・スタイルはフォントと色を設定します。

ビジュアル・スタイルを使って複数の他のコンポーネントと共有する共通のスタイルを作成します。ビジュアル・スタイル・コンポーネントは背景色やフォントなどのスタイル要素を制御します。いったんスタイルを作成すると、ビジュアル・スタイル・プロパティとしてどのコンポーネントでも使用することができます。ビジュアル・スタイルを使用することで、アプリケーションの外観全体を簡単に変更することができます。

ビジュアル・スタイルは複数のビジュアル・スタイル・スキームの1つを構成します。複数言語のアプリケーションを作成する場合、スタイルにつき1つ以上のスキームを定義して、言語により異なるフォントや色を使用することができます。

スキームを使って、3種類の表示オブジェクト (Value、Caption、Title) をビジュアル・スタイル項目にマップできます。ビジュアル・スタイル項目はフォントと色の定義です。

ビジュアル・スタイル・スキームMY_SCHEMEの1つを構成するスタイルMY_VSなどがビジュアル・スタイルの例です。このスキームはValueタイプのオブジェクト (フィールドやリストなど) がビジュアル・スタイル項目のValueを使用し、Captionタイプのオブジェクト (フォームやラベルなど) とTitleオブジェクト (グラフ・タイトル) がビジュアル・スタイル項目Captionを使用することを定義します。Valueはシステム・フォントおよび白い背景を使用することができ、Captionは灰色の背景を使用できます。

ビジュアル・スタイルはアプリケーションがどのように表示されるかを完全にコントロールしている訳ではありません。フォームのタイトル・バー、スクロール・バーやメニューなどのWindowsコントロールはWindowsのコントロール・パネルでユーザーにより設定された色を使用して表示されます。

Default [プロパティ](#)

Default プロパティ

Default ビジュアル・スタイルは言語特有のスキームがない場合に使用されるビジュアル・スタイルを指定します。

区画に定義された全ての言語に対してビジュアル・スタイル・スキームを定義することができます。これは、言語によりフォント・フェイス名を変更したい場合などに便利です。

ビジュアル・スタイル・スキーム

ビジュアル・スタイル・スキームで異なるオブジェクトのスタイル(ビジュアル・スタイル項目)を定義できます。Values、CaptionsおよびTitles

Valuesに定義するスタイルは、フィールドの編集ボックスなどのコンポーネントがどのように表示されるかを決定します。

Captionsに定義するスタイルではフィールド・ラベルなどのコンポーネントがどのように表示されるかを決定します。多くのスタイルではCaptionの背景色は灰色です。

Titlesはグラフ・タイトルのスタイルを設定します。

スタイルを変更する時は、どの項目がどのコンポーネント・タイプに影響を与えるのかをよく理解しておいてください。

Values プロパティ

Captions プロパティ

Titles プロパティ

Values プロパティ

Valueに定義する項目は、フィールドの編集ボックスなどのコンポーネントがどのように表示されるかを決定します。例えば、Values項目の背景色はWindowsの背景色にマップされていることが非常によくあります。これは通常白またはその他の明るい色です。

以下の外観を変更したい場合はValues項目のプロパティを変更します。

コンボ・ボックス

編集ボックス

フィールドの編集ボックス

グリッド・セル

リスト・ボックス

リスト・ビュー

スピン編集

ツリー・ビュー

Captions プロパティ

Captionsはフォーム、ラベルなどのビジュアル・スタイル項目です。Captionsに定義する項目はフィールド・ラベルなどのコンポーネントがどのように表示されるかを決定します。多くのスタイルではCaptionsの背景色は灰色です。

以下の外観を変更したい場合はCaptions項目のプロパティを変更します。

チェック・ボックス	ラジオ・ボタン
フィールドのラベル	ラベル
グリッド見出し	ステータス・バー
グループ・ボックス	タブ・フォルダー
リスト・ビュー見出し	タブ・シート
プッシュ・ボタン	トラック・バー

Titles プロパティ

Titlesはグラフ・タイトルのビジュアル・スタイル項目を定義します。

ビジュアル・スタイル項目

ビジュアル・スタイル項目はフォント、色や枠線のスタイルを定義します。ビジュアル・スタイルはビジュアル・スタイルのスキーム・コンポーネントのValue、CaptionもしくはTitleオブジェクトにマップされています。

1つのビジュアル・スタイル項目は複数のビジュアル・スタイル・スキームで使用できます。また、1つのスキーム内のValue、CaptionとTitleオブジェクトに1つの同じビジュアル・スタイルを使用することも可能です。ただし、普通はValueオブジェクト（リストやフィールドなど）とCaptionオブジェクト（フォームなど）を異なる背景色にするので、それぞれ別のスタイルを定義するのが通常です。

ほとんどのビジュアル・スタイル項目のプロパティ値は、'red'という色や'MS Sans Serif'というフォント名などの特定の値か、もしくは特定のワークステーション上のシステム・オブジェクトの対応するプロパティにすることができます。つまり、例えて言えばウィンドウの背景色を常に'白'にすることもできますし、システム上のウィンドウの背景色の色になるよう定義することも可能です。詳細については、NormBackColorプロパティを参照してください。

AlternBackColor プロパティ	ErrorBackColor プロパティ	StrikeOut プロパティ
BorderColor プロパティ	FaceName プロパティ	Script プロパティ
LineColor プロパティ	FontSize プロパティ	SQLNullBackColor プロパティ
Bold プロパティ	Italic プロパティ	TextColor プロパティ
BorderStyle プロパティ	NormBackColor プロパティ	Underline プロパティ

AlternBackColor プロパティ

AlternBackColorはリスト内の1行おきの色です。

リスト・タイプ・コンポーネントにのみ適用されます。ですから、Valueスタイル項目の定義がされた時のみ効果を発揮します。

AlternBackColorプロパティを使って、リスト・ボックスやコンボ・ボックスの1行おきの背景色を設定します。背景色を交互にすることで、リストが読みやすくなります。

ビジュアル・スタイルの他の色プロパティと同様、'white'などの標準色名を値として定義することもできますし、システム・オブジェクトの色を指定することもできます。システム・オブジェクト色についての詳細は、NormBackColorのヘルプを参照してください。

BorderColor プロパティ

BorderColorはグリッド枠線の色を設定します。

グリッドにのみ適用されます。 ですから、Valueスタイル項目の定義がされた時のみ効果を発揮します。

BorderColor プロパティを使ってグリッドの枠線の色を設定できます。

LineColor プロパティ

LineColorはグリッド行の色を設定します。

グリッドにのみ適用されます。 ですから、Valueスタイル項目の定義がされた時のみ効果を発揮します。

LineColorプロパティを使ってグリッドの行の色を設定できます。 省略値は黒です。

Bold プロパティ

Boldプロパティを使用して、このスタイルで使用されるフォントを太字にします。

BorderStyle プロパティ

BorderStyleはコンポーネントを3Dやフラットなどにします。

BorderStyleプロパティを使用して、コンポーネントをどのように表示するかを指定します。このプロパティは、次の値を取ることができます。

3D Left--コンポーネントが3Dとして表示され、左端に影が付きます。

3D Right--コンポーネントが3Dとして表示され、右端に影が付きます。

None--BorderStyleは適用されません。

Raised--コンポーネントが3Dとして表示されます。

Indented--コンポーネントはインデント表示されます。

ThickSolid--コンポーネントは太い枠線付きでフラット表示されます。

ThinSolid--コンポーネントは細い枠線付きでフラット表示されます。

ErrorBackColor プロパティ

ErrorBackColorはエラーの色を設定します。

リスト・タイプ、フィールドおよび編集ボックス・コンポーネントにのみ適用されます。ですから、Valueスタイル項目の定義がされた時のみ効果を発揮します。

ErrorBackColorプロパティを使用して、フィールドに入力された値が正しくないことを示す際に使用する背景色を定義します。

ErrorBackColorをプログラム上で設定することも可能です。例えば、BEGINCHECK/ENDCHECKブロックのDATECHECK、VALUECHECK、RANGECHECKなどのコマンドと共に使用します。またErrorBackColorは、I/Oモジュールによってオンになることもあります。例えばフィールドを挿入・更新または削除する時に関連のファイルの規則を破ると、エラーの背景色がアクティブになります。

ビジュアル・スタイルの他の色プロパティと同様、'white'などの標準色名を値として定義することもできますし、システム・オブジェクトの色を指定することもできます。システム・オブジェクト色についての詳細は、NormBackColorのヘルプを参照してください。

FaceName プロパティ

FaceName プロパティを使って、使用するフォント名を指定します。

次の特別な値をデスクトップ・フォントにマップすることができます。

VLシェルは、Windows 2000オペレーティング・システムおよびそれ以降（XPを含む）では"MS Shell Dlg2"にマップし、NT、95、98には"MS Shell Dlg"にマップします。

VLシステムはWindowsシステム・フォントにマップしています。

VLシステム固定はWindowsシステムの固定ピッチ・フォントにマップしています。

FontSize プロパティ

FontSize プロパティを利用して、フォントのサイズを設定します。

Italic プロパティ

Italicプロパティを使用して、使用されるフォントを斜体にします。

NormBackColor プロパティ

NormBackColor プロパティを利用して、背景色の色を設定します。

このプロパティの値には色の名前、Windowsオブジェクト名、またはRGB値が指定できます。言い換えると、背景色を標準色 ('red'など)として、またはカスタムの色 (パレットを使用してRGB値として表現)として定義することができ、更に特定のワークステーション上のWindowsオブジェクトに使用されている色と同じものを使うこともできます。例えば、背景をWindowに設定した場合、特定のワークステーション上のウィンドウの背景色として使用されている色になります。

通常システムの色はワークステーションごとに異なります。ですからある色を強制するには色の名前を明確に定義する必要があります。この場合、選択した色がデスクトップの他の色と比べると奇妙に映るかもしれません。またメニューのフォントや色、タイトル・バーのフォントや色、およびスクロール・バーなどの多くのWindowsコントロールはビジュアル・スタイルでは制御できないことにも注意してください。

カスタムの色を指定するには、省略記号 (3つの点) のついたボタンをクリックすると、カラー・パレットが表示されます。カスタムの色を指定する際、色はRGB (赤、緑、青) 値として表示されます。RGB値は色を得るための赤、緑、青の量を示します。形式はRRR:GGG:BBBです。例えば、255:255:255は白、0:0:0は黒です。

選択できるシステム色が多くあることに気がつくと思います。WindowやWindowテキストなどは比較的明らかに分かり、コントロール・パネルに設定された色に対応しています。その他のButtonFaceや3DMediumShadowなどのWindowsシステム色ははっきりとは分かりません。これらの色がアプリケーションでどのような見えるのかを探しだす一番の方法は、実際に試してみることです。

StrikeOut プロパティ

StrikeOutはテキストに取り消し線を引きます。

StrikeOutプロパティを使用して、使用されるフォントに取り消し線を引きます。

Script プロパティ

Scriptはコンピュータに設定されている言語です。

Scriptプロパティを使って、選択したフォントの言語セットを指定します。Scriptはコンピュータに設定された言語に対応しているようにしてください。

SQLNullBackColor プロパティ

SQLNullBackColor プロパティ

SQLNullBackColor プロパティを使用して編集フィールドの背景色を変更してSQLNull値であることを示します。

編集フィールドにデータクラスが添付されており、割り当てられた値が*SQLNullの場合、このフィールドの背景色はDarkGrey（濃い灰色）に変更されSQLNull値であることを示します。 SQLNullBackColorを使ってDarkGrey以外の色を指定することもできます。

SQLNull値に異なる背景色が必要でない場合は、SQLNullBackColorをWindowに変更します。

TextColorプロパティ

TextColorはテキストの色を設定します。

TextColorプロパティを利用して、テキスト色を設定します。

ビジュアル・スタイルの他の色プロパティと同様、'white'などの標準色名を値として定義することもできますし、システム・オブジェクトの色を指定することもできます。システム・オブジェクト色についての詳細は、NormBackColorのヘルプを参照してください。

Underline プロパティ

Underlineはフォントに下線を引きます。

Underlineプロパティを使用して、使用されるフォントに下線を引きます。

タイマー

間隔を置いてコードを実行します。

タイマー・コンポーネントはIntervalプロパティに指定された間隔を置いて、タイマーのTickイベントのコードを実行します。

以下のコードは、タイマーの目盛りごとにイメージを交互に表示、非表示にします。

```
EVTROUTINE HANDLING( #TIMR_1.Tick )
if '#imge_1.Visible *EQ true'
set #imge_1 visible(false)
else
set #imge_1 visible(true)
endif
ENDROUTINE
```

タイマーをアプリケーションにドラッグしてフォームに入れます。ただし、タイマーはビジュアル・コントロールではないので、フォーム上では表示されず、リポシトリ・ブラウザのアウトラインでのみ表示されます。

[タイマーのプロパティ](#)

[タイマーのイベント](#)

[タイマーのメソッド](#)

タイマーのプロパティ

Interval プロパティ

Interval プロパティ

Intervalでタイマーの間隔を設定します。

Intervalプロパティを使って、タイマーのTickイベントをどの間隔で実行するか指定します。間隔はミリ秒単位で指定します。

タイマーを停止するには、Intervalプロパティを0に設定します。

設計時にIntervalプロパティを0に設定して、Interval値を指定することで開始させることができます。設計時に0より大きな値を設定した場合は、タイマーはフォームが初期化された時に開始されます。

タイマーのイベント

タイマーTickイベント

タイマーTickイベント

Tickイベントは間隔の時間が過ぎるごとに実行されます。

Tickイベントを使って、タイマーのIntervalプロパティに指定された時間が経過するごとに実行するコードを指定します。

タイマーのメソッド

Timer Startメソッド

Timer Stopメソッド

Timer Startメソッド

Startメソッドはタイマーを開始します。

Startメソッドを使用して、タイマーを開始します。このメソッドの構文は、以下のとおりです：

```
Invoke #TIMR_1.Start
```

Timer Stopメソッド

Stopメソッドはタイマーを停止します。

Stopメソッドを使用して、タイマーを停止します。このメソッドの構文は、以下のとおりです：

```
Invoke #Timr_1.Stop
```

メニュー・アイテム

メニュー・タイトルまたはメニュー・オプションです。

メニュー・アイテムを使用して、メニューにオプションとタイトルを作成します。

メニュー・アイテムは、メニューバーやサブメニューにドラッグできます。またはメニューバーの点線の長方形をクリックして起動されるメニュー・エディターを使用することもできます。

Checked プロパティ

Default プロパティ

GroupIndex プロパティ

MenuBreak プロパティ

RaioItem プロパティ

SubMenu プロパティ

ShortCut プロパティ

Tag プロパティ

Tip プロパティ

TipShow プロパティ

TipShowOfParent プロパティ

Checked プロパティ

Checkedは、メニュー・オプションが選択されていることを示します。Checkedプロパティを使用して、メニュー・オプションが選択されたことを視覚的に示します。

例えば、ツールバーを表示するメニュー・アイテムがある場合、ツールバーが表示される時はこのアイテムのCheckedプロパティをTrueに、表示されない時はFalseにします。

メニュー・アイテムをRadioItem (True) と指定すると、ラジオ・ボタンはCheckedプロパティがTrueの時のみ表示されます。

以下のコードをコピーして貼り付け、メニュー・アイテムのChecked, RadioItemおよびGroupIndexプロパティがどのような働きをするか確認します。

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('Menus') Clientheight
Define_Com Class(#PRIM_MBAR) Name(#MBAR_1) Parent(#COM_OWNE
Define_Com Class(#PRIM_MITM) Name(#MITM_1) Caption('Menu 1') Disp
Define_Com Class(#PRIM_SMNU) Name(#SMNU_1) Parent(#MITM_1)
Define_Com Class(#PRIM_MITM) Name(#MITM_2) Caption('Small') Check
Define_Com Class(#PRIM_MITM) Name(#MITM_3) Caption('Large') Displa
Define_Com Class(#PRIM_MITM) Name(#MITM_5) Caption('Menu 2') Disp
Define_Com Class(#PRIM_SMNU) Name(#SMNU_2) Parent(#MITM_5)
Define_Com Class(#PRIM_MITM) Name(#MITM_6) Caption('Show picture')
Define_Com Class(#PRIM_IMGE) Name(#IMGE_1) Autosize(False) Display
EVTROUTINE HANDLING(#MITM_2.Click #MITM_3.Click)
If Cond(#mitm_2.checked *eq true)
Set Com(#IMGE_1) Autosize(False)
else
Set Com(#IMGE_1) Autosize(True)
ENDIF
ENDROUTINE
EVTROUTINE HANDLING(#MITM_6.Click)
If Cond(#mitm_6.checked *eq true)
Set Com(#MITM_6) Checked(False)
Set Com(#IMGE_1) Visible(False)
else
Set Com(#MITM_6) Checked(True)
```

```
Set Com(#IMGE_1) Visible(True)
ENDIF
ENDROUTINE
End_Com
```

Default プロパティ

Defaultは、あるメニュー・アイテムを省略値オプションにします。

Defaultプロパティを使用して、あるメニュー・アイテムをポップアップ・メニュー内での省略値オプションにします。ポップアップ・メニューの省略値オプションは、ポップアップ・メニューが属するアイテムをユーザーがダブルクリックすると起動されます。

GroupIndex プロパティ

GroupIndex プロパティを使用して、相互に排他的なメニュー・オプションのグループを作成します。

例えば、リスト・ビューの4つのビュー・スタイル(アイコン、小さいアイコン、レポートおよびリスト)に対応する4つのメニュー・オプションがある場合、4つのオプションすべてに同じグループ・インデックス(例えば1)を与えて一度に1つのみ選択されるようにすることができます。

通常、相互に排他的なメニュー・アイテムは、メニュー・アイテムの RadioItem プロパティを使用し、ラジオ・ボタンとしてビジュアルライズされます。

[Checked プロパティ](#)のヘルプも参照してください。

MenuBreak プロパティ

MenuBreakは、新しい列にメニュー・アイテムを入れます。

以下がプロパティのタイプです。

- None: アイテムは新しい列にはありません。
- Break: アイテムは新しい列にあります。
- BarBreak: アイテムは新しい列にあり、前のアイテムからはバーで分けられています。

RadioItem プロパティ

RadioItemプロパティを使用して、メニュー・アイテムをラジオ・ボタンのようにします。

このプロパティは、GroupIndexプロパティを使用して相互に排他的になった一連のメニュー・アイテムに使用されます。

ラジオ・ボタンはCheckedプロパティをTrueに設定した時のみ表示されます。

SubMenu プロパティ

SubMenuはこのアイテムのサブメニューです。

SubMenuプロパティを使用して、このメニュー・アイテムに表示されるサブメニュー名を指定します。

ShortCut プロパティ

ShortCutプロパティを使用して、メニュー・オプションのショートカットやShortcutコンポーネントを指定します。

メニュー・アイテムにはShortCutプロパティがあります。他のコンポーネントにショートカットキーを設定するには、それらをShortcutコンポーネントと関連付ける必要があります。メニュー・アイテムがPopUp Menuに使われる場合、ShortCutプロパティは無視されることに注意してください。

推奨のショートカットキーは、ファンクションキーおよびCtrl+文字の組み合わせです。ショートカットキーをAlt+文字キーに設定することはできませんが、一般にAlt+文字キーの組み合わせは、キャプションの下線文字によってアクセスキーとして使用されるのでお勧めしません。

F1、Ctrl+f1、およびCtrl+Shift+F1は、必ずアプリケーションのヘルプを表示します。

PageUpとPageDownショートカットキーをメニューで使うと、リスト、ツリーおよびメモボックスなど他のコンポーネント内でPageUpおよびPageDownの機能が使用ができなくなるため、推奨されません。

Tag プロパティ

タグは、メニュー・アイテムを番号で特定します。

Tag プロパティを使用してメニュー・オプションを番号で特定します。

Tip プロパティ

Desc=Tipは、メニュー・アイテムの記述を含みます。

Tipプロパティを使用して、メニュー・アイテムの短い記述を書きます。ヒントはメニュー内のアイテムが選択される時、ステータスバーに表示されます。フォームにステータスバーがない場合、ヒントは表示されません。

TipShow プロパティ

TipShowは、メニュー・アイテムのヒントを表示するかどうかを制御します。

TipShowプロパティを使用して、メニュー・アイテムのヒントをステータスバーに表示するかどうかを制御します。このプロパティをTrueに設定するとヒントが表示されます。

TipShowOfParentプロパティがTrueに設定されると、このプロパティの値と親のTipShowプロパティがTrueに設定されている時のみヒントが表示されることに注意してください。

TipShowOfParent プロパティ

TipShowOfParentは、親のTipShowプロパティを使用するかどうかを制御します。

TipShowOfParentプロパティを使用して、メニュー・アイテムの親のTipShowプロパティを使用するかどうかを制御します。このプロパティをTrueに設定すると、親のTipShowプロパティが使用されます。

基本メニュー

LANSA製品センター内部使用専用です。

[基本メニューのプロパティ](#)

[基本メニューのイベント](#)

基本メニューのプロパティ

TipShow プロパティ

TipShow プロパティ

TipShowは、メニュー・アイテムのヒントを表示するかどうかを制御します。

TipShowプロパティを使用して、メニュー・アイテムのステータスバーにヒントを表示するかどうかを制御します。

このプロパティは、TrueまたはFalseに設定できます。Trueに設定すると、メニュー・アイテムのヒントが表示されます。

基本メニューのイベント

Menu Prepare イベント

Menu Prepare イベント

Prepareイベントは、コンテキスト依存のメニューの表示に使用できません。

Prepareイベントを使用して、メニューを有効にする前に項目を有効化/無効化または挿入/削除します。

Listコントロールでは、Prepareイベントを使用して列見出しやコントロールが右クリックされたかどうかによってポップアップ・メニューを変更します。以下のPrepareイベントは、#GRIDが右クリックされた際に異なるメニュー・アイテムを、列見出し#GRIDCOL1が右クリックされた際に異なる項目を表示します。

```
Evtroutine Handling(#GRIDpopup.Prepare) Options(*noCLEARMESSAGES
*NOCLEARERRORS) Context(#CONTEXT)
If_Ref Com(#CONTEXT) Is(*equal_to #GRID)
Set Com(#GpopupITEM1) Visible(true)
Else
If_Ref Com(#CONTEXT) Is(*equal_to #GRIDCOL1)
Set Com(#GpopupITEM1) Visible(false)
Endif
Endif
Endroutine
```

コントロールに異なるポップアップ・メニューを、列に異なるポップアップ・メニューまたはメニューを定義することでグリッド、ツリー・ビュー、リストにコンテキスト依存のメニューを表示することもできます。

表示エリア (コントロールの項目に使用されない"白い部分") にコンテキスト依存のポップアップ・メニューを表示するには、コントロールのViewpop-upMenuプロパティにその名前を指定してください。

メニューバー

メニュー・アイテムのプライマリ・コンテナです。

メニューを作成するには、最初にメニューバーをフォームにドラッグします。メニューバーは、メニュー・アイテムとサブメニューのコンテナとして機能します。

メニューバーをフォームにドラッグすると、点線の長方形で表示されます。マウスで長方形をクリックします。一次レベルのメニュー・アイテム(メニュー・タイトル)のキャプションを入力できる編集ボックスが表示されます。

長方形を再度クリックすると、別のメニュー・タイトルのキャプションを入力する別の編集ボックスが表示されます。

メニューに従属するメニュー・アイテムを追加するには、タイトルを入力してから下矢印キーを押します。別のメニュー・アイテムのキャプション入力やサブメニュー作成ができる編集エリアが表示されます。

メニュー・エディターを使わずに、メニュー・アイテムやサブメニューをドラッグ・アンド・ドロップしてもメニューを作成することができます。

ポップアップ・メニュー

ポップアップ・メニューは、ユーザーがマウスを右クリックすると表示されます。ポップアップ・メニューを使用して、一番よく使われるオプションに速くアクセスする方法を提供します。

通常ポップアップ・メニューは、リストやツリー・ビューの項目など、デスクトップで選択された項目に適用されます。項目が選択されない場合、フォームにポップアップ・メニューを作成することができます。

コンポーネントのPop-upMenuプロパティを使用して、どのポップアップ・メニューを表示するかを指定します。

同じポップアップ・メニューを複数のコンポーネントで使用することができます。例えばエクスプローラ・タイプのインターフェース(左にツリー・ビュー、右にリスト・ビュー)では、通常1つだけポップアップ・メニューを使用します。

以下も参照してください。

[基本メニューのプロパティ](#)

[基本メニューのイベント](#)

[AutoActions プロパティ](#)

[ポップアップメニューイベント](#)

ポップアップ・メニューのメソッド

Pop-up Show [メソッド](#)

Pop-up Show メソッド

Showメソッドを使用して、ポップアップ・メニューを表示します。ユーザーがポップアップ・メニューに関連付けられたコンポーネントを右クリックすると、自動的にポップアップ・メニューが表示されます。Showメソッドを使用して、別のイベントでポップアップ・メニューを表示することもできます。

以下の例では、Clickイベントによりユーザーの指定した座標にポップアップ・メニューが表示されます。

```
Evtroutine Handling(#BtnTop.Click)
  Invoke Method(#BUTTONpopupMENU.Show) X(#BtnTop.SCREENLEFT) Y
Endroutine
```

X パラメータ

Y パラメータ

X パラメータ

Xパラメータを使用して、ポップアップ・メニューを表示するX座標を指定します。

Y パラメータ

Yパラメータを使用して、ポップアップ・メニューを表示するY座標を指定します。

ポップアップ・メニューのイベント

Prepare イベント

Prepare イベント

Prepareイベントを使用して、コンテキスト依存のポップアップ・メニューを表示することができます。

Prepareイベントを使用して、列見出しやコントロールが右クリックされたかどうかによってポップアップ・メニューを修正します。

以下のPrepareイベントは、#GRIDが右クリックされた際に異なるメニュー・アイテムを、列見出し#GRIDCOL1が右クリックされた際に異なる項目を表示します。

```
EvtRoutine Handling(#GRIDpopup.Prepare) Options(*noCLEARMESSAGES
If_Ref Com(#CONTEXT) Is(*equal_to #GRID)
Set Com(#GpopupITEM1) Visible(true)
Else
If_Ref Com(#CONTEXT) Is(*equal_to #GRIDCOL1)
Set Com(#GpopupITEM1) Visible(false)
Endif
Endif
Endroutine
```

コントロールに異なるポップアップ・メニューを、列に異なるポップアップ・メニューまたはメニューを定義することでグリッド、ツリー・ビュー、リストにコンテキスト依存のメニューを表示することもできます。

表示エリア (コントロールの項目に使用されない"白い部分") にコンテキスト依存のポップアップ・メニューを表示するには、コントロールのViewpop-upMenuプロパティにその名前を指定してください。

以下も参照してください。

[Context パラメータ](#)

Context パラメータ

Contextパラメータは、右クリックされたコントロールや列の名前を引き渡します。

AutoActions プロパティ

ポップアップ・メニューに自動的に追加されるAutoActionsオプションです。

AutoActionsプロパティを使用して、ポップアップ・メニューに自動的に追加されるメニュー・オプションを表示します。

現時点で使用可能なオプションは切り取り、コピー、貼り付けです。フィールドにのみ適用可能です。

オプション・テキストの翻訳の詳細については、『Visual LANSA 開発者ガイド』を参照してください。

Cascading Menu サブメニュー

メニュー・アイテムのコンテナです。

選択されたメニュー・アイテム上にこのコンポーネントをドラッグ・アンド・ドロップしてサブメニューを作成することができます。

または、同じ行でカーソルをアイテムとして移動させ、Shift + Ctrl + 右矢印を押すこともできます。メニュー・タイトルかメニュー・オプションかによって、アイテムの下または右に編集エリアが作成されます。

Tabキーを使用してこのエリアに移動し、サブメニュー・アイテムを入力します。

Base graphic

LANSA製品センター内部使用専用です。

ビットマップ

ビットマップは、リポジトリ内のビットマップ・ファイルです。

ビットマップを使用して、リポジトリ内にコンポーネントとして格納するビットマップ・ファイルを指定します。

リポジトリに登録されたビットマップは、プッシュボタンやツールバー・ボタンで使用できます。プッシュボタン上で、ビットマップに記述をつけることができます。

ビットマップはツールバー・ボタン上での表示と同じように表示されます。ボタンの状態にかかわらず、画像が1つのビットマップも複数のビットマップも使用できます。

画像が1つのビットマップを使用する場合、Designビューの4つのビューには、すべてこの画像が表示されます。

複数の画像を含むビットマップがある場合、ImageCountプロパティを使ってビットマップに含む画像の数を指定する必要があります。最初の画像は選択されていない(押されていない)ボタン上で、2番目の画像は無効にされたボタン上で、3番目の画像は押されたボタン上で使用され、4番目の画像は押されたままの(チェックされている)ボタン上で表示されます。

詳細については、「ImageCountプロパティ」を参照してください。

[ImageCount](#)

[MaskColor](#)

[UseMaskColor](#)

ImageCount

ImageCountは、ビットマップ内の画像の数を指定します。

ImageCountプロパティは、ビットマップが含む画像の数を示します。

ImageCountの値は1から4までです。

最大4つの画像を含むビットマップを利用して、押されていない、無効にされた、押された、チェックされている状態、のボタンの4つの状態を異なる画像で表示することができます。

ビットマップが1つの画像のみを含む場合、ImageCountを1に設定します。実行時、LANSAは異なるボタンの状態で画像がどのように表示されるか、自動的に判断します。Designビューはこれを反映しないため、4つのボタンの表示は同じです。

ビットマップに複数の画像がある場合、ImageCountを画像の数に設定します。最初の画像は、ボタンが選択されていない(押されていない)時、2番目の画像はボタンが無効にされた時に使用され、3番目の画像はボタンが押された時、4番目の画像はボタンが押されたままの(チェックされている)時に表示されます。

ビットマップの幅は、ImageCountの値で正確に割る必要があります。

MaskColor

MaskColorは、透過させる色を設定します。

MaskColorプロパティは、ビットマップ透過的に (マスク) 表示される色を指定します。

例えば、ビットマップのマスクの色に白を指定する場合、ビットマップのすべての白い要素が透過的になり、使われているボタンの色が透けて見えます。

ドロップダウン・リストの値を使用してこのプロパティを標準色 (赤など) に設定するか、特定のワークステーションのWindowsオブジェクトに使われている色に設定します。例えば色をButtonFaceに設定すると、マスクの色はワークステーション上でボタン表面の色に使われている色と同じになります。

または省略記号 (...) のボタンを使用してカラー・パレットを表示し、カスタム色を指定します。カスタム色は、RGB (Red, GreenおよびBlue) 値で表示されます。

UseMaskColor

UseMaskColorは、マスクの色を使うかどうかを制御します。

UseMaskColorプロパティは、MaskColorプロパティで指定したマスクの色を使うかどうかを指定します。

アイコン

Iconはリポジトリ内に格納されるアイコン・ファイルです。

アイコンのコンポーネントは、アイコン・ファイルをリポジトリ内に格納する際に使用されます。

リポジトリに登録されたアイコンは、ツリー・ビュー、リスト・ビューおよびフォームで使用できます。

ツリー・ビューとリスト・ビューにおいて、アイコンは各リスト項目を表示します。アイコンを項目の前に表示してそのタイプを示すことができます (Imageプロパティ)。リスト項目の状態を示すアイコンを表示することもできます (ImageState)。ツリー・ビューでは、展開された項目のアイコンも指定できます (ImageExpanded)。

ツリー・ビューとリスト・ビューには、ImageOverlayプロパティもあり、Imageプロパティで指定したアイコンの上に別のアイコンを重ねることができます。

フォーム内では、Iconプロパティを使ってアイコンを指定します。フォームの左上の角とタスクバー上に表示されます。

アイコンを登録すると、アイコンの4つのビューが表示されます。

小さいステートのアイコンは、リスト・ビューのVisualStyleがIconに設定される場合を除き、ツリー・ビューやリスト・ビューのImageStateプロパティの値として使われた場合、アイコンがどのように表示されるかを示します。

大きいステートのアイコンは、リスト・ビューのVisualStyleプロパティがIconに設定され、アイコンがリスト・ビューのImageStateプロパティの値として使われた場合、どのように表示されるかを示します。

小さいアイコン・ビューは、アイコンが以下の値として使われた場合、どのように表示されるかを示します：

- ツリー・ビューの Image, ImageExpanded およびImageOverlayプロパティ
- リスト・ビューのImageやImageOverlayプロパティ (リスト・ビューのVisualStyleプロパティがIconに設定される場合を除く)
- フォームのIconプロパティ

大きいアイコン・ビューは、リストのVisualStyleプロパティがIconに設定され、アイコンがリスト・ビューのImageやImageOverlayプロパティの値として使われた場合、どのように表示されるかを示します。

StandardIcon プロパティ

StandardIcon プロパティ

StandardIconは、標準アイコンをリストにします。

StandardIcon プロパティを使用して、WindowsシステムアイコンをLANSAリポジトリに登録します。

StandardIconとFileName プロパティは、相互に排他的です。ファイル名とパスまたは標準のWindowsアイコン名を指定することにより、このコンポーネントに格納する画像を指定することができます。

カーソル

カーソル・イメージは、マウス・ポインターと関連付けられています。カーソルを使用して、図形とマウス・ポインターを関連付けます。ユーザーがマウス・ポインターを移動させると、外観が変化し、状態や操作についてのフィードバックを提供します。図形を指定するには、FileNameプロパティを使用するかStandardCursorリストからグラフィックを選択します。

カーソルを作成後、コンポーネントのCursorプロパティを使用して他のコンポーネントと関連付けます。

[HotspotX プロパティ](#)

[HotspotY プロパティ](#)

[StandardCursor プロパティ](#)

HotspotX プロパティ

HotspotXとHotspotYは、カーソルのアクティブ領域を設定します。

HotspotXは、オブジェクトを示すのに使われるカーソルの部分をHotspotYと共に指定します。

ホットスポットを直感的にします。例えば矢印のカーソルの先端をホットスポットにします。

HotspotY プロパティ

HotspotYとHotspotXは、カーソルのアクティブ領域を設定します。

HotspotYは、オブジェクトを示すのに使われるカーソルの部分をHotspotXと共に指定します。

ホットスポットを直感的にします。例えば矢印のカーソルの先端をホットスポットにします。

StandardCursor プロパティ

StandardCursorは、標準カーソルをリストにします。

StandardCursorは、標準カーソルをリストにします。

エレメント・ビジュアル・パート

エレメント・ビジュアル・パートのプロパティ

エレメント・ビジュアル・パートのプロパティ

HintType プロパティ

LabelHorAlignment プロパティ

LabelPosition プロパティ

LabelText プロパティ

LabelType プロパティ

LabelVerAlignment プロパティ

MarginBottom プロパティ

MarginLeft プロパティ

MarginRight プロパティ

MarginTop プロパティ

SetValue プロパティ

HintType プロパティ

HintTypeは、使用されるヒントのタイプを指定します。

HintTypeを使用して、ヒントに表示するテキストを指定します。

Captionを指定すると、Hintプロパティに入力したテキストがヒントとして使用されます。フィールド定義の欄見出し、記述やラベルのテキストをヒントとして使用することもできます。

MarginBottom プロパティ

MarginBottomは、フィールドの下の余白を設定します。

MarginBottomプロパティを使用して、フィールドの下の余白を指定します。通常フィールドのマージンを調整する必要はありませんが、アプリケーションの外観を微調整するために、マージンの調整が必要になる場合があります。

このプロパティは、アプリケーションの実行中ではなく設計時にのみ指定が可能です。

MarginLeft プロパティ

MarginLeftは、フィールドの左の余白を設定します。

MarginLeftプロパティを使用して、フィールドの左の余白を指定します。通常フィールドのマージンを調整する必要はありませんが、アプリケーションの外観を微調整するために、マージンの調整が必要になる場合があります。

このプロパティは、アプリケーションの実行中ではなく設計時にのみ指定が可能です。

MarginRight プロパティ

MarginRightは、フィールドの右の余白を設定します。

MarginRight プロパティを使用して、フィールドの右の余白を指定します。通常フィールドのマージンを調整する必要はありませんが、アプリケーションの外観を微調整するために、マージンの調整が必要になる場合があります。

このプロパティは、アプリケーションの実行中ではなく設計時にのみ指定が可能です。

MarginTop プロパティ

MarginTopは、フィールドの上の余白を設定します。

MarginTopプロパティを使用して、フィールドの上の余白を指定します。通常フィールドのマージンを調整する必要はありませんが、アプリケーションの外観を微調整するために、マージンの調整が必要になる場合があります。

このプロパティは、アプリケーションの実行中ではなく設計時にのみ指定が可能です。

LabelPosition プロパティ

LabelPositionはラベルの位置を設定します。

LabelPositionプロパティを利用して、フィールドに関するラベルの位置を設定します。ラベルは、フィールドの左、右、上または下に表示できます。

このプロパティは、アプリケーションの実行中ではなく設計時にのみ指定が可能です。

LabelType プロパティ

LabelTypeは、ラベルのタイプを設定します。

LabelTypeプロパティを利用して、ラベルのタイプを設定します。
フィールド定義に指定された列見出し、記述やラベル、またはフィールドのCaptionプロパティに入力されたキャプションを使用できます。

一貫性と保守の容易性を保つため、必ずフィールド定義から取り出したラベルを使用することをおすすめします。

LabelHorAlignment プロパティ

LabelHorAlignmentは、ラベルを水平に配置します。

LabelHorAlignmentプロパティを利用して、ラベルの水平方向の整列を設定します。ラベルは、右、中央または左揃えにすることができます。

このプロパティは、アプリケーションの実行中ではなく設計時にのみ指定が可能です。

LabelVerAlignment プロパティ

LabelHorAlignmentは、ラベルを垂直に配置します。

LabelHorAlignmentプロパティを使用して、ラベルの垂直方向の配置を設定します。

このプロパティは、アプリケーションの実行中ではなく設計時にのみ指定が可能です。

LabelText プロパティ

LabelType プロパティを発行する PRIM_EVP は、LabelText プロパティも発行します。

例

以下の例では、"Label Here"文字列を取り出します。

```
Define_Com Class(#STD_TEXTL.Visual) Name(#OutputValue) Caption('Lab
```

```
#OutputValue.LabelText
```

SetValue プロパティ

SetValue()メソッドを使用して、PRIM_EVPおよびPRIM_EVEF Primitive型に値を割り当てることができます。

提供された値が対象エレメントのデータ・タイプでは許可されない場合(例：数値タイプに文字列"abc"を指定)、SetValueはLP_FALSEを返します。正しく割り当てることができた場合は、LP_TRUEが返されます。

結果

設定が成功 (True) したかどうかを示すブール値です。

値

設定する値

例

以下の例では、さまざまな#OutputValueNが#STD_TEXTL.Visualのインスタンスになります。

```
Define_Com Class(#PRIM_EVP) Name(#evpTmp) Reference(*DYNAMIC)
```

```
If (#STD_NUM.SetValue( "a.bc" ) <> True)  
#OutputValue1 := "not a number"  
Endif
```

```
#evpTmp <= #OutputValue2  
#evpTmp.SetValue( "def" )
```

ピックリスト

ピックリスト・アイテム

ピックリストのプロパティ

ピックアップリスト・アイテム

ピックアップリスト・アイテムは、ピックアップリストで使われます。

ピックアップリスト・アイテムは、ピックアップリストで使われます。

[ピックアップリスト・アイテム記述](#)

[省略値のピックアップリスト・アイテム](#)

[ピックアップリスト・アイテムの値](#)

[ピックアップリスト・データ・クラス](#)

ピックアップ・アイテム記述

ピックアップ・アイテムの記述はアイテム内に表示されるテキストです。

省略値のピックアップリスト・アイテム

Defaultプロパティは、それがピックアップリストの省略値アイテムであることを指定します。このプロパティは、TrueまたはFalseに設定できます。

ピックリスト・アイテムの値

このプロパティを使用して、このピックリスト・アイテムが選択された時に返される値を設定します。

ピックリスト・アイテムに表示されるテキストを設定するには、Captionプロパティを使用します。

ピックリスト・データ・クラス

ピックリストは、プロパティ・シート内にあるリストです。

ピックアップのプロパティ

NoMatchAction プロパティ

NoMatchAction プロパティ

NoMatchActionは、リストで値が見つからない時に実行されるアクションです。

NoMatchActionを使用して、ピックリストに誤った値が提供された時に実行されるアクションを設定します。

ピックリストの値は、その項目の値の1つでなければなりません。例えばフィールド#SexのピックリストにMaleとFemaleの値を持った2つの項目がある場合、この2つの値のみがピックリストに指定できます。

Changeステートメントがフィールドの値を 'Unknown'などに変更する場合、ピックリストのNoMatchActionプロパティが、結果を制御します。

NoMatchアクションは、以下のいずれかになります。

Blanks:ピックリストの値をブランクに設定します。

DefaultItem:省略値 (True) を持つピックリスト項目を選択します。

ShowValue:ピックリスト・ビジュアルイゼーションで表示できる場合は、入力した値を表示します。

Alphanumeric データ・クラス PRIM_ALPH

Alphanumericデータ・クラスは、文字列です。

省略値では、このデータ・クラスの長さは0で、最大長はないことを示している点に注意してください。最大長を指定する場合、フィールドにデータ・クラスの値を設定する際は注意してください。フィールドの最大長は255です。

[文字列のケース](#)

[文字列の長さ](#)

[データ・クラスに関連付けられたピックリスト](#)

文字列のケース
文字列のケースです。

文字列の長さ

文字列の長さです。

データ・クラスに関連付けられたピックリスト

データ・クラスに関連付けられたピックリストです。

ピックリストは、エントリーの値のリストを表示するプロパティ・シートに使われます。

ピックリストの使用法の詳細については、『Visual LANSA 開発者ガイド』を参照してください。

Booleanデータ・クラス PRIM_BOLN

ブール型データ・クラスは、ActiveXコンポーネントと共に使用されま
す。

Currencyデータ・クラス PRIM_CRCY
通貨型データ・クラス

Dateデータ・クラス **PRIM_DATE**

日付型データ・クラス

DateTimeデータ・クラス **PRIM_DAT**

日付時刻型データ・クラス

Numberデータ・クラス **PRIM_NMBR**

数値型データ・クラス

小数桁数

DisplayFormatが編集コードを設定

数値の長さ

データ・クラスに関連付けられたピックリスト

小数桁数

小数の桁数です。

DisplayFormatが編集コードを設定

DisplayFormatは、編集コードを設定します。

数値の長さ

数値の長さです。

データ・クラスに関連付けられたピックリスト

データ・クラスに関連付けられたピックリストです。

ピックリストは、エントリーの値のリストを表示するプロパティ・シートに使われます。

ピックリストの使用法の詳細については、『Visual LANSA 開発者ガイド』を参照してください。

Numeric String データ・クラス **PRIM_NSTR**
数値文字列型データ・クラス

Timeデータ・クラス **PRIM_TIME**

時刻型データ・クラス

Variant コンポーネント

Variant プロパティ

Variant メソッド

Variant コンポーネントは、バリエーション値の格納に使用できます。

Variant コンポーネント・クラス #PRIM_VAR を使用してバリエーション・コンポーネント変数を作ることができます。Variant コンポーネント変数は、どのようなタイプのデータも含むことができます(文字列、整数、小数、ブール値、コンポーネント)。

Variant コンポーネントには、コンポーネントに含まれる値の種類を確認するプロパティがあります。

値は、数値、文字列、またはブール値などの変換された形式で取得することができます。バリエーションを使えば、コンポーネントは、静的に判別不可能なタイプの値にアクセスできるようになります。例えば、グリッド内のセルの値を静的に定義することはできません。それはセルの列のタイプに依存するからです。グリッド制御は、バリエーションを返してセル値へのアクセスを可能にします。宣言された値のタイプに関するコンパイル・エラーは発生しません。この場合は、グリッドの値のタイプを識別できるようにプログラムを記述することが必要です。

バリエーションは動的プログラミングにも使用されます。コンパイル時にコンポーネントのタイプ情報を決定できない場合 (特に ActiveX) があります。この場合、実行時に処理する必要があります。バリエーションを使用する場合、Visual LANSA コンパイラはメソッドやプロパティの解決を試みません。バリエーションに格納されているコンポーネントのタイプがわからないためです。このすべての情報を解決するには実行時まで待つ必要があります。この動作により、ActiveX コントロールは不明なタイプのコンポーネントを提供することができます。そして、このコンポーネントが呼び出された時にだけ、タイプ情報が要求されます。

以下のステートメントでは、#MYVARIANT と呼ばれるバリエーション型変数を定義します。

```
Define_Com Class(#PRIM_VAR) Name(#MYVARIANT)
```

バリエーション型変数には、その Value プロパティを使用して値を割り当てるすることができます (この場合、値のタイプは不明なタイプです)。

RDML コマンドでこの変数を使用する場合、明示的に値のタイプを割り当てる方法が必要です。例えば、変数にタイプを割り当てなければ

CHANGEとIFステートメントを使用できません。RDMLXコマンドでは、タイプを不明とすることができます。

変数値を読み取るときに、値は自動的に値を受け取るフィールドのタイプに変換されます。

```
Set Com(#Out_INTEGER) Value(#myvariant)
```

また、値のタイプを明示的に指定することもできます。

```
Set Com(#Out_INTEGER) Value(#myvariant.Integer)
```

バリエーションの値をサポート可能な値とするため、変数のValueTypeプロパティを使用します。

または、バリエーション・データ型の変数を作成することができます。

```
Define_Com Class(*Variant) Name(#lclVariant)
```

バリエーション変数の作業を行う場合は、バリエーション関数を使用します。

Variant プロパティ

Boolean プロパティ

Component プロパティ

Decimal プロパティ

String プロパティ

Integer プロパティ

ValueType プロパティ

Boolean プロパティ

Booleanは変数の値をブール値とします。

Booleanプロパティを使用して、変数にブール値を割り当てたり変数値をブール値として取り出したりすることができます。ブール値は、TrueまたはFalseになります。

以下のコードは、チェック・ボックスが選択されている場合、Booleanプロパティ #MYVARIABLE をTrueに設定します。

```
If Cond('#CHECKBOX.ButtonState = Checked')
```

```
Set Com(#myvariant) Boolean(True)
```

```
Else
```

```
Set Com(#myvariant) Boolean(False)
```

```
Endif
```

#MYVARIABLEのBooleanプロパティがTrueの場合、チェック・ボックスを選択します。:

```
If Cond('#myvariant.Boolean = True')
```

```
Set Com('#CHECKBOX) Buttonstate(Checked)
```

```
Else
```

```
Set Com('#CHECKBOX) Buttonstate(Unchecked)
```

```
Endif
```

Component プロパティ

Componentは変数の値をコンポーネント参照とします。

Component プロパティを使用して、変数値にコンポーネント参照を割り当てたり変数値をコンポーネント参照として取り出したりすることができます。

バリエーション変数を使って参照を設定すると、メソッドが呼び出され、Visual LANSAコンパイラの確認なしにプロパティ SETとGETのステートメントが実行されます。

例えば、存在しないメソッドを呼び出すと、コンパイル・エラーが発生します。

```
Invoke #Com_Owner.MyMethod
```

しかし、コンポーネントのオーナーがバリエーション変数に割り当てられると、変数の内容は不明なため、変数の存在しないメソッドが呼び出されてもコンパイル・エラーは引き起こされません。

```
Set #myvariant Component(#Com_Owner)
```

```
Invoke #myvariant.Value.MyMethod
```

実行時に#myvariantの現在の値が検証されます。#myvariantの現在の値がコンポーネントの場合、MyMethodというメソッドを求めてコンポーネント参照のComponentTypeにアクセスします。存在する場合、メソッドが実行されます。存在しない場合、実行時エラーが起きます。

Decimal プロパティ

Decimalは、変数の値を10進値とします。

Decimalプロパティを使用して、変数に10進値を割り当てたり変数値を10進数として取り出したりすることができます。

以下のコードは、変数に10進数を割り当てます。

```
Set Com(#myvariant) Decimal(#XYZ)
```

以下のコードは、変数値を10進数として読み取ります。

```
Set Com(#XYZ) Value(#myvariant.Decimal)
```

String プロパティ

Stringは、変数の値を文字列とします。

Stringプロパティを使用して、変数に文字列値を割り当てたり変数値を文字列として取り出したりすることができます。

以下のコードは、変数に文字列の値を割り当てます。

```
Set Com(#myvariant) String(#XYZ)
```

以下のコードは、変数値を文字列として読み取ります。

```
Set Com(#XYZ) Value(#myvariant.String)
```

Integer プロパティ

Integerは、変数の値を整数値とします。

Integerプロパティを使用して、変数に整数値を割り当てたり変数値を整数として取り出したりすることができます。

以下のコードは、変数値に整数を割り当てます。

```
Set Com(#myvariant) Integer(#XYZ)
```

以下のコードは、変数値を整数として読み取ります。

```
Set Com(#XYZ) Value(#myvariant.Integer)
```

ValueType プロパティ

ValueTypeは、変数に含まれる値のタイプを確認します。

ValueTypeプロパティを使用して、変数に含まれる値のタイプを確認します。

以下のコードは、変数に含まれる値が整数かどうかを確認します。

```
If Cond('#myvariant.ValueType = VarInteger')
  Set Com(#Out_INTEGER) Value(#myvariant)
Else
  Set Com(#Out_INTEGER) Value(0)
Endif
```

この確認が実行されず、実行時に値が変換できない場合は、実行時エラーになります。

ValueTypeプロパティは、列挙型です。値は、以下のいずれかの記号になります。

- varNull
- varEmpty
- varInteger
- varDouble
- varString
- varDecimal
- varBoolean
- varComponent

Variant メソッド

AssignNull メソッド

AssignEmpty メソッド

AssignNull メソッド

AssignNullは、変数にnull値を割り当てます。

AssignNullメソッドを使用して、変数にnull値を割り当てます。

Nullは、伝播null値で、SQLの3値ロジックで使用されます。Null値は、ActiveXコンポーネントから渡すこともできます。

AssignEmpty メソッド

AssignEmptyは、変数に値を割り当てません。

変数に値を割り当てない時は、AssignEmptyメソッドを使います。

Data class コンポーネント

Data class コンポーネントは、値を保持し記述します。

Decimals プロパティ

Length プロパティ

Decimals プロパティ

プロパティは、フィールド定義のフィールドに規定する小数点以下の桁数を示します。読み取り専用プロパティです。

Length プロパティ

Lengthプロパティは、フィールド定義のフィールドに指定する長さを示します。読み取り専用プロパティです。

Com エラー

ActiveXコンポーネントとの最後の対話処理で返された情報を含みません。

ActiveXコンポーネントとの対話処理の成功や失敗は、ActiveX から HRESULTデータ・タイプで返される変数値で示されます。HRESULT変数は、以下の3つのいずれかに解釈される32ビットの整数を表します。

- 1 ゼロの場合は、対話処理の成功を意味します。
- 2 ゼロ以外の正数値が返される場合は、対話処理が成功しているがステータス情報が返されています。
- 3 負数値が返された場合は、対話処理が失敗してエラー情報が返されています。

#COM_ERR_INFOという変数を使うとActiveXから返された情報にプログラム上でアクセスできます。

#COM_ERR_INFOコンポーネントのIsErrorプロパティは、対話処理が失敗したかどうかを示し、ErrorCodeとErrorWordプロパティは、返されたHRESULT値を数字と文字列で表示して提供します。

Com エラーのプロパティ

Error Code プロパティ

ErrorWord プロパティ

HelpContext プロパティ

HelpFile プロパティ

IsError プロパティ

Error Code プロパティ

現在のHRESULTを32ビットの符号付き整数で返します。

サンプル値については、ErrorWordプロパティを参照してください。

ErrorWord プロパティ

現在のHRESULT値をXXXXXXXXXXの形の16進数文字列で返します。各 'x' は0から9までと'A'から'F'までの16進数文字で構成されます。

以下は、HRESULT値と一般に認められた記号名の例です。

- 1 E_ACCESSDENIED – 一般的なアクセス拒否エラーです。

ErrorWord:X80070005

ErrorCode:-2147024891

- 2 E_FAIL – 不明なエラーが発生しました。

ErrorWord:X80040005

ErrorCode:-2147467259

- 3 E_INVALIDARG – 1つ以上の引数が無効です。

ErrorWord:X80070057

ErrorCode:-2147024809

- 4 E_UNEXPECTED – 突発的エラーが発生しました。

ErrorWord:X8000FFFF

ErrorCode:-2147418113

- 5 S_FALSE – メソッドが成功しブール値FSALSEが返されました。

ErrorWord:X00000001

ErrorCode:1

- 6 S_OK – メソッドが成功しました。ブール戻り値がある場合、戻り値はTRUEです。

ErrorWord:X00000000

ErrorCode:0

HelpContext プロパティ

HelpContext プロパティは、失敗した HRESULT のヘルプ・テキストのコンテキスト識別子を 32 ビットの整数で返します。このコンテキスト識別子を使用して、HelpFile プロパティにより提供されるヘルプ・ファイルのヘルプ・テキストを確認することができます。

このプロパティは、エラー発生時にこの情報を提供する ActiveX コントロールに依存します。

HelpFile プロパティ

HelpFileプロパティは、失敗したHRESULTのヘルプ・テキストを表示できるウィンドウのヘルプ・ファイル名を文字列で返します。

このプロパティは、エラー発生時にこの情報を提供するActiveXコントロールに依存します。

IsError プロパティ

IsErrorは、現在のHRESULTが失敗かどうかを示す値を返します。

IsErrorは、HRESULTが負の正数値であればTrueを、それ以外はFalseを返します。

Com エラーのメソッド

Clearメソッド

ReportErrorメソッド

ReportErrorAndAbortメソッド

ShowErrorHelpメソッド

Clear メソッド

Clearメソッドは、HRESULTが0の状態に対応するCom Errorコンポーネントの状態をリセットします。

ReportError メソッド

このメソッドは、現在の Com Error状態をメッセージボックスでレポートします。

ReportErrorAndAbort メソッド

このメソッドは、アプリケーションを中止する前に現在の Com Error 状態をメッセージボックスでレポートします。

ShowErrorHelp メソッド

ActiveXが現在のエラーのヘルプの表示に必要な情報を提供すると、ヘルプ・ウィンドウが開きます。

Application コンポーネント

Applicationコンポーネントです。

Mouse コンポーネント

Mouse コンポーネントは、マウスの位置を示します。

Mouse コンポーネントは、マウスカーソルの位置を示します。

定義済みの変数SYS_MOUSEは、Mouse コンポーネントに基づいています。この変数のプロパティを使うと、マウスの位置を最も簡単に確認できます。

Hor Position: 水平位置を示す左からのピクセル数です。

VerPosition: 垂直位置を示す上からのピクセル数です。

例えば、フィールドに水平位置の値を割り当てることができます：

```
change #Std_Num #SYS_MOUSE.HorPosition
```

Mouse イベントのパラメータ

PosX パラメータ

PosY パラメータ

IsAltDown パラメータ

IsControlDown パラメータ

IsShiftDown パラメータ

IsLeftButtonDown パラメータ

IsMiddleButtonDown パラメータ

IsRightButtonDown パラメータ

PosX パラメータ

PosXは、X座標に対するマウスの位置を示します。

PosY パラメータ

PosYは、Y座標に対するマウスの位置を示します。

IsAltDown パラメータ

IsAltDownは、Altキーが押されたかどうかを示します。

IsControlDown パラメータ

IsControlDownは、Controlキーが押されたかどうかを示します。

IsShiftDown パラメータ

IsShiftDownは、Shiftキーが押されたかどうかを示します。

IsLeftButtonDown パラメータ

IsLeftButtonDownは、マウスの左ボタンが押されたかどうかを示します。

IsMiddleButtonDown パラメータ

IsMiddleButtonDownは、マウスの中央ボタンが押されたかどうかを示します。

IsRightButtonDown パラメータ

IsRightButtonDownは、マウスの右ボタンが押されたかどうかを示します。

Keyboard コンポーネント

Keyboardコンポーネントは、Shift、Alt、またはCtrlキーが押されたかどうかを示します。

Keyboardコンポーネントは、Shift、Alt、またはCtrlキーが押されたかどうかを示します。

事前定義された変数SYS_KEYBD は、Keyboardコンポーネントに基づいています。この変数のプロパティを使うと、最も簡単にAlt、Ctrl、またはShiftキーが押されたかどうかを確認できます。

AltKeyDownがTrueまたはFalse。Altキーが押されたかどうかを示します。

ShiftKeyDownがTrueまたはFalse。Shiftキーが押されたかどうかを示します。

ControlKeyDownがTrueまたはFalse。Controlキーが押されたかどうかを示します。

以下のように変数を使うことができます。

```
if cond'(#sys_keybd.AltKeyDown *eq True)...
```

プロパティ・シート

オブジェクトのプロパティを表示します。

プロパティ・シートを使用して、オブジェクトのプロパティを表示します。エディターの[詳細]タブはプロパティ・シートの例です。

通常、プロパティ・シートには、プロパティの記述とプロパティの値の2つの列があります。値は単純な値か値の一覧(ピックリスト)のどちらかです。

すべてのリスト・タイプのコンポーネントで、プロパティ・シート・コンポーネントをフォームまたは再利用可能パーツにドラッグし、プロパティ・シートの作成を開始します。次にリポジトリからフィールドをドラッグしてプロパティ・シートに列を追加します。

通常、プロパティ・シートには、プロパティの記述とプロパティの値の2つの列を定義します。プロパティ・シートでは、(リストと異なり)列はデータのフィールドに基づきません。プロパティ・シートの各エントリーは、異なるフィールドに基づく異なる属性を表示するからです。

プロパティ・シートにエントリーを追加するには：

- 1 プロパティ・シートのInitializeイベントで、エントリーの記述と値を指定します(以下の例では1列目はフィールド#PROPERTYに基づき、2列目はフィールド#VALUEに基づきます)。

```
CHANGE field(#PROPERTY) to("First Name")
CHANGE field(#VALUE) to(#GIVENAME)
```

- 2 エントリーをプロパティ・シートに追加します。

```
ADD_ENTRY to_list(#PROP_1)
```

- 3 (コンポーネントの先頭に) エントリーのデータ・クラスを定義します。

```
DEFINE_COM class(#GIVENAME) name(#GIVENAME)
```

次にInitializeイベント・ルーチンで、エントリーにデータ・クラスを割り当てます。データ・クラスは、値の表示と入力方法进行处理します。

```
SET com(#prop_1.CurrentItem) DATACLASS(#givename)
```

このステートメントは、プロパティ・シートの2列目の現在のエント

リーの値を#GiveNameクラスに従って表示 (および編集) するよう、プロパティ・シートに指示します。例えば、最大長は20文字で、大文字または小文字の文字を含むことができます。

プロパティ・シートにエントリーを追加するたびにこの基本的なステップを繰り返します。

プロパティ・シートの使用法の例については、『Visual LANSA 開発者ガイド』を参照してください。

プロパティ・シート・カラム

[↑プロパティ・シート](#)

プロパティ・シート項目

[プロパティ・シート項目 Ancestor](#) [プロパティ](#)

[↑プロパティ・シート](#)

プロパティ・シート項目 Ancestor プロパティ

Ancestor プロパティは、このコンポーネントの継承元のコンポーネントを示します。

Ancestor プロパティを使用し、このコンポーネントの継承元のコンポーネントを選択します。

省略値では、すべてのフォームは基本の Visual LANSA フォーム #PRIM_FORM を継承します。#PRIM_FORM は変更できません。他のどんなフォームでも祖先のフォームに指定することができます。継承先のフォームは、祖先のすべてのコンポーネントを継承します。祖先のフォームを変更すると、変更は継承先のフォームに反映されます。

継承元のフォームとそのコンポーネントのプロパティを変更できます。継承元のコンポーネントのプロパティを変更すると、永続的に元の値を上書きします。そのため、キャプションが [保存] のプッシュ・ボタンを含む祖先のフォームがあり、これを継承先のフォームで [変更を保存] に変更した場合、このキャプションは祖先のフォームのボタンのキャプションを変更しても変更されません。

省略値では、再利用可能パーツは、他のコンポーネントを置くことができるパネル #PRIM_PANEL を継承します。ボタン、タブ・フォルダ、ツリー・ビューなど、どんな Visual LANSA コントロールでも再利用可能パーツの祖先にすることができます。ある再利用可能パーツを他の再利用可能パーツの祖先として使用することもできます。

[DefaultProperty プロパティ](#)

[Public プロパティ](#)

[Private プロパティ](#)

[Protect プロパティ](#)

[Framework プロパティ](#)

[Group プロパティ](#)

[Description プロパティ](#)

DefaultProperty プロパティ

DefaultProperty プロパティは、コンポーネントの省略値プロパティを定義します。

DefaultProperty プロパティを使用して、コンポーネントの省略値を設定します。そうすると、コンポーネント名だけで省略値プロパティにアクセスできるようになります。

例えば#VL_DEM57コンポーネントの省略値プロパティがCaption プロパティである場合、コンポーネント名を参照するだけでコンポーネントにアクセスできます。

```
Change Field(#STD_TEXTS) To(#VL_DEM57)
```

明確にCaption プロパティを参照するのと同じです。

```
Change Field(#STD_TEXTS) To('#VL_DEM57.caption')
```

Public プロパティ

Public プロパティは、非表示のカスタム定義のイベント、メソッドまたはプロパティを表示します。

フォームまたは再利用可能パーツのPublic プロパティを使用して、祖先のカスタム定義で保護されていたイベント、メソッドまたはプロパティを公開します。

フォームまたは再利用可能パーツは、Protect プロパティを使用して、カスタム定義のイベント、メソッド、プロパティを外部からアクセスできないように定義することができます。定義により保護されたイベント、メソッド、プロパティは、継承先のフォームおよび再利用可能パーツには表示可能ですが、オーナーのフォームまたは再利用可能パーツには表示されません。継承元のフォームまたは再利用可能パーツは、継承元の保護されたイベント、メソッド、プロパティをPublicと定義することにより、他のすべてのコンポーネントで表示することができます。

Publicの値リストは、CreateInstanceおよびDestroyInstanceという2つの内部システム値を表示します。これらの値は使用できません。

Private プロパティ

Private プロパティは、カスタム定義のイベント、メソッド、プロパティを非表示にします。

フォームまたは再利用可能パーツのPrivate プロパティを使用して、カスタム定義のイベント、メソッド、プロパティを他のフォームまたは再利用可能パーツで非表示にできます。Private と定義されるイベント、メソッド、プロパティは、継承先または所有する再利用可能パーツおよびフォームでは表示されません。

Protect プロパティ

Protect プロパティは、カスタム定義のイベント、メソッド、プロパティを部分的に非表示にします。

このプロパティを使用して、オーナーの再利用可能パーツまたはフォームでカスタム定義のイベント、プロパティ、メソッドを非表示にします。定義により保護されたイベント、メソッド、プロパティは、継承元の再利用可能パーツおよびフォームで表示可能になります。これにより保護されたイベント、プロパティ、メソッドがPublicと定義され、オーナーに見えるようになります。

Framework プロパティ

フレームワークはビジネスです。

フレームワークは、ビジネス向けの項目グループです。例えば、LANSA 人事デモ1・アプリケーションの作成に使用されるすべてのコンポーネントは、人材フレームワークに存在します。

製造や経営情報などもっと汎用的なフレームワークがあり、このようなアプリケーションで通常必要なコンポーネントの格納に使われています。

独自のフレームワークを作成する時は、LANSA/AD内の [システム保守メニュー] の [システム区画定義の作成/変更] オプションを使います。新しいフレームワークを作成したら、LANSA for Windowsの区画の詳細を更新してください。

コンポーネントを作成する際、格納するフレームワークを指定することができます。エディターでフレームワークを変更することもできます。

Group プロパティ

Groupは、コンポーネントを格納するユーザー定義のグループです。

グループとは、開発向けの類似項目をひとつにまとめる方法です。例えば、標準ボタン、メニューおよびフィールドの省略値グループがあります。

ユーザーがグループを定義することもできますが、LANSAには標準セットのグループが組み込まれています。コンポーネントは、1つ以上のグループに属することができます。

独自のグループを作成する時は、LANSA/AD内の [システム保守メニュー] の [システム区画定義の作成/変更] オプションを使います。新しいグループを作成したら、LANSA for Windowsの区画の詳細を更新してください。コンポーネントを作成する時に、格納するグループを指定することができます。エディターでグループを変更することもできます。

Description プロパティ

コンポーネントの記述です。

Descriptionプロパティは、コンポーネントに関連付けられた記述を指定する際に使用されます。この記述は、コンポーネントのユーザーの識別を支援します。

基本レイアウト

表示されません。

Manage プロパティ

Parent プロパティ

MarginTop プロパティ

MarginLeft プロパティ

MarginRight プロパティ

MarginBottom プロパティ

Manage プロパティ

制御対象のビジュアル・コンポーネントを定義します。

Parent プロパティ

項目のレイアウト・マネージャ・コンポーネントを定義します。

MarginTop プロパティ
上の余白を定義します。

MarginLeft プロパティ

左の余白を定義します。

MarginRight プロパティ

右の余白を定義します。

MarginBottom プロパティ

下の余白を定義します。

基本レイアウト・マネージャ

基本レイアウト・マネージャです。

LANSA製品センター内部使用専用です。

MaxHeight プロパティ

MinHeight プロパティ

MaxWidth プロパティ

MinWidth プロパティ

MaxHeight プロパティ

MaxHeight プロパティは、レイアウト・マネージャの最大高を設定します。

MinHeight プロパティ

MinHeight プロパティは、レイアウト・マネージャの最小高を設定します。

MaxWidth プロパティ

MaxWidthプロパティは、レイアウト・マネージャの最大幅を設定します。

MinWidth プロパティ

MinWidthプロパティは、レイアウト・マネージャの最小幅を設定します。

基本レイアウト・アイテム

LANSA製品センター内部使用専用です。

添付 レイアウト・マネージャ

添付レイアウトは、コンテナを5つのセルに分割します。中央のセルは、使用可能なスペースをすべて埋めるよう広がります。

[ProcessingOrder](#) プロパティ

[添付レイアウト・アイテム](#)

ProcessingOrder プロパティ

添付レイアウト・アイテムの処理命令に使われるスキーマを定義します。

垂直：項目は、上、下、左、右の順に添付されます。中央に添付された項目は、最後に処理されます。

水平：項目は、左、右、上、下の順に添付されます。中央に添付された項目は、最後に処理されます。

表示位置：上、下、左、右に添付された項目は、表示位置順に処理されます。中央に添付された項目は、最後に処理されます。

省略値は垂直です。

添付レイアウト・アイテム

Attachment プロパティ

Percentage プロパティ

Attachment プロパティ

コンポーネントを添付する場所を定義します。

None:添付されません。レイアウト・マネージャはコンポーネントを制御しません。

Top:親の表示エリアの上部にコンポーネントを添付します。

Bottom:親の表示エリアの下部にコンポーネントを添付します。

Left:親の表示エリアの左側にコンポーネントを添付します。

Right:親の表示エリアの右側にコンポーネントを添付します。

Center:親の表示エリアの中央にコンポーネントを添付します。

Attachment プロパティを処理する場合、項目は、その親の添付レイアウトコンポーネントのProcessingOrder プロパティの設定に基づいて添付されます。

Percentage プロパティ

コンポーネントのサイズ調整に使うパーセンテージを定義します。

分割レイアウト・マネージャ

分割レイアウト・マネージャは、子コンポーネントが表示エリアの水平方向または垂直方向に分割されるよう、コンポーネントの表示エリアを制御します。Orientationプロパティは、分割の方向を指定します。オプションとして、コンポーネントのサイズが変更されると、レイアウト・マネージャが制御する項目のサイズを自動的に変更し、サイズ調整分割線を表示して、ユーザーが自分の好みに合わせて分割を調整できるようにします。

[Orientation プロパティ](#)

[分割レイアウト・マネージャ](#)

Orientation プロパティ

分割線の方法を定義します。

Vertical:項目を、コンポーネントの表示エリアの垂直方向に分割します。

Horizontal:項目を、コンポーネントの表示エリアの水平方向に分割します

分割レイアウト・マネージャ

DividerStyle プロパティを使用して、レイアウト分割線のスタイルを設定します。スタイルには、バー、隆起、ラインまたは間隔があります（間隔の上に移動させるとカーソルは変化します）。

分割レイアウト・アイテム

Sizable プロパティ

Sizable プロパティは、表示エリア内での項目とそれに続く項目の間の分割線の可用性を制御します。Sizable プロパティがFalseの場合、アプリケーションのユーザーは表示アエリアの分割方法を調整することはできません。

[Sizable プロパティ](#)

[Weight プロパティ](#)

Weight プロパティ

Weight プロパティは、制御されている表示エリアのサイズを変更する際、スペースの割り当てを制御します。Weight プロパティがゼロの場合、この項目が制御しているコンポーネントのサイズは変更されません。すべての項目の値が同じ場合、スペースは均等に分けられます。

フロー・レイアウト・マネージャ

フロー・レイアウト・マネージャは、項目を行または列に分割し、項目を適切な行/列に順番に移動させ、コンポーネントを表示エリアに配置します。

Directionプロパティにより行と列の振り分けが制御され、ItemsPerDivisionプロパティにより行/列の数が制御されます。

Direction プロパティ

ItemPerDivision プロパティ

FlowOperation プロパティ

FlowOperationHorizontal プロパティ

FlowOperationVertical プロパティ

MaxColumns プロパティ

MaxRows プロパティ

MinColumns プロパティ

MinRows プロパティ

SizingRuleHorizontal プロパティ

SizingRuleVertical プロパティ

Spacing プロパティ

SpacingItems プロパティ

Direction プロパティ

フロー制御の方向を定義します。

表示エリアで項目がどのように分割され、最終的な行と列に並べられるかを定義します。

LeftToRight:項目を行に分割します。行を上から下に並べ、各行で項目を左から右に並べます。

RightToLeft:項目を行に分割します。行を上から下に並べ、各行で項目を右から左に並べます。

TopToBottom:項目を列に分割します。列を左から右に並べ、各列で項目を上から下に並べます。

BottomToTop:項目を列に分割します。列を左から右に並べ、各列で項目を下から上に並べます。

ItemPerDivision プロパティ

各行/列に配置する項目の数を定義します。

Direction プロパティが LeftToRight または RightToLeft の場合、ItemsPerDivision プロパティは表示エリアがどのように行に分割されるかを制御します。ItemsPerDivision プロパティがゼロの場合、表示エリアはすべての項目を含むのに十分な数の行を持ちます。ゼロ以外の場合、ItemsPerDivision プロパティは各行の項目の数を定義し、表示エリアはすべての項目を含むのに十分な数の行を持つようになります。

Direction プロパティが TopToBottom または BottomToTop の場合、ItemsPerDivision プロパティは表示エリアがどのように列に分割されるかを制御します。ItemsPerDivision プロパティがゼロの場合、表示エリアはすべての項目を含むのに十分な数の列を持ちます。ゼロ以外の場合、ItemsPerDivision プロパティは各列の項目の数を定義し、表示エリアにはすべての項目を含むのに十分な数の列を持つようになります。

このプロパティは、MaxColumns, MinColumns, MaxRows および MinRows の値を上書きします。

FlowOperation プロパティ

行/列を並べるフローを定義します。

Direction プロパティがLeftToRightの場合：

Center:行はSpacing プロパティの値で分割され、すべて表示エリアの垂直方向に中央揃えされます。

Increase:行はSpacing プロパティの値で分割され、表示エリアの上から下に配置されます。

Decrease:行はSpacing プロパティの値で分割され、表示エリアの下から上に配置されます。

Spread:行はSpacing プロパティの値で分割され、残りの垂直方向のスペースは行間に分散されます。

Direction プロパティがRightToLeftの場合：

Center:行はSpacing プロパティの値で分割され、すべて表示エリアの垂直方向に中央揃えされます。

Increase:行はSpacing プロパティの値で分割され、表示エリアの下から上に配置されます。

Decrease:行はSpacing プロパティの値で分割され、表示エリアの上から下に配置されます。

Spread:行はSpacing プロパティの値で分割され、残りの垂直方向のスペースは行間に分散されます。

Direction プロパティがTopToBottomの場合：

Center:列はSpacing プロパティの値で分割され、すべて表示エリアの水平方向に中央揃えされます。

Increase:列はSpacing プロパティの値で分割され、表示エリアの左から右に配置されます。

Decrease:列はSpacing プロパティの値で分割され、表示エリアの右から左に配置されます。

Spread:列はSpacing プロパティの値で分割され、残りの水平方向のスペースは列間に分散されます。

Direction プロパティがBottomToTopの場合：

Center:列はSpacing プロパティの値で分割され、すべて表示エリアの水平方向に中央揃えされます。

Increase:列はSpacing プロパティの値で分割され、表示エリアの右から左

に配置されます。

Decrease:列はSpacingプロパティの値で分割され、表示エリアの左から右に配置されます。

Spread:列はSpacingプロパティの値で分割され、残りの水平方向のスペースは列間に分散されます。

FlowOperationHorizontal プロパティ

行または列におけるコンポーネントの水平方向のフローを定義します。

行または列におけるコンポーネントの水平方向のフローを定義します。

表示エリアが行に分割される場合：

Center:項目はSpacingItems プロパティの値で分割され、すべて行の水平方向に中央揃えされます。

Increase:項目はSpacing プロパティの値で分割され、左または右から増加しつつ行の水平方向に配置されます。

Decrease:項目はSpacingItems プロパティの値で分割され、左または右から減少しつつ表示エリアを横切って配置されます。

Spread:項目はSpacing プロパティの値で分割され、残りの水平方向のスペースは項目の間に分散されます。

表示エリアが列に分割される場合：

Center:項目は、列の水平方向に中央揃えされます。

Increase:項目は、列の左または右側に配置されます。

Decrease:項目は、列の右または左側に配置されます。

Spread:中央揃えと同じです。

FlowOperationVertical プロパティ

行または列におけるコンポーネントの垂直方向のフローを定義します。

表示エリアが行に分割される場合：

Center:項目は、行の水平方向に中央揃えされます。

Increase:項目は、行の上または下に配置されます。

Decrease:項目は、行の上または下に配置されます。

Spread:中央揃えと同じです。

表示エリアが列に分割される場合：

Center:項目はSpacingItemsプロパティの値で分割され、すべて列の垂直方向に中央揃えされます。

Increase:項目はSpacingItemsプロパティの値で分割され、表示エリアの上または下から、垂直方向に増加しながら配置されます。

Decrease:項目は、SpacingItemsプロパティの値で分割され、表示エリアの上または下から垂直方向に減少しながら配置されます。

Spread:項目はSpacingプロパティの値で分割され、残りの垂直方向のスペースは項目の間に分散されます。

MaxColumns プロパティ

MaxColumns プロパティを使用してレイアウトの最大列数を設定します。

MaxColumnsはMinColumnsを上書きすること、つまり最小値が最大値より大きい場合、その値がMaxColumnsの値に設定されることに注意してください。

ItemsPerDivision プロパティは、このプロパティを上書きします。

MaxRows プロパティ

MaxRows プロパティを使用してレイアウトの最大行数を設定します。

MaxRowsはMinRowsを上書きすること、つまり最小値が最大値より大きい場合、その値はMaxRowsの値に設定されることに注意してください。

ItemsPerDivision プロパティは、このプロパティを上書きします。

MinColumns プロパティ

MinColumns プロパティを使用して、レイアウトの最小列数を設定します。

MaxColumns は MinColumns を上書きすること、つまり最小値が最大値より大きい場合、その値が MaxColumns の値に設定されることに注意してください。

ItemsPerDivision プロパティは、このプロパティを上書きします。

MinRows プロパティ

MinRows プロパティを使用して、レイアウトの最小行数を設定します。
MaxRows は MinRows を上書きすること、つまり最小値が最大値より大きい場合、その値は MaxRows の値に設定されることに注意してください。
ItemsPerDivision プロパティは、このプロパティを上書きします。

SizingRuleHorizontal プロパティ

水平方向のサイズ変更ルールを定義します。

None: コンポーネントの幅は調整されません。

Maximum: コンポーネントの幅は、項目が割り当てられた列におさまるように変更されます。

SizingRuleVertical プロパティ

垂直方向のサイズ変更ルールを定義します。

None: コンポーネントの高さは調整されません。

Maximum: コンポーネントの高さは、項目が割り当てられた行におさまるように変更されます。

Spacing プロパティ

行または列の間隔を定義します。

SpacingItems プロパティ

行または列の項目の間隔を定義します。

フロー・レイアウト・アイテム

グリッド・レイアウト・アイテム

グリッド・レイアウト・マネージャ

RuleAlignment プロパティ

Top プロパティ

Left プロパティ

Width プロパティ

Height プロパティ

RuleAlignment プロパティ

RuleSizing プロパティ

WeightHorizontal プロパティ

WeightVertical プロパティ

グリッド・レイアウト・マネージャ

グリッド・レイアウト・マネージャ

RuleAlignment プロパティ

コンポーネントにおけるグリッドの配置ルールを規定します。

Top プロパティ

レイアウト・セルの上からの行位置を定義します。

Topプロパティを使用して、レイアウトセルをどこに置くかを上からの行位置で定義します。

Left プロパティ

レイアウトセルの左からの列位置を定義します。

Left プロパティを使用して、レイアウトセルをどこに置くかを左からの列位置で定義します。

Width プロパティ

レイアウトに含まれる列の数を定義します。

Width プロパティを使用して、レイアウトに含まれる列の数を定義します。

Height プロパティ

レイアウトに含まれる行の数を定義します。

Height プロパティを使ってレイアウトに含まれる行の数を定義します。

RuleAlignment プロパティ

コンポーネントのセル内での配置ルールを定義します。

RuleAlignment プロパティを使用して、コンポーネントのセル内での配置を定義します。

RuleSizing プロパティ

セルのコンポーネントのサイズ変更ルールを定義します。

RuleSizingプロパティを使用して、コンポーネントのセル内でのサイズ変更を定義します。

WeightHorizontal プロパティ

水平方向のコンポーネントのサイズ変更に使われるウェイトを定義します。

WeightHorizontal プロパティを使用して、水平方向のコンポーネントのサイズ変更に使われるウェイトを設定します。

Weight は、制御されている表示エリアのサイズを変更する際、スペースの割り当てを制御します。Weight プロパティがゼロの場合、この項目が制御しているコンポーネントのサイズは変更されません。すべての項目の値が同じ場合、スペースは均等に分けられます。

WeightVertical プロパティ

垂直方向のコンポーネントのサイズ変更を使用されるウェイトを定義します。

WeightVertical プロパティを使用して、垂直方向のコンポーネントのサイズ変更を使用されるウェイトを設定します。

Weight は、制御されている表示エリアのサイズを変更する際、スペースの割り当てを制御します。Weight プロパティがゼロの場合、この項目が制御しているコンポーネントのサイズは変更されません。すべての項目の値が同じ場合、スペースは均等に分けられます。

ProgID プロパティ

ActiveXコンポーネントのProgIDプロパティを使用して、ActiveXコントロールのProgID、またはLANSAリポジトリに登録したいActiveXを使用可能なアプリケーションを指定します。省略記号 (...) のボタンをクリックしてリストからコントロールまたはアプリケーションのProgIDを選択し、自身のPCにActiveXコントロールやアプリケーションを表示してください。

ProgID (プログラムID) パラメータとは、SSCalendar.SSDateComboCtrl.1のような、人が読み取れるActiveXコンポーネント名のことです。

ProgIDは、ActiveXコンポーネントのクラスIDの別名として使えます。このクラスIDは、GUID (Globally Unique Identifier: グローバル一意識別子)、すなわち一意性が保証された128ビットの値で、通常は16桁の16進数の形で表現します。

TypeLibId プロパティ

TypeLibIdプロパティは、登録しようとしているActiveXコンポーネントのタイプ・ライブラリを指定します。

タイプ・ライブラリには、アプリケーションに対して、ActiveXコンポーネントと対話するためのインターフェースを提供する、という役割があります。ライブラリには、コンポーネント内のクラスとそのメソッド、プロパティ、イベント、定数などが詳しく説明されています。

ショートカットキー

ショートカットキー・コンポーネントを使用して、実行時にショートカットキーをコンポーネントまたは一連のコンポーネントに関連付けます。ショートカットキー・コンポーネントにはPressedイベントがあり、ショートカットキーが使われると起動されます。

ショートカットキーをコンテナ・コンポーネント（フォーム、パネル、再利用可能パーツなど）に割り当てると、ショートカットキーは、含まれるすべてのコンポーネントに対して有効になります。言い換えると、コンテナ上のどのコンポーネントにフォーカスがあっても、ショートカットキーはPressedイベントを起動します。

ショートカットパラメータに提供された値のリストからショートカットキーを選ぶことができます。値には、ファンクションキー (Fn) およびCtrl、Alt、Shiftキーと文字の組み合わせがあります。

以下のフォームでは、ショートカットキー・コンポーネント（ショートカットはF4）がフォームに割り当てられます。フォームには2つのフィールドがあります。実行時にF4を押すと、フォームのどこにフォーカスがあっても、ショートカットキーのPressedイベントを起動します。

```
FUNCTION OPTIONS(*DIRECT)
BEGIN_COM HEIGHT(188) LEFT(416) TOP(153) WIDTH(425)
DEFINE_COM CLASS(#PRIM_STPG) NAME(#FORM_SHORTCUTKEY)
DEFINE_COM CLASS(#ADDRESS1.Visual) NAME(#ADDRESS1) DISPL/
DEFINE_COM CLASS(#ADDRESS2.Visual) NAME(#ADDRESS2) DISPL/
EVTROUTINE HANDLING(#FORM_SHORTCUTKEY.Pressed)
use builtin(MESSAGE_BOX_SHOW) with_args(OK OK 'Shortcut' " 'Shortcut
ENDROUTINE
END_COM
```

フォーカスがフィールド内にある時、フォーム上のフィールドに別々のショートカットキー・コンポーネント（同じまたは異なるショートカット）を割り当て、異なるアクションを起動することができます。

さらに、NormalChild、OwnedChildまたはStayOnTopChildのFormStyleのフォームを表示する場合、どのショートカットキー・コンポーネントをフォームのFormOwnerに割り当てても有効になります。フォーカスがSC_CHILD_1の編集ボックスの1つにある時にShift+F1を押し、SC_MAINとSC_CHILDのフォームを使用してキーを押すと、SC_MAIN内でShift+F1を押すのと同じように機能します。一方、SC_CHILD_2の

編集ボックスの1つでShift+F1を押すとVisual LANSヘルプが起動されます。

HotKeyPressed イベント

Pressedイベントは、ユーザーがショートカットキーを押すと起動されます。このイベントを使用して、ショートカットキーで起動されるアクションを指定します。

以下のPressedイベントは、ショートカットキーが押された時にメッセージ・ボックスを表示します。

```
EVTROUTINE HANDLING(#FORM_SHORTCUTKEY.Pressed)
use builtin(MESSAGE_BOX_SHOW) with_args(OK OK 'Shortcut' " 'Shortcut
ENDROUTINE
```

Form SC_MAIN:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(79)
Clientwidth(492) Height(106) Left(244) Top(124)
Define_Com Class(#SC_CHILD) Name(#SC_CHILD_1)
Componentversion(1) Formstyle(NormalChild) Left(400)
Define_Com Class(#SC_CHILD) Name(#SC_CHILD_2)
Componentversion(1)
Define_Com Class(#PRIM_STPG) Name(#STPG_1)
Parent(#COM_OWNER) Shortcut(Shift+F1)

EvtRoutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)

#SC_CHILD_1.FormOwner <= #Com_Owner
#SC_CHILD_1.ShowForm

#SC_CHILD_2.ShowForm
Endroutine
EvtRoutine Handling(#STPG_1.Pressed) Options(*NOCLEARMESSAGES
*NOCLEARERRORS)
Use Builtin(OV_Message_Box) With_Args("Short cut has been pressed")
Endroutine
End_Com
```

Form SC_CHILD:

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Clientheight(100)
Clientwidth(333) Height(127) Left(46) Top(412) Width(341)
Define_Com Class(#PRIM_EDIT) Name(#EDIT_1) Displayposition(1)
Left(8) Parent(#COM_OWNER) Showselection(False)
Showselectionhilight(False) Tabposition(1) Top(8) Width(313)
Define_Com Class(#PRIM_EDIT) Name(#EDIT_2) Displayposition(2)
Left(8) Parent(#COM_OWNER) Showselection(False)
Showselectionhilight(False) Tabposition(2) Top(48) Width(313)
```

```
Evtroutine Handling(#com_owner.Initialize)
Set Com(#com_owner) Caption(*component_desc)
Endroutine
```

```
End_Com
```

Safe Array

Safe Arrayとは、次元の数とその範囲の情報を含む配列のことです。Safe Arrayは、最大で3つの次元を含むことができます。

配列には範囲の情報が含まれており、そのデータにアクセスする前に範囲のインデックスを確認できるため、Safe Array (安全な配列) と呼ばれます。

[Safe Arrayのプロパティ](#)

[Safe Arrayのメソッド](#)

Safe Arrayのプロパティ

Collects プロパティ

Dimension1Size プロパティ

Dimension2Size プロパティ

Dimension3Size プロパティ

Safe Array Item プロパティ

MaxDimensions プロパティ

Collects プロパティ

Collects プロパティを使用して、配列に含まれるコンポーネントのタイプを指定します。

Dimension1Size プロパティ

Dimension1Size プロパティは、配列の最初の次元にあるエレメントの数を指定します。

Dimension2Size プロパティ

Dimension2Size プロパティは、配列の2番目の次元にあるエレメントの数を指定します。

Dimension3Size プロパティ

Dimension3Size プロパティは、配列の3番目の次元にあるエレメントの数を指定します。

Safe Array Item プロパティ

Itemプロパティは、配列の要素を特定します。このプロパティは、3つのパラメータを受け入れます。1番目は必須で、あとの2つは任意です。

MaxDimensions プロパティ

MaxDimensions プロパティは、配列の次元の最大数を設定します。この値は3以下です。

Safe Arrayのメソッド

Add [メソッド](#)

Add メソッド

Addメソッドを使用して、配列の最後に順次項目を追加します。

入力フィールド

入力フィールドコンポーネントです。

コレクション
コレクションです。

Array コレクション

Arrayコレクションは、サイズや順序を動的に変えられるコンポーネントのコレクションで、インデックスにより検索することができます。インデックスには、常に1を基準とした値が付けられます。

Array コレクション・アクセサー

アクセサーは、コレクションへの読み取り専用アクセスを提供するコンポーネントです。

コレクション・アクセサー

アクセサーは、コレクションへの読み取り専用アクセスを提供するコンポーネントです。

Keyed コレクション・アクセサー

アクセサーは、コレクションへの読み取り専用アクセスを提供するコンポーネントです。

Array コレクション・アイテレータ

アイテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

コレクション・アイテレータ

アイテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

Keyed コレクション アイテレータ

アイテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

Dictionary コレクション

Dictionaryコレクションは、キーの重複がないキー値コンポーネントのペアを順不同に並べたものです。

KDictionary コレクション・アクセサー

アクセサーは、コレクションへの読み取り専用アクセスを提供するコンポーネントです。

Dictionary コレクション・アイテレータ

アイテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

List コレクション

List コレクションは、順序付けされたコンポーネントのコレクションを提供します。Listコンポーネントの機能は、指定されたインデックスまたはリストの始まりや終わりを参照して実際の位置を決めることです。インデックスには、常に1を基準とした値が付けられます。

[Listコレクション・メソッド](#)

List コレクションのメソッド

Append メソッド

Concatenate メソッド

AppendのためのOtherList パラメータ

ConcatenateのためのOtherList パラメータ

Result パラメータ

Append メソッド

Appendメソッドは、提供されたコレクションの最後にコンポーネントを挿入します。

構文は以下のとおりです。

```
Append(#CollectionVariable)
```

以下のステートメントは、コレクション #Listを #Collectionに追加します。

```
Invoke Method(#Collection.Append) Otherlist(#List)
```

Concatenate メソッド

Concatenateメソッドは、メソッドを実行しているコレクションと OtherListパラメータで提供されたコレクションの2つのコレクションの項目を組み合わせる新しいコレクションを作成します。

操作を実行するためには、コレクションは同じタイプでなければなりません。

以下のステートメントは、#Collectionと#Listを組み合わせるResultに新しいコレクションを返します。

```
Define_Com Class(#PRIM_LCOL<#PRIM_PHBN>) Name(#COLLECTION)
Define_Com Class(#PRIM_LCOL<#PRIM_PHBN>) Name(#List)
Define_Com Class(#PRIM_LCOL<#PRIM_PHBN>) Name(#Result) Referenc
Invoke Method(#Collection.Concatenate) Otherlist(#List) Result(#Result)
```

AppendのためのOtherList パラメータ

OtherListパラメータを使用して、コレクションの最後に追加される他のコレクションを指定します。

ConcatenateのためのOtherList パラメータ

OtherListパラメータを使用して、連結する他のコレクションを指定します。

Result パラメータ

Resultパラメータは、新しく作成されたコレクションを返します。

List コレクション・アクセサー

アクセサーは、コレクションへの読み取り専用アクセスを提供するコンポーネントです。

List コレクション アイテレータ

アイテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

[IsEqual イベント](#)

基本コレクション

基本コレクション項目

基本コレクションのイベント

基本コレクションのメソッド

基本コレクションのプロパティ

基本コレクション項目

Item パラメータ

Itemパラメータを使用して、検索する項目を指定します。

Object パラメータ

Objectパラメータを使用して、Subjectパラメータで指定されたコンポーネントと比較されるコンポーネントを指定します。

Subject パラメータ

Subjectパラメータを使用して、比較されるコンポーネントを指定します。

Result パラメータ

Resultパラメータは、比較の結果です。

KeyOf プロパティ

KeyOfプロパティを使用して、コレクションの項目のキーの値を返します。

以下のステートメントは、コレクション全体をループし、各項目のキーの値をフィールド#STD_COUNTに、項目のキャプションをフィールド#STD_DESCSに割り当て、グリッドに追加します。

```
For Each(#Current) In(#Collection)
  Change Field(#STD_COUNT) To('#Collection.KeyOf<#Current>')
  Change Field(#STD_DESCS) To('#Current.Caption')
  Add_Entry To_List(#GRID_1)
Endfor
```

Item パラメータ

Itemパラメータを使用して、キーを取り出す項目を指定します。

Value パラメータ

ハッシュされる値です。

Valueパラメータはバリエーションなので、どんなタイプの値でもとることができ、適切にハッシュできます。

Result パラメータ

Resultパラメータは、ハッシュ値の計算結果です。

基本コレクションのイベント

IsEqual イベント

Compare イベント

Hash イベント

Hash イベント

Hashは、コレクションを整理するために正数値を決定する必要がある場合に起動されます。

Hashイベントは、コレクション内の2つのコンポーネントが同じであるかコレクションの判断が必要な場合に起動されます。

このイベントが処理されない場合、コレクションはコンポーネント参照の一致について比較をします。

名前や識別子などの値でディクショナリを検索したい場合があります。そのような値のハッシュを取得するには、ハッシュしたい値を提供するHashイベントの実行中にCalculateHashメソッドを起動します。

```
EvtRoutine Handling(#COLLECTION.Hash) Options(*NOCLEARMESSAGE  
Invoke Method(#Collection.CalculateHash) Value(#Subject) Result(#Hash)  
Endroutine
```

Result パラメータ

Resultパラメータは、次のコンポーネントがあるかどうかを示します。
次のコンポーネントがある場合、Resultの値はTrueです。

[Subject](#) パラメータ

[Hash](#) パラメータ

Subject パラメータ

Subjectパラメータは、ハッシュされる値です。

Hash パラメータ

Hashパラメータは、ハッシュ値の計算結果です。

[Result](#) パラメータ

Result パラメータ

Resultパラメータは、挿入された項目に置換された項目への参照を返します。

IsEqual イベント

コレクションが同じ場合、このイベントが起動されます。

IsEqualイベントは、コレクション内の2つのコンポーネントが同じかどうかコレクションの判断が必要な場合に起動されます。このイベントが処理されない場合、コレクションはコンポーネント参照の一致について比較をします。

以下の例では、コレクション内の2つのコンポーネントを比較します。

```
Evtroutine Handling(#COLLECTION.IsEqual) Options(*NOCLEARMESSA(
```

コレクションがIsEqualイベントに、コレクションのタイプによって知らせる場合：

- 1 ほとんどのコレクションは、検索に関する機能が起動されるとイベントに知らせます。IsEqualメソッドは、探す項目の検索に使われます。RemoveなどのメソッドやKeyOfなどのキー付きプロパティは、アクションのオブジェクトが見つかる何度でもイベントを起動します。
- 2 重複を許可しないコレクション (DictionaryやSetなど) は、Hashイベントを使用してハッシュ兄弟を検出し、IsEqualイベントを使用して兄弟を区別します。これらのコレクションは、InsertでもIsEqualイベントを起動します。

IsEqualはコレクションにコンポーネントを追加し、各コンポーネント内の1つまたは複数の値に基いてこれらの値を検索する場合に便利です。例えば、転送リスト内に一連のEmployeeオブジェクトをまとめ、リストにEmployeeオブジェクトを1回以上追加しないようにしたい場合などです。この場合、従業員のEMPNO値をIsEqual識別子として使うことができます。

IsEqual パラメータ

[Object](#) パラメータ

[Subject](#) パラメータ

[Equal](#) パラメータ

Object パラメータ

Objectは、比較されるコンポーネントです。

Objectパラメータは、Subjectパラメータに指定されたコンポーネントと比較されるコンポーネントです。

Subject パラメータ

Subjectパラメータは、比較されるコンポーネントです。

Equal パラメータ

Equalは、比較の結果です。

Equalパラメータは、比較の結果です。TrueまたはFalseになります。

Compare イベント

Compareは、2つのコンポーネントの比較が必要な場合に起動されます。Compareイベントは、コレクションがコレクション内の2つのコンポーネントを比較する必要がある時に起動されます。このイベントが起動されない場合、コレクションはコンポーネント参照を比較します。

Compareイベントは、より大きい、より小さいまたは等しいという結果を戻す必要があります。

Compareイベント・ルーチンはSubjectとObjectの値を比較し、等しいかどうか確認します。

```
Evtroutine Handling(#COLLECTION.Compare) Options(*NOCLEARMESS/  
If Cond('#Subject.Value > #Object.Value')  
Set Com(#Result) Value(Greater)  
Else  
If Cond('#Subject.Value < #Object.Value')  
Set Com(#Result) Value(Less)  
Else  
Set Com(#Result) Value(Equal)  
Endif  
Endif  
Endroutine
```

Object パラメータ

Subject パラメータ

Result パラメータ

基本コレクションのメソッド

CreateAccessor メソッド

CreateIterator メソッド

MoveNext メソッド

Reset メソッド

CalculateHash メソッド

Remove メソッド

RemoveAll メソッド

RemoveAt メソッド

RemoveFirst メソッド

RemoveLast メソッド

ReplaceAt メソッド

ContainsItem メソッド

OccurrencesOf メソッド

Find メソッド

Contains メソッド

Insert メソッド

InsertAfter メソッド

InsertBefore メソッド

InsertFirst メソッド

InsertLastメソッド

CreateAccessor メソッド

CreateAccessor メソッドは、コレクションへの読み取り専用アクセスを提供するアクセサコンポーネントを作成します。

コレクションへの読み取り専用アクセスを作成する際は、Define_Pty with Get(*Collection)の使用を推奨します。例については、「Visual LANS Components A Developer's View」を参照してください。

[Result](#) **パラメータ**

Result パラメータ

Resultパラメータは、作成したアクセサへの参照を受け取ります。

CreateIterator メソッド

CreateIteratorメソッドは、コレクション内の反復を可能にするイテレータ・コンポーネントを作成します。

CreateIteratorメソッドは、コレクションのコンテンツ上の反復を可能にする反復コンポーネントを作成します。

コレクション内の反復には、FOR/ENDFORループの使用を推奨します。

[Result](#) パラメータ

Result パラメータ

CreateIteratorメソッドで作成されたコンポーネントにコンポーネント参照を返します。

MoveNext メソッド

MoveNextメソッドは、イテレータを次のコンポーネントに移動させ、以下のコンポーネントがあるかどうかを示す結果を返します。

Result [パラメータ](#)

Reset メソッド

Resetは、アイテレータを起動します。

Resetメソッドを使用してアイテレータを起動し、すべてのコレクション上の反復を可能にします。

CalculateHash メソッド

CalculateHashメソッドを使用して、ハッシュ値を計算します。

ハッシュ値は、常にコンポーネントを識別するハッシュを得るためにコレクションに使用されます。ハッシュを使用して、コレクション内の同じハッシュを持つコンポーネントに限り、同一かどうか確認します。省略値ハッシュはコンポーネントのポインターです。

名前や識別子などの値でディクショナリを検索したい場合があります。そのような値のハッシュを取得するには、ハッシュしたい値を提供するHashイベントの実行中にCalculateHashメソッドを起動します。

```
EvtRoutine Handling(#COLLECTION.Hash) Options(*NOCLEARMESSAGE  
Invoke Method(#Collection.CalculateHash) Value(#Subject) Result(#Hash)  
Endroutine
```

Hashイベントは、コレクションに多数の項目があり、迅速な検索が必要な順不同のコレクション (SetとDictionary) のみで使われることに注意してください。

その他のクシオンでは、項目をソートする手段としてCompareイベントを使用して検索します。

順序付けされたコレクションは、一致するものを探してコレクション内を検索します。これらのコレクションは、IsEqualイベントを使用し、ListとArrayコレクション・クラスを含みます。

[Result パラメータ](#)

[Value パラメータ](#)

Remove メソッド

Removeメソッドは、コレクションから項目を消去します。

このステートメントは、コレクションから #PHBN ボタンを消去します。

```
Invoke Method(#Collection.Remove) Object (#Phbn_1) Result(#Item)
```

このメソッドは、キーで識別された項目を消去します。

```
Invoke Method(#Collection.Remove) Key(#Key) Result(#Item)
```

Removeメソッドのパラメータは、コレクションのタイプによって異なります：

- キー付きコレクションでは、消去される項目はキーで識別されます。
- その他のコレクションでは、消去される項目はObjectパラメータで識別されます。

- [Result パラメータ](#)

- [Object パラメータ](#)

[消去の為のKeyパラメータ](#)

Result パラメータ

Resultパラメータは、消去した項目への参照を返します。

Object パラメータ

Objectパラメータを使用して、コレクションから消去されるオブジェクトを指定します。

消去の為のKey パラメータ

Keyパラメータを使用して、コレクションから消去される項目のキーを指定します。

RemoveAll メソッド

RemoveAllメソッドを使用して、コレクションのすべての項目を消去します。

以下のステートメントは、コレクションのすべての項目を消去します。

```
Invoke Method(#Collection.RemoveAll)
```

RemoveAt メソッド

RemoveAtメソッドを使用して、指定されたインデックス番号の項目を消去します。

以下のステートメントは、フィールドで指定されたインデックス値の項目をコレクションから消去します。

```
Invoke Method(#Collection.RemoveAt) Index(#STD_NUM)
```

Result パラメータ

Index パラメータ

Result パラメータ

Resultパラメータは、消去した項目への参照を返します。

Index パラメータ

Indexパラメータを使用して、コレクションから消去する項目のインデックス番号を指定します。

RemoveFirst メソッド

RemoveFirstメソッドを使用して、コレクションの最初の項目を消去します。

以下のステートメントは、コレクションの最初の項目を消去します：

```
Invoke Method(#Collection.RemoveFirst)
```

[Result](#) パラメータ

Result パラメータ

Resultパラメータは、消去した項目への参照を返します。

RemoveLast メソッド

RemoveLastメソッドを使用して、コレクションの最後の項目を消去します。

以下のステートメントは、コレクションの最後の項目を消去します：

```
Invoke Method(#Collection.RemoveLast)
```

[Result](#) パラメータ

Result パラメータ

Resultパラメータは、消去した項目への参照を返します。

ReplaceAt メソッド

ReplaceAtメソッドは、指定のインデックス番号の項目を置換します。
ReplaceAtメソッドを使用して、指定したインデックス番号の項目を別の項目に置換します。

以下のステートメントは、フィールド#STD_NUMで指定したインデックス番号の項目を #PHBN_1 ボタンで置換します：

```
Invoke Method(#Collection.ReplaceAt) Item(#Phbn_1) Index(#STD_NUM)
```

Result パラメータ

Item パラメータ

Index パラメータ

Result パラメータ

Resultは、指定した位置で置換した項目への参照を返します。
コレクションの最後の項目の後に挿入する時の値は*NULLです。

Item パラメータ

Itemパラメータを使用して、コレクションに追加する項目を指定します。

Index パラメータ

Indexパラメータを使用して、置換される項目のインデックス番号を指定します。

ContainsItem メソッド

ContainsItemメソッドを使用して、項目がコレクションに存在するか確認します。

このステートメントは、コンポーネント #PHBN_1 がコレクションに存在するか確認します。

```
Invoke Method(#Collection.ContainsItem) Item(#Phbn_1) Result(#STD_BOO
```

[ContainsItem の結果](#)

ContainsItem の結果

ContainsItemメソッドの結果は、TrueまたはFalseになります。

OccurrencesOf メソッド

OccurrencesOfメソッドは、オブジェクトの発生数を返します。

OccurrencesOfメソッドを使用して、オブジェクトの発生数を返します。

以下のメソッドは、#PHBN_1 ボタンの発生数を返します。

Invoke Method(#Collection.OccurrencesOf) Object(#Phbn_1)

Result(#STD_COUNT)

Result パラメータ

Object パラメータ

Result パラメータ

Resultパラメータは、指定したオブジェクトの発生数を返します。

Object パラメータ

Objectパラメータを使用して、発生数をカウントするオブジェクトを指定します。

Find メソッド

Findメソッドは、コレクションの項目への参照を返します。

Findメソッドを使用して、コレクションの項目を検索し、参照を設定します。

以下のステートメントはキーに基づいて項目を検索し、Resultパラメータに参照を設定します。

```
Invoke Method(#Collection.Find) Key(#Key) Result(#Item)
```

Findメソッドには2つの異なる形があります：

- キー付きコレクションのFindメソッドは、Keyパラメータを使用してコレクション内に一致するキーコンポーネントがあるか確認します。
- それ以外のFindメソッドは、Objectパラメータを使用します。
- [Key パラメータ](#)
- [Object パラメータ](#)
- [Result パラメータ](#)

Key パラメータ

Keyパラメータを使用して、検索する項目を特定します。

Object パラメータ

Objectパラメータを使用して、検索する項目を指定します。

Result パラメータ

Resultパラメータは、見つかった項目への参照を返します。

Contains メソッド

Containsは、キーに基づいて項目を検索します。

Containsメソッドを使用して、オブジェクトがコレクションに存在するか確認します。存在する場合、ContainsメソッドのResultsはTrueに設定されます。

Containsメソッドには2つの異なる形があります：

DictionaryおよびSorted Dictionary コレクションのContainsメソッドは、Keyパラメータを使用して、コレクションに一致するキーパラメータがあるか確認します。

Invoke Method(#Collection.Contains) key(#index) Result(#STD_bool)

それ以外のコレクションのContainsメソッドは、Objectパラメータを使用します。提供される参照は検索のオブジェクトとして使われ、一致条件はIsEqualとCompareイベントに使われるロジックに依存するためです。

Invoke Method(#Collection.Contains) Object(#Phbn_1) Result(#STD_BOOL)

Result [パラメータ](#)

Result パラメータ

Containsメソッドの結果は、TrueまたはFalseになります。

Object パラメータ

Objectパラメータを使用して、確認する項目を指定します。オブジェクトはコレクションで収集されるコンポーネント・クラスのインスタンスでなければなりません。

[Key パラメータ](#)

Key パラメータ

Keyパラメータを使用して、確認するオブジェクトのキーを指定します。

Insert メソッド

Insertメソッドは、コレクションに項目を追加します。

Insertメソッドを使用して、コレクションに項目を追加します。

以下のステートメントは、コレクションに #PHBN_1 ボタンを追加します。

```
Invoke Method(#Collection.Insert) Item(#Phbn_1)
```

以下のステートメントは、Keyパラメータが特定した位置に#PHBN_2コンポーネントを挿入します。

```
Invoke Method(#Collection.Insert) Key(#Key) Item(#PHBN_2)
```

Insertメソッドのパラメータは、コレクションのタイプによって異なります：

- キー付きのコレクションでは、挿入される項目の位置はKey値で特定されます。
- その他のコレクションには、特定の位置に項目を挿入する別のメソッドがあります (InsertAfter、InsertFirstなど)。
- [挿入の為のKey パラメータ](#)
- [Item パラメータ](#)

挿入の為のKey パラメータ

Keyパラメータを使用して、項目を挿入するキー位置を指定します。
以下のステートメントは、コンポーネント#PHBN_1をフィールド
#LastIndexで指定されたキー位置に挿入します。

```
Invoke Method(#Collection.Insert) Key(#LastIndex) Item(#PHBN_1)
```

Item パラメータ

Itemパラメータを使用して、コレクションに挿入する項目を指定します。

InsertAfter メソッド

InsertAfterメソッドは、コレクションの指定された項目の後に項目を追加します。

InsertAfterメソッドを使用して、コレクション内の指定された項目の後に項目を追加します。

以下のステートメントは、コレクションの指定されたインデックス番号の項目の後に #PHBN_1 ボタンを追加します。

```
Invoke Method(#Collection.InsertAfter) Item(#Phbn_1) Index(#Index.Value)
```

- [Item](#) パラメータ
- [Index](#) パラメータ

Item パラメータ

Item パラメータ

Itemパラメータを使用して、コレクションに挿入する項目を指定します。

Index パラメータ

Indexパラメータを使用して、その後にこの項目を追加する、コレクションのインデックス番号を指定します。

InsertBefore メソッド

InsertBeforeは、コレクションの指定された項目の前に項目を追加します。

InsertBeforeメソッドを使用して、コレクションの指定された項目の前に項目を追加します。

以下のステートメントは、コレクションの指定されたインデックス番号の項目の前に #PHBN_1 ボタンを追加します。

```
Invoke Method(#Collection.InsertBefore) Item(#Phbn_1) Index(#Index.Value)
```

- [Item](#) パラメータ
- [Index](#) パラメータ

Item パラメータ

Itemパラメータを使用して、コレクションに挿入する項目を指定します。

Index パラメータ

Indexパラメータを使用して、その前にこの項目を追加する、コレクション内のインデックス番号を指定します。

InsertFirst メソッド

InsertFirstは、コレクションの最初に項目を追加します。

InsertFirstメソッドを使用して、コレクションの最初に項目を追加します。

このステートメントは、コレクション内で最初の項目として #PHBN_1 ボタンを追加します。

Invoke Method(#Collection.InsertFirst) Item(#Phbn_1)

Item パラメータ

Item パラメータ

Item パラメータ

Itemパラメータを使用して、コレクションに挿入する項目を指定します。

InsertLastメソッド

InsertLastは、コレクションの最後に項目を追加します。

InsertLastメソッドを使用して、コレクションの最後に項目を追加します。

以下のステートメントは、コレクションの最後の項目として #PHBN_1 ボタンを追加します。

```
Invoke Method(#Collection.InsertLast) Item(#Phbn_1)
```

Item [パラメータ](#)

Item パラメータ

Itemパラメータを使用して、コレクションに挿入する項目を指定します。

基本コレクションのプロパティ

StartingAt パラメータ

AtEnd プロパティ

Current プロパティ

CurrentKey プロパティ

LastIndexOf プロパティ

AllowsDuplicates プロパティ

IsEmpty プロパティ

IsOrdered プロパティ

Item プロパティ

StartingAt パラメータ

StartingAtパラメータを使用して、コレクション内でオブジェクトの検索を開始する位置を指定します。

このパラメータは、重複を許可するコレクション(配列やリストなど)を検索する際、1つの項目を見つけた後、検索再開を可能にするのに便利です。

AtEnd プロパティ

AtEnd プロパティ

AtEndは、コレクションの最後を示します。

AtEndプロパティは、イテレータに現在の項目がない場合、Trueに設定されます。イテレータが動かされるたびに、イテレータに使用可能な項目があるか判断するため、AtEndが確認されます。使用可能な項目がない場合、AtEndはTrueとなり、コレクションの最後に到達したことになります。

コレクション内の反復に最もよい方法はFOR/ENDFORループを使うことです。

Current プロパティ

Current プロパティはアイテレータが現在のコンポーネントにアクセスすると参照を返します。

CurrentKeyプロパティ

CurrentKeyプロパティは、アイテレータが現在のコンポーネントにアクセスするとキーを返します。

LastIndexOf プロパティ

LastIndexOfプロパティは、リストを検索し、コンポーネントの最後の発生を探します。

Object パラメータ

Objectパラメータを使用して、最後の発生を検索するコンポーネントを指定します。

AllowsDuplicates プロパティ

AllowsDuplicatesは、コレクションが重複を許可するかどうかを示します。

読み取り専用プロパティです。

AllowsDuplicatesプロパティを使用して、コレクションが重複を許可するかどうか決定します。

ListとArrayコレクションは重複を確認しないため、重複をサポートします。この2つのコレクションのAllowsDuplicatesプロパティはTrueです。

First プロパティ

Firstは、コレクションの最初の項目です。

Firstプロパティを使用して、コレクション内の最初のアイテムにアクセスします。

以下のステートメントは、コレクション内の最初の項目のキャプションをフィールド#STD_DESCSに割り当てます。

```
Change Field(#STD_DESCS) To('#Collection.First.Caption')
```

Last プロパティ

Lastは、コレクションの最後の項目です。

コレクション内の最後のアイテムにアクセスするには、Lastプロパティを使用します。

このステートメントは、コレクション内の最後のアイテムのキャプションをフィールド#STD_DESCSに割り当てます。

```
Change Field(#STD_DESCS) To('#Collection.Last.Caption')
```

IndexOf プロパティ

IndexOfは、コレクション内の項目のインデックス番号を返します。
IndexOfプロパティを使用して、コレクションの項目のインデックス番号を返します。

このステートメントは、コレクション全体をループし、各項目のインデックス番号をフィールド#STD_COUNTに、項目のキャプションをフィールド#STD_DESCSに割り当て、グリッドに追加します。

```
For Each(#Current) In(#Collection)
  Change Field(#STD_COUNT) To('#Collection.IndexOf<#Current>')
  Change Field(#STD_DESCS) To('#Current.Caption')
  Add_Entry To_List(#GRID_1)
Endfor
```

Object パラメータ

Objectパラメータを使用して、そのインデックスを取り出すオブジェクトを指定します。

StartingAt パラメータ

StartingAtパラメータを使用して、コレクション内でオブジェクトの検索を開始する位置を指定します。

このパラメータは、重複を許可するコレクション(配列やリストなど)を検索する際、1つの項目を見つけた後、検索再開を可能にするのに便利です。

IsEmpty プロパティ

IsEmptyは、コレクションに項目があるかどうかを示します。

IsEmptyプロパティは、コレクションに項目があるかどうかを示します。

このプロパティがTrueの場合、コレクションに項目はありません。

IsOrdered プロパティ

IsOrderedはコレクションが順序付けされているかどうかを示します。

配列やList コレクションのように、項目にその順序でアクセスできるように項目を保持するコレクションがあります。

その他のコレクションは、項目の記憶域を最適化し、特定のパフォーマンスの目的を達成します。

- キー付きコレクションは、バランスのとれたバイナリ・ツリーを使って項目を保持します。これは項目のキーの値に基づいて並び替えられた順に項目が取り出されることを意味します。
- Dictionary コレクションは、ハッシュ・テーブルを使います。項目の順序はまったく予期できないため、ディクショナリを検索するにはキーを使う必要があるということです。

Item プロパティ

Itemプロパティを使用して、コレクションの項目にアクセスします。

[Index](#) パラメータ

[ItemCount](#) プロパティ

[識別の為のKey](#)パラメータ

Index パラメータ

Indexパラメータを使用して、コレクションの項目を特定します。

ItemCount プロパティ

ItemCount プロパティを使用して、コレクションの項目番号を取り出します。

以下のifステートメントは、ItemCount プロパティを使用して、指定されたインデックス値がコレクションの項目数以下かどうか確認します。

```
If Cond('#Index.Value <= #Collection.ItemCount')
Change Field(#STD_COUNT) To('#Index.Value')
Change Field(#STD_DESCS) To('#Collection<#Index>.Caption')
Add_Entry To_List(#GRID_1)
Endif
```

識別の為のKeyパラメータ

Keyパラメータを使用して、コレクションの項目を特定します。

Object パラメータ

Objectパラメータを使用して、コレクション内の項目を特定します。

Set コレクション

Set コレクションは、重複を含むことができない順不同のコンポーネントのコレクションです。

以下も参照してください。

[Set コレクション・アクセサー](#)

[Set コレクション・イテレータ](#)

Set コレクション・アクセサー

アクセサーは、コレクションへの読み取り専用アクセスを提供するコンポーネントです。

Set コレクション・アイテレータ

アイテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

Sorted array コレクション

Sorted array コレクションは、サイズやソート順を動的に変えられるコンポーネントのコレクションで、インデックスにより検索することができます。インデックスには、常に1を基準とした値が付けられます。

以下も参照してください。

[Sorted array コレクション・アクセサー](#)

[Sorted array コレクション・イテレータ](#)

[Sorted dictionary コレクション](#)

[Sorted dictionary コレクション・アクセサー](#)

[Sorted dictionary コレクション・イテレータ](#)

Sorted array コレクション・アクセサー

アクセサーは、コレクションへの読み取り専用アクセスを提供するコンポーネントです。

Sorted array コレクション・イテレータ

イテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

Sorted dictionary コレクション

Sorted dictionary コレクションは、キー値コンポーネントのペアのコレクションです。このコレクションは、キー・コンポーネントの順で並んでおり、重複キーは許されません。

Sorted dictionary コレクション・アクセサー

アクセサーは、コレクションへの読み取り専用アクセスを提供するコンポーネントです。

Sorted dictionary コレクション・アイテレータ

アイテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

Component type コンポーネント

Component type コンポーネントは、Visual LANSAのコンポーネント・タイプを反映します。

反映は、コンポーネント・クラスが自分自身を反映している場合、RDMLXコードでコンポーネント・クラス内を見ることができるようになる機能です。デザイン/RDMLX生成機能を提供するツールは、コンポーネントを初期化/操作するためにコンポーネント・タイプの機能を問い合わせる必要がありますが、このコンポーネントはそのようなツールで使うことができます。

Component type コンポーネントには、コンポーネントの属性や機能へのアクセスを提供するプロパティがあります。Component type の機能は、Event、Method、Propertyコンポーネントのコレクションとして提供されます。各コンポーネントは、各機能の属性とパラメータを記述します。

Component Typeコンポーネントのプロパティ

Ancestor プロパティ IsVariant プロパティ
Events プロパティ Methods プロパティ
IsAbstract プロパティ Pattern プロパティ
IsClass プロパティ Properties プロパティ
IsInterface プロパティ TypeName プロパティ
IsPrimitive プロパティ

Ancestor プロパティ

Ancestor プロパティは、祖先のコンポーネント・タイプです。

Ancestor プロパティは、このタイプのコンポーネントの祖先のタイプである Component Type コンポーネントを返します。

コンポーネント・タイプに祖先がない場合、*NULLを返します。

IsAbstract プロパティ

IsAbstractは、このコンポーネント・タイプからインスタンスが作成できるかを示します。

IsAbstract プロパティは、このタイプのコンポーネントがインスタンスの作成に使えるかどうかを示します。

タイプがインスタンス作成に使えない場合、IsAbstract プロパティはTrueに設定されます。

IsClass プロパティ

IsClass プロパティは、コンポーネント・タイプがクラスかどうかを示します。

コンポーネント・タイプがクラスの場合、このプロパティはTrueに設定されます。

IsInterface プロパティ

IsInterface プロパティは、コンポーネント・タイプがインターフェースかどうかを示します。

このプロパティは現在導入されていません。

IsPrimitiveプロパティ

IsPrimitiveプロパティは、コンポーネント・タイプがLANSAの基本コンポーネント・タイプかどうかを示します。

このプロパティ値は、TrueまたはFalseになります。

IsVariant プロパティ

IsVariant プロパティは、コンポーネントタイプがバリエーションかどうかを示します。

このプロパティ値は、True または False になります。

Pattern プロパティ

Patternプロパティは、このコンポーネント・タイプのパターンであるコンポーネントを返します。

Properties プロパティ

Properties プロパティは、コンポーネント・タイプがPropertyコンポーネントのコレクションです。

Methodsプロパティ

Methodsプロパティは、コンポーネント・タイプがMethodコンポーネントのコレクションです。

Events プロパティ

Events プロパティは、コンポーネント・タイプが Event コンポーネントのコレクションです。

TypeName プロパティ

TypeName プロパティは、コンポーネント・タイプの名前を返します。

Event コンポーネント

Eventコンポーネントは、Visual LANSAイベントを反映します。

[Event コンポーネントのプロパティ](#)

Event コンポーネントのプロパティ

FeatureName プロパティ

IsFinal プロパティ

OuterFeature プロパティ

Parameters プロパティ

FeatureNameプロパティ

FeatureNameプロパティは、イベントの名前です。

IsFinalプロパティ

IsFinalプロパティは、イベントに子孫のクラスから信号が送られるかどうかを示します。

このプロパティは、子孫のクラスから信号が送られない場合、Trueに設定されます。

OuterFeature プロパティ

OuterFeature プロパティは、イベントを Component Type コンポーネントです。

Parametersプロパティ

Parametersプロパティは、イベントのパラメータを記述するProperty Event Parameterコンポーネントのコレクションです。

Event Parameterコンポーネント

Event Parameterコンポーネントは、Visual LANSAイベント・パラメータを反映します。

[Event Parameterコンポーネントのプロパティ](#)

Event Parameterコンポーネントのプロパティ

FeatureName プロパティ

IsByReference プロパティ

IsByValue プロパティ

IsInput プロパティ

IsMandatory プロパティ

IsOutput プロパティ

OuterFeature プロパティ

FeatureName プロパティ

FeatureName プロパティは、パラメータの名前です。

IsByReference プロパティ

IsByReference プロパティは、参照からパラメータが渡されるかどうかを示します。

参照から渡される場合、このプロパティはTrueに設定されます。

IsByValue プロパティ

IsByValue は値からパラメータが渡されるかどうかを示します。

IsByValue プロパティは、値からパラメータが渡されるかどうかを示します。

値からパラメータが渡される場合、このプロパティはTrueに設定されます。参照から渡される場合、このプロパティはFalseに設定されます。

IsInput プロパティ

IsInputは、値から入力のパラメータが渡されるかどうかを示します。

IsInputプロパティは、イベント処理ルーチンに入力のパラメータが渡されるかどうかを示します。

このプロパティは、入力のパラメータがイベント処理ルーチンに渡される場合、Trueに設定されます。

IsMandatory プロパティ

IsMandatory プロパティは、パラメータが必要かどうかを示します。
このプロパティは、パラメータが必要な場合、Trueに設定されます。

IsOutput プロパティ

IsOutputは、イベント処理ルーチンから出力のパラメータが渡されるかどうかを示します。

IsOutputプロパティは、イベント処理ルーチンから出力のパラメータが渡されるかどうかを示します。

このプロパティは、イベント処理ルーチンから出力のパラメータが渡される場合、Trueに設定されます。

OuterFeature プロパティ

OuterFeatureは、イベント・パラメータのEventコンポーネントです。

OuterFeatureプロパティは、イベント・パラメータのEventコンポーネントです。

Method コンポーネント

Methodコンポーネントは、Visual LANSAメソッドを反映します。

[Method コンポーネントのプロパティ](#)

Method コンポーネントのプロパティ

FeatureName プロパティ

IsFinal プロパティ

IsRedefined プロパティ

OuterFeature プロパティ

Parameters プロパティ

FeatureName プロパティ

FeatureName プロパティは、メソッドの名前です。

IsFinal プロパティ

IsFinalは、メソッドが子孫のクラスで再定義されるかどうかを示します。

IsFinalプロパティは、メソッドが子孫のクラスで再定義されるかどうかを示します。

このプロパティは、メソッドが子孫のクラスで再定義されない場合、Trueに設定されます。

IsRedefined プロパティ

IsRedefined プロパティは、メソッドが祖先のメソッドを再定義するかどうかを示します。

このプロパティは、メソッドが祖先からのメソッドを再定義する場合、True に設定されます。

OuterFeature プロパティ

OuterFeatureは、メソッドを含むComponent Typeコンポーネントです。

OuterFeatureプロパティは、メソッドを含むComponent Typeコンポーネントです。

Parameters プロパティ

Parameters プロパティは、Method Parmeter コンポーネントのコレクションです。

Parameters プロパティは、メソッドのパラメータを記述する Method Parameter コンポーネントのコレクションです。

Method parameter コンポーネント

Method parameterコンポーネントは、Visual LANSAMETHOD・パラメータを反映します。

[Method parameter コンポーネントのプロパティ](#)

Method parameter コンポーネントのプロパティ

FeatureName プロパティ

IsByReference プロパティ

IsInput プロパティ

IsMandatory プロパティ

OuterFeature プロパティ

IsResult パラメータ

IsByValue プロパティ

IsOutput プロパティ

FeatureName プロパティ

FeatureNameは、パラメータの名前です。

FeatureName プロパティは、パラメータの名前です。

IsByReference プロパティ

IsByReference プロパティは、参照からパラメータが渡されるかどうかを示します。

参照から渡される場合、このプロパティはTrueに設定されます。

IsInput プロパティ

IsInputは、パラメータが入力として渡されるかどうかを示します。

IsInputプロパティは、パラメータが入力としてメソッド・ルーチンに渡されるかどうかを示します。

このプロパティは、パラメータが入力としてメソッド・ルーチンに渡される場合、Trueに設定されます。

IsMandatory プロパティ

IsMandatory プロパティは、パラメータが必要かどうかを示します。
このプロパティは、パラメータが必要な場合、Trueに設定されます。

OuterFeature プロパティ

OuterFeature プロパティは、Method パラメータにMethodコンポーネントを返します。

IsResult パラメータ

IsResultパラメータは、パラメータが Result パラメータかどうかを示します。

このプロパティ値は、TrueまたはFalseになります。

IsByValue プロパティ

IsByValue プロパティは、値からパラメータが渡されるかどうかを示します。

値からパラメータが渡される場合、このプロパティはTrueに設定されます。参照から渡される場合、このプロパティはFalseに設定されます。

IsOutput プロパティ

IsOutputプロパティは、メソッド・ルーチンからパラメータが出力として渡されるかどうかを示します。

このプロパティは、メソッド・ルーチンからパラメータが出力として渡される場合、Trueに設定されます。

Property コンポーネント

Property コンポーネントは、Visual LANSAPロパティを反映します。

[Property コンポーネントのプロパティ](#)

Property コンポーネントのプロパティ

FeatureName プロパティ

GetParameters プロパティ

IsGetKeyed プロパティ

IsSetKeyed プロパティ

OuterFeature プロパティ

SetParameters プロパティ

FeatureName プロパティ

FeatureName プロパティは、プロパティの名前です。

GetParameters プロパティ

GetParametersは、プロパティのGETパラメータです。

GetParameters プロパティは、プロパティのGETパラメータを記述する Property Parameterコンポーネントのコレクションです。

IsGetKeyedプロパティ

IsGetKeyedは、GETにパラメータがあるかどうかを示します。

IsGetKeyedプロパティは、プロパティにGETパラメータがあるかどうかを示します。

このプロパティは、プロパティにGET上のパラメータ(キー)がある場合、Trueに設定されます。

IsSetKeyed プロパティ

IsSetKeyedは、SETにパラメータがあるかどうかを示します。

IsSetKeyed プロパティは、プロパティにSETパラメータがあるかどうかを示します。

このプロパティは、プロパティにSET上のパラメータ (キー) がある場合、Trueに設定されます。

OuterFeature プロパティ

OuterFeature プロパティは、プロパティを含むComponent Typeコンポーネントです。

SetParameters プロパティ

SetParametersは、プロパティのSETパラメータです。

SETParametersプロパティは、プロパティのSETパラメータを記述するProperty Parameterコンポーネントのコレクションです。

Property parameter コンポーネント

Property parameter コンポーネントは、Visual LANSAPロパティを反映します。

[Property parameter コンポーネントのプロパティ](#)

Property parameter コンポーネントのプロパティ

FeatureName プロパティ

IsByReference プロパティ

IsByValue プロパティ

IsInput プロパティ

IsMandatory プロパティ

IsOutput プロパティ

IsResult プロパティ

OuterFeature プロパティ

FeatureName プロパティ

FeatureName プロパティは、パラメータの名前です。

IsByReference プロパティ

IsByReference プロパティは、参照からパラメータが渡されるかどうかを示します。

参照から渡される場合、このプロパティはTrueに設定されます。

IsByValue プロパティ

IsByValue プロパティは、値からパラメータが渡されるかどうかを示します。

値からパラメータが渡される場合、このプロパティはTrueに設定されます。参照から渡される場合、このプロパティはFalseに設定されます。

IsInput プロパティ

IsInputプロパティは、プロパティ・ルーチンにパラメータが入力として渡されるかどうかを示します。

このプロパティは、パラメータが入力としてプロパティ・ルーチンに渡される場合、Trueに設定されます。

IsMandatory プロパティ

IsMandatoryは、パラメータが必要かどうかを示します。

IsMandatory プロパティは、パラメータが必要かどうかを示します。

このプロパティは、パラメータが必要な場合、Trueに設定されます。

IsOutput プロパティ

IsOutputプロパティは、プロパティ・ルーチンからパラメータが出力として渡されるかどうかを示します。

このプロパティは、プロパティ・ルーチンからパラメータが出力として渡される場合、Trueに設定されます。

IsResult プロパティ

IsResult プロパティは、パラメータが Result パラメータかどうかを示します。

OuterFeature プロパティ

OuterFeature プロパティは、プロパティ・パラメータのPropertyコンポーネントです。

Meta pattern コンポーネント

Meta patternコンポーネント・タイプは、Visual LANSAスキーマのすべてのパターンに使えるパターンです。Visual LANSAのフォーム、再利用可能パーツ、フィールド、ビットマップ、アイコン、カーソル、ActiveXおよびWebパターンのタイプを記述するクラスがあります。

Messageコンポーネント

Messageコンポーネントは、ステータスバーのメッセージのコレクションです。

Messageコンポーネントは、ステータスバーのメッセージのコレクションです。ステータスバーのMessagesプロパティを使用して、Messageコンポーネントにアクセスできます。

以下のClickイベント・コードは、メッセージ・コレクション内を繰り返し通り、一次レベルのメッセージ・テキストをリストに追加します。

```
EvtRoutine Handling(#STbr_1.click) Options(*NOCLEARERRORS *NOCLE  
For Each(#Message) In(#stbr_1.messages)  
Change Field(#STD_TEXT) To(#MESSAGE.FIRSTLEVELTEXT)  
Add_Entry To_List(#TRACE)  
Endfor  
Endroutine
```

FirstLevelText プロパティ

HasSecondLevelText プロパティ

SecondLevelText プロパティ

FirstLevelText プロパティ

FirstLevelText プロパティは、メッセージ・テキストです。

HasSecondLevelText プロパティ

HasSecondLevelTextは、メッセージに2次レベルのテキストがあるかどうかを示します。このプロパティは、TrueまたはFalseに設定できます。

SecondLevelText プロパティ

SecondLevelText プロパティは、メッセージ記述です。

Primitive コレクション

LANSAPrimitive コレクション (PRIM_PCOL) は、すべてのコレクション・コンポーネントの基礎となります。

FOR/ENDFORループを使用して、どのコレクションの項目にもアクセスできます。

例えば、このClickイベントは、サブメニュー#SMNU_1を探して項目を反復し、サブメニューを無効にします。

```
Evtroutine Handling(#PHBN_1.Click)
For Each(#Current) In(#smnu_1.Items)
Set Com(#Current) Enabled(False)
Endfor
Endroutine
```

Primitive アイテレータ・コンポーネント

アイテレータ・コンポーネントは、コレクションのコンテンツ全体の反復を可能にします。

Variant コンポーネント

SetValue プロパティ

SetValue プロパティ

バリエーションの値を設定します。結果は、割り当てが成功するとTrueに、失敗するとFalseになります。

RESULT

設定が成功 (True) したかどうかを示すブール値です。

VALUE

設定する値です。

例

```
Define_Com Class(#PRIM_VAR) Name(#myVariant)  
#myVariant.SetValue( "123.45" )
```

DefaultPicklist プロパティ

DefaultPicklistは、ピックアップリストを省略値のピックアップリストにします。
フィールドのコンポーネント定義に複数のピックアップリストがある場合、
DefaultPicklist プロパティを使用して、ピックアップリストを省略値として使用するかどうかを指定します。
ピックアップリストを省略値として使用するには、このプロパティをTrueに設定します。

DefaultPrompter プロパティ

DefaultPrompterは、プロンプターを省略値のプロンプターにします。フィールドのコンポーネント定義に複数のプロンプターがある場合、DefaultPrompterプロパティを使用して、プロンプターを省略値として使用するかどうかを指定します。プロンプターを省略値として使用するには、このプロパティをTrueに設定します。

DefaultVisual プロパティ

DefaultVisualは、ビジュアルライゼーションを省略値のビジュアルライゼーションにします。

フィールドのコンポーネント定義に複数のビジュアルライゼーションがある場合、DefaultVisualプロパティを使用して、ビジュアルライゼーションを省略値として使用するかどうかを指定します。

ビジュアルライゼーションを省略値として使用するには、このプロパティをTrueに設定します。

フィールドがリストやグリッドに使われている場合、省略値のビジュアルライゼーションは無視されることに注意してください。リストやグリッドでのフィールドの表示は、列のDisplayAppearanceと EditAppearanceにより、決定されます。

Desktop application コンポーネント・クラス

Desktop applicationコンポーネントは、Visual LANSA Desktop Applicationコンポーネントの機能を定義するコンポーネント・クラスです。

実行時、変数名#SYS_APPLNを使用して、実行中のアプリケーションを示すオブジェクトにアクセスします。

Desktopアプリケーションプロパティです。

Desktop applicationのメソッド

Desktop applicationのプロパティ

Desktop applicationのメソッド

CaptureDesktop メソッド

CaptureWindowsメソッド

CaptureWindowResult パラメータ

CaptureWindowFileName パラメータ

CaptureWindowControl パラメータ

CaptureDesktop メソッド

CaptureDesktopを使用して、デスクトップのビットマップを取得し、ファイルに保存します。

以下の例をコピー・貼り付けして、CaptureDesktopメソッドがどのように動作するかを確認することができます。

Function Options(*DIRECT)

CaptureWindow メソッド

アクティブ・ウィンドウのイメージをキャプチャします。

CaptureWindowメソッドを使用して、アクティブなウィンドウまたはコントロールのビットマップを取得し、ファイルに保存します。

以下の例をコピー・貼り付けして、CaptureWindowメソッドがどのように動作するかを確認することができます。

```
Function Options(*DIRECT)
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('Capture Screen') Click
* Menu definition
Define_Com Class(#PRIM_MBAR) Name(#MBAR_1) Parent(#COM_OWNER)
Define_Com Class(#PRIM_MITM) Name(#MITM_1) Caption('File') Display
Define_Com Class(#PRIM_SMNU) Name(#SMNU_1) Parent(#MITM_1)
Define_Com Class(#PRIM_MITM) Name(#MITM_SCREEN) Caption('Capture Screen')
Define_Com Class(#PRIM_MITM) Name(#MITM_ACTIVE) Caption('Wait 5 Seconds')
Define_Com Class(#PRIM_MITM) Name(#MITM_TAB) Caption('Capture Tab')
* File name edit
Define_Com Class(#STD_TEXT.Visual) Name(#STD_TEXT) Caption('File name')
* Timer for active window capture delay
Define_Com Class(#prim_timr) Name(#timer) Interval(5000)
* Tab, Sheets and Contents for SYS_APPL.CaptureScreen Control testing
Define_Com Class(#PRIM_TAB) Name(#TAB_1) Displayposition(2) Height(40)
Define_Com Class(#PRIM_IMGE) Name(#IMAGE) Displayposition(1) Height(40)
Define_Com Class(#PRIM_TBSH) Name(#TBSH_RESULT) Caption('Results')
* Form Layout
Define_Com Class(#PRIM_ATLM) Name(#ATLM_1)
Define_Com Class(#PRIM_ATLI) Name(#ATLI_1) Attachment(Center) Mana
Define_Com Class(#PRIM_ATLI) Name(#ATLI_2) Attachment(Bottom) Mana
EvtRoutine Handling(#MITM_SCREEN.Click)
* Capture the screen
#sys_appln.capturedesktop( #std_text )
* Update image
#IMAGE.filename := ""
#IMAGE.filename := #std_text
Endroutine
EvtRoutine Handling(#MITM_ACTIVE.Click)
* Start timer (will tick 5 seconds later and capture the active window )
#timer.start
```

```
Endroutine
Evtroutine Handling(#MITM_TAB.Click)
* Capture the sheet contents
#sys_appln.capturewindow( #std_text #TAB_1 )
* Update image
#IMAGE.filename := ""
#IMAGE.filename := #std_text
Endroutine
Evtroutine Handling(#COM_OWNER.CreateInstance) Options(*NOCLEARM
* Initialize form
#IMAGE.filename := ""
#std_text := "c:\Bitmap.bmp"
#timer.stop
Endroutine
Evtroutine Handling(#timer.Tick) Options(*NOCLEARMESSAGES *NOCLE
* Stop the timer
#timer.stop
* Capture the foreground window
#sys_appln.capturewindow( #std_text )
* Update image
#IMAGE.filename := ""
#IMAGE.filename := #std_text
Endroutine
End_Com
```

CaptureWindowResult パラメータ

CaptureWindowメソッドは、画面のキャプチャが成功しビットマップが作成された場合、Trueのブール値を返し、失敗した場合はFalseのブール値を返します。

CaptureWindowFileName パラメータ

キャプチャされた画面が保存されるファイルです。

CaptureWindowControl パラメータ

CaptureWindowのコントロールです。このパラメータに値がない場合、アクティブなデスクトップ・ウィンドウをキャプチャします。

Desktop applicationのプロパティ

VirtualScreenLeft

VirtualScreenTop

VirtualScreenWidth

VirtualScreenHeight

EnsureVisibleForms

PrimaryMonitor

Monitors

ScalingFactorHor プロパティ

ScalingFactorVer プロパティ

ScreenHeight プロパティ

ScreenWidth プロパティ

ComponentForms プロパティ

Disable Process Ghosting プロパティ

Themes

VirtualScreenLeft

仮想画面は、すべての表示モニターの境界矩形です。VirtualScreenLeftプロパティは仮想画面の左側の座標を提供します。

VirtualScreenTop

仮想画面は、すべての表示モニターの境界矩形です。VirtualScreenTopプロパティは、仮想画面の上側の座標を提供します。

VirtualScreenWidth

仮想画面は、すべての表示モニターの境界矩形です。VirtualScreenWidthプロパティは、仮想画面の幅の座標を提供します。

VirtualScreenHeight

仮想画面は、すべての表示モニターの境界矩形です。VirtualScreenHeightプロパティは、仮想画面の高さの座標を提供します。

EnsureVisibleForms

セカンダリ・モニターの電源を抜く場合やモニター・レジストリ設定の破損に対応するために、SYS_APPLN.EnsureVisibleフォームや PRIM_FORM.EnsureVisibleプロパティを使うことができます。このプロパティをTrueに設定する場合、フォームのレジストリ値をロードする時にモニターが使用できない場合は、フォームが仮想画面に戻ります。このプロパティは、必ずフォームが認識される前にのみ適用されることに注意してください。

アプリケーション全体に対してEnsureVisible機能を簡単に有効にするには、SYS_APPLN.EnsureVisibleFormsプロパティをTrueに設定します。

PrimaryMonitor

オペレーティング・システムに接続されたプライマリ・モニターへのアクセスを可能にします。

Monitors

オペレーティング・システムに接続されたすべてのモニターを含むコレクションへのアクセスを可能にします。

ScalingFactorHor プロパティ

ScalingFactorHorは、アプリケーションを実行しているデスクトップの、現在の水平スケール・ファクタを返します。

ScalingFactorVer プロパティ

ScalingFactorVerは、現在アプリケーションを実行しているデスクトップの、現在の垂直スケール・ファクタを返します。

ScreenHeight プロパティ

ScreenHeight プロパティは、プライマリ・モニターの高さの座標をピクセル値で提供します。

ScreenWidth プロパティ

ScreenWidth プロパティは、プライマリ・モニターの幅の座標をピクセル値で提供します。

ComponentFormsプロパティ

ComponentFormsプロパティは、実行中のアプリケーション内で実現されたすべてのフォームを含むコレクションへのアクセスを可能にします。

CloseAllFormsメソッド

CloseAllFormsメソッドを使用してすべてのフォームを閉じ、realized されたが表示されていないフォームをunrealized にします。このメソッドを呼び出すと、アプリケーションは終了します。

CaptureDesktopResult パラメータ

CaptureDesktopメソッドは、画面のキャプチャが成功しビットマップが作成された場合Trueのブール値を返し、失敗した場合はFalseのブール値を返します。

CaptureDesktopFileName パラメータ

CaptureDesktop.FileNameパラメータです。

キャプチャされた画面が保存されるファイルです。

Disable Process Ghosting プロパティ

ウィンドウ・ゴーストは、ユーザーが応答していないアプリケーションのメイン・ウィンドウを最小化、移動または閉じられるようにするためのウィンドウ・マネージャ機能です。

DisableProcessGhostingプロパティを使用して、Windows 2000以降のシステムのゴースト機能を無効にします。

WheelOnMouseプロパティ

WheelOnMouseプロパティを使用して、マウス下のコントロール上のマウス・ホイールが、コントロールのフォーカスがあるかどうかに関係なく作動するかどうかを指定します。

省略値では、プロパティはTrueです。

Themes

この機能の詳細と有効化の情報については、『Visual LANSA 開発者ガイド』の「[テーマ](#)」を参照してください。

#Sys_Appln.Theme

- クラシック (省略値)
- シェル
- 2007ブルー
- 2007シルバー
- 2007オブシディアン
- 2007オリーブ
- 2003ブルー
- 2003シルバー
- 2003オリーブ
- 2003カラー

シェルテーマはオペレーティングシステムを確認し、最適なものを選択します。(お使いのWindows XP PCのブルー/オリーブ/シルバーの設定に従います。)

2003Color Themeを使用して、Office2003のテーマカラーをカスタマイズすることができます。(詳細は「[Theme 2003 Colors](#)」を参照してください。) 各テーマの微妙な違いが最も速くよくわかる例を見るには、Visual LANSAエディターの *Settings* ダイアログを使用してください。

以下も参照してください。

[Themes](#)

[Theme 2003 Colors](#)

[Theme Operating System](#)

[Theme Style プロパティ](#)

[Theme Draw Style プロパティ](#)

Themed Forms

この機能の詳細と有効化の情報については、『Visual LANSA 開発者ガイド』の「[テーマ](#)」を参照してください。

#Sys_Appln.ThemedForms

- True
- False

省略値では、すべてのアプリケーションのフォームのテーマは有効になっていません。

テーマを有効にするには、各フォームのThemeStyleをThemedに設定する必要があります。

または、#Sys_Appln.ThemedFormsを使用して、すべてのフォームの省略値の動作を切り替えることができます。#Sys_Appln.ThemedFormsをTrueに設定すると、アプリケーションのすべてのフォームは、#Sys_Appln.Themeプロパティで指定したテーマを使用します。

[↑ Themes](#)

Theme 2003 Colors

#Sys_Appln.Theme2003Color1

#Sys_Appln.Theme2003Color2

#Sys_Appln.Theme2003Colors

- パンプキン
- ローズ
- ナス紺
- レイニーデイ
- スプルース
- ライラック

#Sys_Appln.ThemeType := 2003Color とすると、Office2003のテーマカラーをカスタマイズすることができます。ベースカラーとハイライトカラーを指定する必要があります。

- #Sys_Appln.Theme2003Color1 が全ての背景のベースカラーとして使用されます。
- #Sys_Appln.Theme2003Color2 が全てのハイライトのベースカラーとして使用されます。

または、#Sys_Appln.Theme2003Colors を使って

#Sys_Appln.Theme2003Color1 と #Sys_Appln.Theme2003Color2を初期化することができます。使い方は「Visual LANSA エディター設定」を参照してください。

[↑Themes](#)

Theme Operating System

#Sys_Appln.OperatingSystem

プログラム・ルーチンにおいて現在のプラットフォームを見つけるには、OperatingSystemプロパティを確認します。

- Unknown
- Windows95
- Windows98
- WindowsME
- WindowsNt4
- Windows2000
- WindowsXp
- WindowsVista

[↑ Themes](#)

Theme Style プロパティ

- None
- Parent
- Themed
- Application

コントロール（すべてのコントロールではありません）のThemeStyleプロパティは、コントロールがそのテーマの外観を変更できるようにします。省略値では、すべてのコントロールは親のテーマ・スタイルに従い、親のフォームがテーマを使用する場合のみテーマを使用することができます。

コントロール（パネルなど）をThemeStyle := Noneに変更すると、テーマを有効にしたフォームをテーマのないパネルにペイントします。

コントロールをThemeStyle := Themedに設定しても、フォームのテーマが有効な場合にのみ有効であることに注意してください。

[↑ Themes](#)

Theme Draw Style プロパティ

- None
- Standard
- Image
- Toolbar
- DarkTitle
- MediumTitle
- LightTitle

Visual LANSA テーマから最もよい結果を得るために、わずかに異なる外観にする必要がある場合があります。最も明確な例では、グループボックスとパネルがアプリケーション内でツールバーとして使われてきました。テーマによっては、濃すぎるコントロールのバックグラウンドを表示し (2007 Obsidian テーマ)、アイコンの見栄えを悪くするものがあります。そのような場合は、ThemeDrawStyleでコントロールの表示を変更します。

- ツールバーとして使用されるコントロールについては、ThemeDrawStyle = Toolbarにすると、外観がかなりよくなります。
- また、画面の多くの部分を占めるコンテナ・コントロール (例えばパネルなど) のテーマは、ThemeStyle := Imageと指定することで画質を上げることができます。ThemeStyle := Imageは、2007のテーマと一緒に使用した場合しか影響がないことに注意してください。
- DarkTitle、MediumTitleおよびLightTitleはフォームの中で目立ってアプリケーションのテーマにふさわしく見える必要のあるラベルに使用します。これらは、ビジュアルスタイルのカラーがVisual LANSAアプリケーションのテーマの振る舞いをまねしようとして使用されている場所にも使用することができます。

[↑ Themes](#)

Component proxy コンポーネント

Component proxyは、非表示のActiveXコンポーネントのAncestorクラスです。

LANSA製品センター内部使用専用です。

[Component Proxyのプロパティ](#)

[Component Proxyのイベント](#)

Component Proxyのプロパティ

ProgID プロパティ

TypeLibId プロパティ

ComponentOnFailure プロパティ

ComponentOnSuccess プロパティ

ProgID プロパティ

ActiveXコンポーネントのProgIDプロパティを使用して、ActiveXコントロールのProgID、またはLANSAリポジトリに登録したり、ActiveXを使用可能なアプリケーションを指定します。省略記号(...)のボタンをクリックしてリストからコントロールまたはアプリケーションのProgIDを選択し、自身のPCにActiveXコントロールやアプリケーションを表示してください。

ProgID (プログラムID) パラメータとは、SSCalendar.SSDateComboCtrl.1のような、人が読み取れるActiveXコンポーネント名のことです。ProgIDは、ActiveXコンポーネントのクラスIDの別名として使えます。このクラスIDは、GUID (Globally Unique Identifier: グローバル一意識別子)、すなわち一意性が保証された128ビットの値で、通常は16桁の16進数の形で表現します。

TypeLibId プロパティ

TypeLibId プロパティは、登録しようとしているActiveXコンポーネントのタイプ・ライブラリを指定します。

タイプ・ライブラリには、アプリケーションに対して、ActiveXコンポーネントと対話するためのインターフェースを提供する、という役割があります。ライブラリには、コンポーネント内のクラスとそのメソッド、プロパティ、イベント、定数などが詳しく説明されています。

Tag:LpComComponentProxyComponentOnFailureHelp

Parent:'Com Components'

ComponentOnFailure プロパティ

ComponentOnFailure プロパティは、ActiveXコンポーネントのインスタンスがFailure Com Errorのフラグを出す場合、アプリケーションの動作を制御します。

許容される値は以下のとおりです。

Continue:アプリケーションの通常の実行を再開します。

SignalError:失敗したActiveXコンポーネントのComponentErrorイベントを知らせます。コントロールがイベントの処理から戻ると、アプリケーションの通常の実行が再開されます。

Abort:アプリケーションを中止する前にComErrorをレポートします。

省略値はAbortです。

ActiveXのリポジトリ定義に割り当てられた値はActiveXの各インスタンスの初期化に使われます。このプロパティの値は、プログラム実行時に修正できます。

ComponentOnSuccess プロパティ

ComponentOnSuccess プロパティは、ActiveX コンポーネントのインスタンスが Success (または情報提供の) Com Error のフラグを出す場合、アプリケーションの動作を制御します。

許容される値は以下のとおりです。

Continue: アプリケーションの通常の実行を再開します。

SignalError: 失敗した ActiveX コンポーネントの ComponentError イベントを知らせます。コントロールがイベントの処理から戻ると、アプリケーションの通常の実行が再開されます。

Abort: アプリケーションを中止する前に ComError をレポートします。

省略値は Continue です。

ActiveX のリポジトリ定義に割り当てられた値は ActiveX の各インスタンスの初期化に使われます。このプロパティの値は、プログラム実行時に修正できます。

Component Proxyのイベント

ComponentError イベント

ComponentError イベント

ComponentErrorイベントは、コンポーネントとの対話処理でCom Errorが起き、SignalEventアクション・コードで処理されるようフラグが出された場合、ActiveXコンポーネントによって起動されます。

ComponentErrorイベントは、Failure Com Errorフラグが出され、ActiveXコンポーネントがComponentOnFailureをSignalErrorに設定した場合に起動されます。

ComponentErrorイベントは、Success Com Errorフラグが出され、ActiveXコンポーネントがComponentOnSuccessをSignalErrorに設定した場合に起動されます。

Control container コンポーネント

Control containerコンポーネントは、非複合ビジュアルActiveXコンポーネントの祖先です。

LANSA製品センター内部使用専用です。

Composite container クラス

Composite containerコンポーネントは、複合ビジュアルActiveXコンポーネントの祖先です。

LANSA製品センター内部使用専用です。

ProgID プロパティ

ActiveXコンポーネントのProgIDプロパティを使用して、ActiveXコントロールのProgID、またはLANSAリポジトリに登録したり、ActiveXを使用可能なアプリケーションを指定します。省略記号(...)のボタンをクリックしてリストからコントロールまたはアプリケーションのProgIDを選択し、自身のPCにActiveXコントロールやアプリケーションを表示してください。

ProgID (プログラムID) パラメータとは、SSCalendar.SSDateComboCtrl.1のような、人が読み取れるActiveXコンポーネント名のことです。ProgIDは、ActiveXコンポーネントのクラスIDの別名として使えます。このクラスIDは、GUID (Globally Unique Identifier: グローバル一意識別子)、すなわち一意性が保証された128ビットの値で、通常は16桁の16進数の形で表現します。

TypeLibId プロパティ

TypeLibIdプロパティは、登録されているActiveXコンポーネントのタイプ・ライブラリを指定します。

タイプ・ライブラリには、アプリケーションに対して、ActiveXコンポーネントと対話するためのインターフェースを提供する、という役割があります。ライブラリには、コンポーネント内のクラスとそのメソッド、プロパティ、イベント、定数などが詳しく説明されています。

Draglist

Draglistのプロパティ

Draglistのプロパティ

Listview DraglistStyle

Listview Dragimage

Listview DraglistStyle

リポジトリ内でDragListStyleをImageに設定し、そのDragImageをイメージに設定すると、イメージはドラッグ時にカーソルに添付されます。

DragListStyleをSelectionに設定すると、選択したコントロール内のアイテムのスナップショットが、ドラッグ時にカーソルに添付されます。

Listview Dragimage

リポジトリ内でDragListStyleをImageに設定し、そのDragImageをイメージに設定すると、イメージはドラッグ時にカーソルに添付されます。

DragListStyleをSelectionに設定すると、選択したコントロール内のアイテムのスナップショットが、ドラッグ時にカーソルに添付されます。

日時ピッカー

日時ピッカーを使用して、ユーザーが日時を選択できるようにします。
ピッカーから戻される値はDateTimeフィールドです。

以下の例をコピー・貼り付けして、日時ピッカーがどのように動作するかを確認することができます。

```
Function Options(*DIRECT)
```

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('DateTime Tester') Cl  
Define_Com Class(#PRIM_DTIM) Name(#DATETIME) Displayposition(1) F  
Define_Com Class(#PRIM_DTIM) Name(#MIN) Displayposition(3) Height(2  
Define_Com Class(#PRIM_DTIM) Name(#MAX) Displayposition(2) Height(  
Define_Com Class(#PRIM_LABL) Name(#LABL_1) Caption('Min Date') Dis  
Define_Com Class(#PRIM_LABL) Name(#LABL_2) Caption('Man Date') Di  
Define_Com Class(#PRIM_GPBX) Name(#GPBX_1) Caption('Format') Displ  
Define_Com Class(#PRIM_RDBN) Name(#RDBN_1) Buttonchecked(True) C  
Define_Com Class(#PRIM_RDBN) Name(#RDBN_2) Caption('UserTime') D  
Define_Com Class(#PRIM_RDBN) Name(#RDBN_3) Caption('SystemDate')  
Define_Com Class(#PRIM_RDBN) Name(#RDBN_4) Caption('UserDate') Di  
Define_Com Class(#PRIM_RDBN) Name(#RDBN_5) Caption('SystemTime')
```

```
Evtroutine Handling(#COM_OWNER.CreateInstance) Options(*NOCLEARM  
Set Com(#DATETIME) Value(*DateTime)
```

```
Set Com(#MIN) Value(#DATETIME.mindate)  
Set Com(#MAX) Value(#DATETIME.maxdate)
```

```
Set Com(#DATETIME) Dateformat(SystemLongDate)  
Endroutine
```

```
Evtroutine Handling(#RDBN_1.Click)  
* Set SystemLongDate  
Set Com(#DATETIME) Dateformat(SystemLongDate)  
Endroutine  
Evtroutine Handling(#RDBN_2.Click)  
* Set SystemShortDate  
Set Com(#DATETIME) Dateformat(SystemDate)  
Endroutine
```

```
Evtroutine Handling(#RDBN_3.Click)
* Set SystemTime
Set Com(#DATETIME) Timeformat(SystemTime)
Endroutine
Evtroutine Handling(#RDBN_4.Click)
* Set UserShortDate
Set Com(#DATETIME) Dateformat(UserDate)
Endroutine
Evtroutine Handling(#RDBN_5.Click)
* Set UserTime
Set Com(#DATETIME) Timeformat(UserTime)
Endroutine
```

```
Evtroutine Handling(#MIN.Changed) Options(*NOCLEARMESSAGES *NO
Set Com(#DATETIME) Mindate(#MIN.Value)
Endroutine
Evtroutine Handling(#MAX.Changed) Options(*NOCLEARMESSAGES *NO
Set Com(#DATETIME) Maxdate(#MAX.Value)
Endroutine
End_Com
```

DisplayAsUTC プロパティ
Checkbox プロパティ
Checked プロパティ
CustomDateFormat プロパティ
CustomTimeFormat プロパティ
DateFormat プロパティ
TimeFormat プロパティ
FormatText プロパティ
ShowDate プロパティ
ShowDateButton プロパティ
ShowTime プロパティ
ShowTimeButton プロパティ
Format プロパティ
MaxDate プロパティ

MinDate プロパティ
Value プロパティ

DisplayAsUTC プロパティ

DisplayAsUTCは、日時の省略値をUTC形式で表示するかどうかを制御します。

DisplayAsUTCを使用して、日時の省略値がUTC形式で表示されることを示します。

注：このプロパティは、DataClassが添付されない日時コントロールにのみ適用されます。

Checkbox プロパティ

Checkboxは、日時ピッカーを有効にするかどうかを制御します。

Checkboxプロパティを使用して、日時ピッカー内にチェック・ボックスを表示します。チェック・ボックスを選択すると、日時コントロール内の値を変更できます。

Checked プロパティ

Checkedは、チェック・ボックスが選択されているかどうかを示します。

Checkedプロパティは、日時ピッカー内のチェック・ボックスが選択されているかどうかを示します。

このプロパティは、ピッカーのCheckboxプロパティがTrueに設定される時にのみ適用されます。

CustomDateFormat プロパティ

CustomDateFormat プロパティを使用して、カスタム日付形式を指定します。

2004/02/28を例にとると、カスタムフォーマットに指定できる値は以下のとおりです。

年の形式：

Y = 4

YY = 04

YYY = 2004

月の形式：

M = 2

MM = 02

MMM = Feb

MMMM = February

日付の形式：

D = 28

DD = 28 (0が前につく場合もあります)

DDD = Sat

DDDD = Saturday

CustomTimeFormat プロパティ

CustomTimeFormat プロパティを使用して、カスタム時間形式を指定します。

23:55:03.004.003.002を例にとると、カスタム形式に指定できる値は以下のとおりです。

時間の形式：

h = 11

hh = 11 (0が前につく場合もあります)

H = 23

HH = 23 (0が前につく場合もあります)

分の形式：

m = 55

mm = 55 (0が前につく場合もあります)

秒の形式：

s = 3

ss = 03 (0が前につく場合もあります)

ミリ秒の形式：

f = 4

ff = 04

fff = 004

マイクロ秒の形式：

u = 3

uu = 03

uuu = 003

ナノ秒の形式：

n = 2

nn = 02

nnn = 002

t = "A" or "P"

tt = "AM" or "PM"

DateFormat プロパティ

DateFormat プロパティを使用して既定の形式を選択するか、CustomDateFormat プロパティを使用して指定されるカスタム形式を使うように指定します。

既定の形式は、以下のとおりです。

User Dateは、ユーザー・ロケールで定義されるDate形式の文字列です（コントロール・パネルの地域設定で見ることができます）。

UserLongDateはユーザー・ロケールで定義されるLong Date形式の文字列です（コントロール・パネルの地域設定で見ることができます）。

SystemDateは、UserDateに相当するシステム・ロケールです。

SystemLongDateは、UserLongDateに相当するシステム・ロケールです。

TimeFormat プロパティ

TimeFormat プロパティを使用して既定の形式を選択するか、CustomTimeFormat プロパティを使用して指定されるカスタム形式を使うように指定します。

既定の形式は、以下のとおりです。

UserTimeはユーザー・ロケールで定義されるTime形式の文字列です（コントロール・パネルの地域設定で見ることができます）。

SystemTimeは、UserTimeに相当するシステム・ロケールです。

FormatText プロパティ

FormatText プロパティは、現在日時がどのようにフォーマットされているかを見られるようにする読み取り専用のプロパティです。省略値では、日時はWindowsで定義された形式で表示されます。実際の形式を確認するには、FormatTextを使います。

ShowDate プロパティ

ShowDate プロパティを使用して、日時フィールドの日付部分を表示または非表示にします。このプロパティの値は、True または False になります。

ShowDateButton プロパティ

ShowDateButtonプロパティを使用して、日時フィールドの日時選択ボタンを表示または非表示にします。

ShowTime プロパティ

ShowTimeプロパティを使用して、日時フィールドの時間部分を表示または非表示にします。このプロパティの値は、TrueまたはFalseになります。

ShowTimeButton プロパティ

ShowTimeButtonプロパティを使用して、日時フィールドの時間選択ボタンを表示または非表示にします。

Format プロパティ

Format プロパティは、日時の表示を制御します。

Format プロパティを使用して、日時をどのように表示するかを制御します。

- LongDateは次のように表示されます。 Monday, January 01, 1900
- LongDateTimeは次のように表示されます。 Monday, January 01, 1900 12:00:00 AM.
- ShortDateは次のように表示されます。 1/ 1/1900
- ShortDateTimeは次のように表示されます。 1/ 1/1900 12:00:00 AM
- Timeは次のように表示されます。 12:00:00 AM.

MaxDate プロパティ

MaxDateは指定できる最大の日時を設定します。

MaxDateプロパティを使用して、日付コントロール内で指定できる最大の日時を設定します。

MinDate プロパティ

MinDateは指定できる最小の日時を設定します。

MinDateプロパティを使用して、日付コントロール内で指定できる最小の日時を設定します。

Value プロパティ

値は、日時を設定したり取り出したりします。

Value プロパティを使用して、日時ピッカー内の日時を設定したり取り出したりします。

値は、DateTime形式で返されます。

カレンダー

カレンダー・コントロールを使用して、ユーザーの日付選択を手助けします。

カレンダーから返される値は、Dateフィールドです。

以下の例をコピー・貼り付けして、カレンダーがどのように動作するかを確認することができます。

Function Options(*DIRECT)

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Caption('calendar Tester') Clie
Define_Com Class(#PRIM_MTCL) Name(#calendar) Displayposition(6) Heig
Define_Com Class(#PRIM_DTIM) Name(#MIN) Displayposition(3) Height(2
Define_Com Class(#PRIM_DTIM) Name(#MAX) Displayposition(1) Height(
Define_Com Class(#PRIM_CKBX) Name(#CKBX_TODAY) Buttonstate(Che
Define_Com Class(#PRIM_CKBX) Name(#CKBX_TODAYCIRCLE) Button
```

```
Define_Com Class(#PRIM_LABL) Name(#LABL_1) Caption('Minimum date
Define_Com Class(#PRIM_LABL) Name(#LABL_2) Caption('Maximum date
```

```
Define_Com Class(#PRIM_LABL) Name(#LABL_3) Caption('Start of week')
Define_Com Class(#PRIM_CMBX) Name(#CMBX_1) Autoselectitem(False)
Define_Com Class(#PRIM_CBCL) Name(#CBCL_1) Displayposition(1) Pare
```

```
Evtroutine Handling(#COM_OWNER.CreateInstance) Options(*NOCLEARM
Set Com(#MIN) Value(#calendar.mindate)
Set Com(#MAX) Value(#calendar.maxdate)
Set Com(#CMBX_1) Value(#calendar.Startofweek)
```

```
Change Field(#std_text) To("MONDAY")
Add_Entry To_List(#CMBX_1)
Change Field(#std_text) To("TUESDAY")
Add_Entry To_List(#CMBX_1)
Change Field(#std_text) To("WEDNESDAY")
Add_Entry To_List(#CMBX_1)
Change Field(#std_text) To("THURSDAY")
Add_Entry To_List(#CMBX_1)
Change Field(#std_text) To("FRIDAY")
```

```
Add_Entry To_List(#CMBX_1)
Change Field(#std_text) To("SATURDAY")
Add_Entry To_List(#CMBX_1)
Change Field(#std_text) To("SUNDAY")
Add_Entry To_List(#CMBX_1)
```

```
Change Field(#std_text) To(#calendar.StartOfWeek)
Endroutine
```

```
Evtoutine Handling(#CKBX_TODAY.Click)
If Cond('#CKBX_TODAY.buttonstate = Checked')
Set Com(#calendar) Showtoday(True)
Else
Set Com(#calendar) Showtoday(False)
Endif
```

```
Endroutine
Evtoutine Handling(#CKBX_TODAYCIRCLE.Click)
If Cond('#CKBX_TODAYcircle.buttonstate = Checked')
Set Com(#calendar) Showtodaycircle(True)
Else
Set Com(#calendar) Showtodaycircle(False)
Endif
Endroutine
```

```
Evtoutine Handling(#MIN.Changed) Options(*NOCLEARMESSAGES *NO
Set Com(#calendar) Mindate(#MIN.Value)
Endroutine
```

```
Evtoutine Handling(#MAX.Changed) Options(*NOCLEARMESSAGES *NO
Set Com(#calendar) Maxdate(#MAX.Value)
Endroutine
```

```
Evtoutine Handling(#CMBX_1.ComboChanged) Options(*NOCLEARMESS
Set Com(#calendar) Startofweek(#std_text)
Endroutine
End_Com
```

MaxDate プロパティ

MinDate プロパティ

ShowToday プロパティ

ShowTodayCirle プロパティ

StartOfWeek プロパティ
Value プロパティ

MaxDate プロパティ

MaxDateプロパティを使用して、カレンダー内で指定できる最大の日付を設定します。

MinDate プロパティ

MinDateプロパティを使用して、カレンダー内で指定できる最大の日付を設定します。

ShowToday プロパティ

ShowTodayプロパティを使用して、今日の日付を表示するかどうかを制御します。

このプロパティは、TrueまたはFalseに設定できます。

ShowTodayCirle プロパティ

ShowTodayCirleは、今日の日付を丸で囲むかどうかを制御します。

ShowTodayCirleプロパティを使用して、カレンダー内で今日の日付を丸で囲むかどうかを制御します。

このプロパティは、TrueまたはFalseに設定できます。

StartOfWeek プロパティ

StartOfWeekプロパティを使用して、カレンダー内で1週間の最初の日とする曜日を指定します。

どの曜日でも1週間の最初の曜日に指定できます。

Value プロパティ

値は、日付を設定したり取り出したりします。

Value プロパティを使用して、カレンダー内の日付を設定したり取り出したりします。

属性ベース・クラス

属性のベース・クラスです。

LANSA製品センター内部使用専用です。

インターフェース・プロキシ・ベース・クラス
インターフェース・プロキシのベース・クラスです。
LANSA製品センター内部使用専用です。

コンテキスト・メッセージセット・コンポーネント
コンテキスト・メッセージセット・コンポーネントです。
LANSA製品センター内部使用専用です。

カレンダー・コントロール・ベース・クラス
カレンダー・コントロール・ベース・クラスです。
LANSA製品センター内部使用専用です。

マルチライン編集ボックス・ベース・クラス
マルチライン編集ボックス・ベース・クラスです。
LANSA製品センター内部使用専用です。

プログレスバー・ベース・クラス
プログレスバー・ベース・クラスです。
LANSA製品センター内部使用専用です。

スピン編集ボックス・ベース・クラス
スピン編集ボックス・ベース・クラスです。
LANSA製品センター内部使用専用です。

データ・クラス **DateTime**

LANSA製品センター内部使用専用です。

トラックバー・ベース・クラス
LANSA製品センター内部使用専用です。

WAM ベースのクラス

LANSA製品センター内部使用専用です。

WAM ベースのイベント

SessionInvalid イベント

SessionInvalid イベント

SessionInvalidは、ユーザーのセッションが無効の場合に送信されます。セッションが無効な時にアクティビティを試みると、SessionInvalidイベントの信号が送られます。

パラメータ：

SenderName - SessiooonInavlidイベントの信号の送信を引き起こしたウェブルーチンを示します。

SessionStatus - 無効なセッションの状態を示します。

Expired - セッションが時間切れになったことを意味します。

Invalid - セッションがエラーにより終了したことを意味します。

以下も参照してください。

[SenderName](#) パラメータ

SenderName パラメータ

SenderNameパラメータは、送信しているウェブルーチン名を示します。

SenderNameパラメータは、送信させるSessionInvalidイベントを引き起こしたウェブルーチンを示します。

WAM ベース・プロパティ

SessionKeyMethod プロパティ

SessionTimeout プロパティ

SessionStatus プロパティ

CheckNumeric プロパティ

LayoutWeblet プロパティ

SessionGroupName プロパティ

SessionKeyMethod プロパティ

SessionKeyMethodプロパティは、セッションが状態を維持できるようにするKeyメソッドを決定します。

SessionKeyMethodは以下のようになります。

- HiddenField (省略値) - 非表示フィールドは、セッションの状態を維持するため、HTMLソースセッション内で使用されます。
- Cookie - クッキーはセッションの状態を維持するために使用されます。URL - URL上のパラメータはセッションの状態を維持するために使用されます。

このプロパティは、開発時のみ設定できます。

SessionTimeout プロパティ

SessionTimeoutプロパティは、タイムアウトになる前にセッションが非アクティブになる秒数を決定します。

値0は、システム省略値を使うことを示します。

このプロパティは、開発時のみ設定できます。

SessionStatus プロパティ

SessionStatus プロパティは、実行時にウェブルーチン内で使われ、エラーが発生しセッションが終了されることを示します。

例：

```
Set COM(#COM_OWNER) SessionStatus(Invalid)
```

CheckNumeric プロパティ

LayoutWeblet プロパティ

SessionGroupName プロパティ

SessionGroupName プロパティは、このWAMが属するセッション・グループを決定します。

このプロパティは、開発時のみ設定できます。

Weblet ベース・クラス

LANSA製品センター内部使用専用です。

データ・クラス・ベース・クラス
内部システム専用です。

DateTime ライブラリ

DateTimeライブラリです。

日時関数のライブラリです。

LANSA製品センター内部使用専用です。このライブラリをインポートする代わりに、組み込み日時関数を使います。

DateTime ベース・クラス

LANSA製品センター内部使用専用です。

Intrinsic 関数ライブラリ

組み込みメソッドは、フィールドの値の操作に使用します。

F2キーを押して組み込み関数ライブラリの機能を表示し、クラスドロップダウン・リストを使ってライブラリクラスから選択します。

インターフェース・ベース・クラス

LANSAが発行したインターフェースのクラスです。

LANSA製品センター内部使用専用です。

メッセージ コンテキスト・ベース・クラス
LANSA製品センター内部使用専用です。

数字関数ライブラリ

LANSA製品センター内部使用専用です。このライブラリをインポートする代わりに、組み込み数字関数を使います。

文字列関数ライブラリ

LANSA製品センター内部使用専用です。

このライブラリをインポートする代わりに、組み込み文字列関数を使います。

バリエント関数ライブラリ

バリエント関数ライブラリ#PRIM_LIBVです。

関数ライブラリは、MthRoutineコマンドで定義した一連のルーチンを含むコンポーネントとして定義されます。関数ライブラリのルーチンは、式の中で使用できます。

関数ライブラリをRDML/RDMLXオブジェクトに導入するには、FUNCTIONステートメントの直後にIMPORTコマンドを指定します。

```
FUNCTION OPTIONS(*DIRECT)
  Import Libraries(#Prim_Libv) As(#VL)
```

関数ライブラリをインポートすると、そのライブラリに定義されるルーチンを式の中で使用できるようになります。

構文は以下のとおりです。

```
[LibraryName.]#RoutineName( [Parameters] )
```

この構文では、RoutineNameの直後にパラメータを開始する左かっこが続く必要があります。埋め込みスペースは使用できません。パラメータは、位置で指定することも名前をつけることもできます。

バリエーション関数ライブラリのメソッド

VarAsBoolean メソッド

VarAsDate メソッド

VarAsDecimal メソッド

VarAsInteger メソッド

VarAsReference メソッド

VarAsString メソッド

VarType メソッド

VarIsBoolean メソッド

VarIsDate メソッド

VarIsEmpty メソッド

VarIsNull メソッド

VarIsNullReference メソッド

VarIsNumber メソッド

VarIsReference メソッド

VarIsString メソッド

VarTypeメソッドです。

VarAsBoolean メソッド

VarAsBooleanは、Subjectバリエントをブール値として返します。
構文は以下のとおりです。

```
VarAsBoolean( Subject )
```

以下の例では、#PHBN_1がクリックされると#PHBN_2が無効になります。

```
Function Options(*DIRECT)
Import Libraries(#Prim_Libv) As(#VL)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM *implements
#Prim_App.IHelpHandler) CLIENTHEIGHT(141) CLIENTWIDTH(152)
COMPONENTTAG('Help for the form') HEIGHT(168) LEFT(681) TOP(135)
WIDTH(160)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1)
DISPLAYPOSITION(1) LEFT(24) PARENT(#COM_OWNER)
TABPOSITION(1) TOP(64)
DEFINE_COM CLASS(*Variant) NAME(#MyVariant)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_2)
DISPLAYPOSITION(2) LEFT(24) PARENT(#COM_OWNER)
TABPOSITION(2) TOP(32)
EVTROUTINE HANDLING(#PHBN_1.Click)
#myvariant := false
set #phbn_2 enabled(#VL.VarAsboolean(#myvariant))
ENDROUTINE
EVTROUTINE HANDLING(#PHBN_2.Click)
set #phbn_2 caption('Clicked')
ENDROUTINE
End_Com
```

Result パラメータ

Resultパラメータは、対象のバリエントをブール値として返します。
メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarAsDate メソッド

VarAsDateメソッドは、対象のバリエントを日付として返します。
構文は以下のとおりです。

```
VarAsDate(subject)
```

Result パラメータ

Resultパラメータは、対象のバリエーションを日付として返します。
メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarAsDecimal メソッド

VarAsDecimalは、対象のバリエーションを小数として返します。
構文は以下のとおりです。

```
VarAsDecimal( Subject )
```

This example resizes #PHBN_1 using a variant:

```
Function Options(*DIRECT)
```

```
Import Libraries(#Prim_Libv) As(#VL) BEGIN_COM ROLE(*EXTENDS  
#PRIM_FORM *implements #Prim_App.IHelpHandler)
```

```
CLIENTHEIGHT(141) CLIENTWIDTH(152) COMPONENTTAG('Help for  
the form') HEIGHT(168) LEFT(681) TOP(135) WIDTH(160)
```

```
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1)
```

```
DISPLAYPOSITION(1) LEFT(24) PARENT(#COM_OWNER)
```

```
TABPOSITION(1) TOP(64)
```

```
DEFINE_COM CLASS(*Variant) NAME(#MyVariant)
```

```
EVTROUTINE HANDLING(#PHBN_1.Click)
```

```
#myvariant := 50
```

```
set #phbn_1 Height(#VL.VarAsDecimal(#myvariant))
```

```
ENDROUTINE
```

```
End_Com
```

Result パラメータ

Resultパラメータは、対象のバリエントを小数として返します。
メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarAsInteger メソッド

VarAsIntegerは、対象のバリエントを整数として返します。
構文は以下のとおりです。

```
VarAsInteger( Subject )
```

Result パラメータ

Resultパラメータは、対象のバリエーションを整数として返します。
メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarAsReference メソッド

VarAsReferenceは、対象のバリエントをコンポーネント参照として返します。

構文は以下のとおりです。

```
VarAsReference( Subject )
```

Result パラメータ

Resultパラメータは、対象のバリエントをコンポーネント参照として返します。

メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarAsString メソッド

VarAsStringは、対象のバリエントを文字列として返します。
構文は以下のとおりです。

```
VarAsString( Subject )
```

Result パラメータ

Resultパラメータは、対象のバリエントを文字列として返します。
メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarType メソッド

VarTypeメソッドです。

VarIsBoolean メソッド

VarIsBooleanメソッドは、対象のバリエントをブール値に変換できる場合はTrueのブール値を、できない場合はFalseのブール値を返します。

構文は以下のとおりです。

```
VarIsBoolean( Subject )
```

以下は、変数がブール値であるかどうかを確認するプロパティ・ルーチンです。

```
Ptyroutine Name(Set_uSignalSelection)
Define_Map For(*input) Class(*variant) Name(#lcVariant)
If Cond(VarIsBoolean( #lcVariant ) *EQ True)
Execute Subroutine(FP_SETB) With_Parms(#USE_NAME uSignalSelection
1 #USE_OCUR #lclVariant.Boolean)
Else
Execute Subroutine(FP_SET) With_Parms(#USE_NAME uSignalSelection 1
#USE_OCUR #LCLVARIANT.String)
Endif
```

Result パラメータ

Resultパラメータは、対象のバリエーションをブール値に変換できる場合はTrueのブール値を、できない場合はFalseのブール値を返します。

メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarIsDate メソッド

VarIsDateメソッドは、対象のバリエーション内の値が日付に変換できるかを示します。

メソッドは、TrueまたはFalseを返します。構文は以下のとおりです。

```
VarIsDate( subject)
```

Result パラメータ

Resultパラメータは、対象のバリエーションを日付として返します。
メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarIsEmpty メソッド

VarIsEmptyは、対象のバリエーションに値が入っていない場合はTrueのブール値を、入っている場合はFalseのブール値を返します。

```
VarIsEmpty( Subject )
```

以下は、グリッドのセルに値が割り当てられているかどうかを確認するステートメントです。

```
If (#VL.VarIsEmpty( #grid_1.focusCell.column.EditorPAr ))  
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK Information 'Varia  
Endif
```

Result パラメータ

対象のバリエーションに値が入っていない場合はTrueのブール値を、入っている場合はFalseのブール値を返します。

メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarIsNull メソッド

VarIsNullは、対象のバリエーションに特別なNULL値が入っている場合はTrueのブール値を、入っていない場合はFalseのブール値を返します。

```
VarIsNull( Subject )
```

以下は、グリッドのセルに入っている値が不明であったり欠落していたりするかどうかを確認するステートメントです。

```
If (#VL.VarIsNull( #grid_1.focusCell.value ))  
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK Information 'Varia  
Endif
```

Result パラメータ

Resultパラメータは、対象のバリエーションに特別なNULL値が入っている場合はTrueのブール値を、入っていない場合はFalseのブール値を返します。

メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarIsNullReference メソッド

VarIsNullReferenceは、対象のバリエーションにコンポーネント参照が入っていて、その参照が*NULLコンポーネント参照である場合はTrueのbool値を、そうでない場合はFalseのbool値を返します。

VarIsNullReference(Subject)

Result パラメータ

Resultパラメータは、対象のバリエントにコンポーネント参照が入っていて、その参照が*NULLコンポーネント参照である場合はTrueのbool値を、そうでない場合はFalseのbool値を返します。

メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarIsNumber メソッド

VarIsNumberは、Subjectバリエーションを数字に変換できる場合はTrueのブール値を、そうでない場合はFalseのブール値を返します。

```
VarIsNumber( Subject )
```

以下の例では値が数字かどうか確認します。

```
If Cond(#VL.VarIsNumber( #MyVariant ) *EQ True)  
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK Information 'Varia  
Endif
```

Result パラメータ

Resultパラメータは、対象のバリエントを数字に変換できる場合はTrueのブール値を、できない場合はFalseのブール値を返します。

メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarIsReference メソッド

VarIsReferenceは、Subjectバリエントがコンポーネント参照である場合はTrueのブール値を、そうでない場合はFalseのブール値を返します。

VarIsReference(Subject)

Result パラメータ

Resultパラメータは、Subjectバリエーションがコンポーネント参照である場合はTrueのブール値を、そうでない場合はFalseのブール値を返します。メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarIsString メソッド

VarIsStringは、Subjectバリエーションを文字列に変換できる場合はTrueのブール値を、できない場合はFalseのブール値を返します。

構文は以下のとおりです。

```
VarIsString( Subject )
```

以下のifステートメントは、値が数字であるかどうかを確認します。

```
If Cond(#V1.VarIsString( #myVariant) *EQ True)
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK Information 'Varia
Else
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK Information 'Varia
Endif
```

Result パラメータ

Resultパラメータは、対象のバリエーションを文字列に変換できる場合はTrueのブール値を、できない場合はFalseのブール値を返します。

メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

VarTypeメソッドです。

VarTypeメソッドは、Subjectバリエントに現在格納されている値のタイプを返します。

構文は以下のとおりです。

```
VarType( Subject )
```

以下の例では、グリッドから返される値が文字列かどうかを確認します。

```
If Cond(#VL.VarType( #TheValue ) *EQ VarString)
Use Builtin(MESSAGE_BOX_SHOW) With_Args(OK OK Information 'Varia
Endif
```

Result パラメータ

Resultパラメータは、Subjectバリエーションに現在格納されている値のタイプを返します。

メソッドは、通常Resultパラメータを明確に指定せずに起動されます。

Subject パラメータ

メソッド適用対象のバリエーションです。

グラフ・アイテム・ベース・クラス
LANSA製品センター内部使用専用です。

ビジュアル・スタイル 言語 ベース・クラス

LANSA製品センター内部使用専用です。

ShowMessages メソッド

ShowMessagesSet メソッド

MessageSetOfFeature プロパティ

PrompterActivate イベント

Form パラメータ

PrompterDeactivate イベント

共通ダイアログ

Color ダイアログ

Save File ダイアログ

Open File ダイアログ

Color ダイアログ

Colorダイアログを使用して、エンドユーザーが色を選択できるようにします。

Color ダイアログのプロパティ

ChosenColor プロパティ

FullyOpen プロパティ

PreventFullOpen プロパティ

ChosenColor プロパティ

ChosenColor プロパティを使用して、ダイアログ内で選択した色を設定したり取り出したりします。

FullyOpen プロパティ

FullyOpenプロパティを使用して、Colorダイアログがどのように表示されるかを指定します。このプロパティの値がTrueの場合、ダイアログがカスタム色の定義と共に表示されます。

PreventFullOpen プロパティ

Colorダイアログ上の [Define Custom Colors] ボタンを無効にするには、PreventFullOpenプロパティをTrueに設定します。

Color ダイアログのメソッド

Show メソッド

Show メソッド

Showメソッドを使用して、ダイアログを表示します。

例えば次のようになります。

```
Invoke Method(#COLORDLG.Show) OKPRESSED(#boolRes) FORMOWNE
```

FormOwner パラメータ

OKPressed パラメータ

FormOwner パラメータ

FormOwnerパラメータを使用して、表示されているダイアログを所有するフォームを指定します。

OKPressed パラメータ

OKPressedパラメータは、ダイアログ上で[OK]ボタンがクリックされるとTrueを返します。

Save File ダイアログ

SaveAsダイアログを使用して、エンドユーザーがファイルを保存できるようにします。

[OverwritePrompt](#) プロパティ

OverwritePrompt プロパティ

OverwritePrompt プロパティを使用して、ファイルを上書きする際に確認メッセージを表示するかどうかを指定します。

Open File ダイアログ

Openダイアログを使用して、エンド・ユーザーがファイルを開けるようにします。

Open File ダイアログのメソッド

Show [メソッド](#)

Show メソッド

Showメソッドを使用して、Openダイアログを表示します。

以下のステートメントは、Openダイアログを表示し、フォームそのものがダイアログのオーナーであり、OKPressedパラメータを#BoolResに返すように指定します。

```
Invoke Method(#OpenFileDLG.Show) OKPRESSED(#boolRes)  
FORMOWNER(#com_self)
```

FormOwner パラメータ

FormOwnerパラメータを使用して、Openダイアログを所有するフォームを指定します。

OKPressed パラメータ

読み取り専用パラメータです。

OKPressedパラメータは、Openダイアログ上で[OK]ボタンがクリックされるとTrueを返します。

Add Filter メソッド

Name

Extension

Name

Nameプロパティを使用して、フィルター名を指定します。

Extension

Extensionプロパティを使用して、表示されるファイルの拡張子を指定します。

ExplorerStyleのプロパティ

新しいエクスプローラ・スタイルのカスタマイズ・メソッドを使用するには、ExplorerStyleプロパティを使って、ダイアログのカスタマイズを指定します。

このフラグの設定にかかわらず、省略値では、OpenとSave Asダイアログボックスはエクスプローラ・スタイルのユーザー・インターフェイスを使用します。このフラグは、MultiSelectをTrueに設定する場合にのみ必要です。

MultiSelect プロパティ

FilterIndex プロパティ

FilterCount プロパティ

FileCount プロパティ

File プロパティ

FileTitle プロパティ

InitialDir プロパティ

Title プロパティ

DefExtension プロパティ

FilterCaption プロパティ

FilterExtension プロパティ

Files プロパティ

AddFilter メソッド

MultiSelect プロパティ

MultiSelectプロパティを使用して、複数のファイルが一度に選択できるかどうかを指定します。

FilterIndex プロパティ

FilterIndex プロパティは、現在選択されているファイル・フィルターのインデックスです。

FilterCount プロパティ

FilterCount プロパティを使用して、ファイル・フィルターの数を指定します。

FileCount プロパティ

FileCountプロパティを使用して、選択されるファイルの数を取り出します。このプロパティは、ゼロまたは1つのファイルが選択されるとゼロとなり、2つのファイルが選択されると2となります。

File プロパティ

File プロパティを、File Name 編集コントロールの初期化に使うファイル名を含むバッファーへのポインターとして使用します。

MultiSelect フラグが設定され、ユーザーが複数のファイルを選択する場合、バッファーには現在のディレクトリがあります。ユーザーが1つだけファイルを選択する場合、Fileにはバスとファイル名があります。

FileTitle プロパティ

FileTitleプロパティは、パス情報なしのファイル名（拡張子付き）を返します。

InitialDir プロパティ

InitialDir プロパティを使用して、ダイアログが最初に開くディレクトリを指定します。

Title プロパティ

Titleプロパティを使用して、ダイアログのタイトルを指定します。

DefExtension プロパティ

DefExtensionを使用して、ユーザーが拡張子を入力しない場合にファイル名に追加する拡張子を指定します。

この文字列の長さに制限はありませんが、最初の3文字のみが追加されます。文字列には、ピリオド (.) を含まないようにしてください。このプロパティに値がなく、ユーザーが拡張子を入力しない場合、拡張子は追加されません。

FilterCaption プロパティ

FilterCaptionプロパティを使用して、例えば「ドキュメントファイル」というようにフィルターの記述を指定します。

FilterExtension プロパティ

FilterExtension プロパティを使用して、フィルターのパターン (*.TXT など) を指定します。

1つの表示文字列で複数のフィルターのパターンを指定する時は、セミコロンでパターンを区切ります (例 *.TXT;*.DOC;*.BAK)。

パターンの文字列には、有効なファイル名の文字とワイルドカード文字のアスタリスク (*) を組み合わせることができます。パターンの文字列には、スペースを含まないようにしてください。

Files プロパティ

Files プロパティは、2つ以上のファイルが選択されると、選択されたファイルの配列を返します。

AddFilter メソッド

AddFilterメソッドを使用して、表示するファイルを選別するフィルターを作成します。

例えば、以下のステートメントは、拡張子.docを含むファイルのみを含んだWord filesという名前のフィルターを作成します。

```
#saveFileDLG.AddFilter( 'Word files' '*.doc' )
```

ダイアログが表示される時に表示されるフィルターは、FilterIndexプロパティを使用して指定します。

HideReadOnly プロパティ

HideReadOnlyプロパティを使用して、読み取り専用ファイルを表示または非表示にします。

このプロパティの値は、TrueまたはFalseになります。

Name プロパティ

Nameプロパティを使用して、フィルター名を指定します。

Extension プロパティ

Extensionプロパティを使用して、表示されるファイルの拡張子を指定します。

フィールド・ビジュアルイゼーションのカレンダー

フィールド・ビジュアライゼーションのカレンダー カレン
ダー **primitive**

ヘルプ・ハンドラー・インターフェイス

ヘルプ・ハンドラー・インターフェイスです。

メソッド

[ProcessFocusControl](#) メソッド

[ProcessHelpRequest](#) メソッド

パラメータ

[Handled](#) パラメータ

[Requestor](#) パラメータ

[Tag](#) パラメータ

[Handled](#) パラメータ

[Requestor](#) パラメータ

プロパティ

[Tag](#) プロパティ

ProcessFocusControl メソッド

内部専用です。

ProcessHelpRequest メソッド

ProcessHelpRequestメソッドを使用して、ユーザーがF1を押した時に処理を行います。

ProcessHelpRequestの詳しい説明と例については、『*Visual LANSА* 開発者ガイド』の「[iHelpHandler Interface](#)」を参照してください。

Handled パラメータ
内部専用です。

Requestor パラメータ
内部専用です。

Tag パラメータ
LANSA内部専用です。

Handled パラメータ

HandledパラメータはTrueまたはFalseとなり、ヘルプ要求が処理されたかどうかを示します。

HandledパラメータがFalseで他の処理が指定されない場合、LANSAはリポジトリ・ヘルプを表示します。

Requestor パラメータ

Requestorパラメータは、ヘルプを要求するコンポーネントです。

Tag プロパティ

LANSA製品センター内部使用専用です。

タブシート

タブシートのプロパティ

タブシートのイベント

タブシートのメソッド

タブシートのプロパティ

[DockAllowedPositions](#) プロパティ

[DockAllowUndock](#) プロパティ

[DockCloseButton](#) プロパティ

[DockPosition](#) プロパティ

DockAllowedPositions プロパティ

タブ・フレームワーク・プロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#) を参照してください。

タブ・シートを添付 (ドッキング) できる画面のパーツを指定するには、DockAllowedPositions プロパティを使用します。

値はLeft, Right, Top, Bottom および None です。すべての位置が選択できます。None は、複数選択のオプションを上書きし、タブ・シートがどこにもドッキングできないことを指定します。

このプロパティは、ユーザーの操作にのみ適用され、プログラムでの変更には対応しないことに注意してください。そのため、DockPosition (Left) と DockAllowedPositions (None) のタブ・シートは、DockLeft のタブ・シートとして表示されます。

DockAllowUndockプロパティ

タブ・フレームワーク・プロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

DockAllowUndockプロパティを使用して、タブ・シートをタブ・フォルダから移動（切り離し）できるようにするか指定します。

このプロパティは、TrueまたはFalseに設定できます。

プロパティがTrueの場合、タブ・シート上にドッキング・バー（二重線）が表示されます。ユーザーは、ドッキング・バーからドラッグしてタブ・シートを切り離します。

LDockAllowUndockメソッドの効果を確認するには、以下のコードをコピー・貼り付けしてください。

```
Function Options(*DIRECT)
BEGIN_COM ROLE(*EXTENDS #PRIM_FORM) CAPTION('Docking
Framework Sample') CLIENTHEIGHT(378) CLIENTWIDTH(549)
HEIGHT(405) LAYOUTMANAGER(#ATLM_1) LEFT(418) TOP(163)
WIDTH(557)
DEFINE_COM CLASS(#PRIM_TAB) NAME(#TAB_1)
DISPLAYPOSITION(1) HEIGHT(378) LEFT(0) LEFTTABWIDTH(154)
PARENT(#COM_OWNER) TABPOSITION(1) TABSTOP(False) TOP(0)
VISUALSTYLE(#VS_TAB) WIDTH(549)
DEFINE_COM CLASS(#PRIM_ATLM) NAME(#ATLM_1)
DEFINE_COM CLASS(#PRIM_ATLI) NAME(#ATLI_1)
ATTACHMENT(Center) MANAGE(#TAB_1) PARENT(#ATLM_1)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_1)
CAPTION('Page1') DISPLAYPOSITION(1) DOCKCLOSEBUTTON(True)
DOCKPOSITION(Left) HEIGHT(349) LEFT(4) PARENT(#TAB_1)
TABPOSITION(1) TABSTOP(False) TOP(25) WIDTH(146)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_2)
CAPTION('Page2') DISPLAYPOSITION(2) DOCKCLOSEBUTTON(True)
DOCKPOSITION(Left) HEIGHT(349) LEFT(4) OPENED(True)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(146)
DEFINE_COM CLASS(#PRIM_TBSH) NAME(#TBSH_3)
CAPTION('Page3') DISPLAYPOSITION(1) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(3) TABSTOP(False) TOP(25)
WIDTH(382)
```

```
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_4)
CAPTION('Page4') DISPLAYPOSITION(2) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(2) TABSTOP(False) TOP(25)
WIDTH(382)
DEFINE_COM CLASS(#PRIM_TBESH) NAME(#TBESH_5)
CAPTION('Page5') DISPLAYPOSITION(3) HEIGHT(349) LEFT(4)
PARENT(#TAB_1) TABPOSITION(1) TABSTOP(False) TOP(25)
WIDTH(382)
DEFINE_COM CLASS(#PRIM_PHBN) NAME(#PHBN_1)
CAPTION('Make Page 1 and Page 2 Dockable') DISPLAYPOSITION(1)
LEFT(32) PARENT(#TBESH_3) TABPOSITION(1) TOP(171) WIDTH(321)
EVTROUTINE HANDLING(#PHBN_1.Click)
set com(#tbsh_1 #tbsh_2) dockallowundock(True)
ENDROUTINE
End_Com
```

DockCloseButton プロパティ

タブ・フレームワーク・プロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#) を参照してください。

DockCloseButtonプロパティを使用して、[Close] ボタンをタブ・シートに表示するかどうかを指定します。タブ・シートに [Close] ボタンがあれば、閉じることができます。

このプロパティは、TrueまたはFalseに設定できます。

DockPosition プロパティ

タブ・フレームワーク・プロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#)を参照してください。

DockPositionプロパティを使用して、タブ・シートの添付位置を指定します。

タブ・シートは画面の中央、左、右または上にドッキングできます。

タブ・シートのDockPositionプロパティを使用して、1つのタブ・フォルダからタブ・フレームワーク・アプリケーションを構築することができます。

タブシートのイベント

AutoHideClick イベント

CloseClick イベント

Docked イベント

Undocked イベント

AutoHideClick イベント

タブ・フレームワーク・プロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#) を参照してください。

AutoHideClickイベントは、タブ・シートの [添付] ボタンをクリックすると起動されます。

[添付]ボタンは、タブ・フォルダのBottomAllowAutohide, LeftAllowAutohide, RightAllowAutohideまたはTopAllowAutohideプロパティがTrueに設定される場合のみ表示されます。

CloseClick イベント

タブ・フレームワーク・プロパティです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#) を参照してください。

CloseClick イベントは、ユーザーがタブ・シートの [Close] ボタンをクリックすると起動されます。

[Close] ボタンは、DockCloseButton プロパティが True の場合表示されません。

Docked イベント

タブ・フレームワーク・イベントです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#) を参照してください。

Dockedイベントは、タブ・シートがある位置に添付 (ドッキング) されるときに起動されます。このイベントは、ユーザーがタブ・シートをドッキングを解除した位置からからドッキングするとき、およびドッキングされている場合にDockPositionを変更したとき (DockPositionをLeftからRightに変更するなど) に起動されます。

Undocked イベント

タブ・フレームワーク・イベントです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#) を参照してください。

Undocked イベントは、タブ・シートがある位置からデタッチ (装着解除) されるときに起動されます。

タブシートのメソッド

[HideSheet メソッド](#)

[ShowSheet メソッド](#)

HideSheet メソッド

タブ・フレームワーク・メソッドです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#) を参照してください。

HideSheetメソッドを使用してタブ・シートを非表示にします。これは、プログラム上、[Close] ボタンのクリックと同じです。

ShowSheet メソッド

タブ・フレームワーク・メソッドです。タブ・フレームワーク・アプリケーションの例については、[タブ・フォルダー](#) を参照してください。

ShowSheetメソッドを使用して、[Close] ボタンを使って非表示にされたタブ・シートを再表示します。

Activate パラメータ

Activateパラメータを使用して、再表示されたタブ・シートを有効にするかどうかを指定します。

このプロパティの値は、TrueまたはFalseになります。

影響分析リスト・フィルター

フィルターは影響分析リスト内で使われ、検索条件を定義します。異なる種類のフィルターです。

- 日付フィルター
- 文字列フィルター
- 数字フィルター
- 1つのドロップダウン・フィルター
- 複数のドロップダウン・フィルター

RDMLX 使用可能 がすべてのコンパイル可能なオブジェクトに適用されるように、複数のオブジェクトに適用されるフィルターがあります。

ファイル・タイプ がファイルにのみ適用されるように、1つのオブジェクト・タイプにのみ適用されるフィルターもあります。

Like 演算子を使用する場合、ワイルドカードの値は '*' です。検索値の最初、最後および検索値内のどこにでも使えます。

Equal 演算子は '*' を文字の '*' として (ワイルドカードではなく) 扱いません。

検索値を持ったどんなソース行でも含むすべてのオブジェクトを返すフィルター *Text Search* はこの規則には当てはまりません。

詳しくは「[影響分析リスト定義の変更](#)」を参照してください。

リボン

リボンはプログラム機能をフォーム最上部の一連のタブまたはシートにまとめるコマンド・バーです。リボンは従来のメニューバーやツールバーを一つのコントロールにまとめます。

リボンの例はバージョン13 LANSА IDEおよびMicrosoft Office 2007以降に見られます。

すべてのアプリケーションの環境がリボンに適しているわけではないため、リボンが要件に対する最善のソリューションかどうか、慎重に検討しなければなりません。

リボンの幅は常に現在使用されているコンポーネントの幅と同じで、高さは固定されています。

Minimized

True

リボンは最小化して表示されます。シートのキャプションのみ表示されます。

False

リボン全体が表示されます。

OpenPage

「リボン・シート」を参照してください。 (#Prim_RBBN.Sheet)
アクティブ・リボン・シートへのアクセスができるようにします。

QuickAccessToolbarOnTop

True

クイック・アクセス・ツールバーはリボンの上に表示されます。ガラスの外観のフォームでは、ツールバーはタイトルバー領域に表示されません。

False

クイック・アクセス・ツールバーはリボンの真下に表示されます。

Access Key

アクセス・キーは、リボンの特定のコントロールに添付されるショートカット・コマンド（F3, Ctrl+Sなど）とリボンの簡単アクセスのKeyTipを定義します。

Control

KeyTipおよびショートカットが適用される「コントロール」のインスタンス（#Prim_CTRL）を参照してください。

Application Menu

アプリケーション・メニュー（#Prim_RBBN.ApplicationMenu）は、ブ
ルーの[アプリケーション]ボタンがクリックされた時に表示されるポッ
プアップの左側に表示されます。

特別なデザイン・ルールはないため、アプリケーション・メニューはブ
レーンパネルです。動作や表示内容はすべて開発者に依存します。

```
Define_Com Class(#Prim_rbbn) Name(#Ribbon) Displayposition(1)  
Height(140) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(8)  
Width(1033)
```

```
Define_Com Class(#Prim_Rbbn.ApplicationMenu)  
Name(#ApplicationMenu) Caption('File') Displayposition(1) Height(80)  
Keytip('F') Left(30) Parent(#Ribbon) Tabposition(8) Top(0) Width(100)
```

```
Define_Com Class(#Prim_Rbbn.ApplicationMenuContent)  
Name(#ApplicationMenuContent) Displayposition(5) Height(80)  
Parent(#Ribbon) Tabposition(7) Top(0)
```

```
Define_Com Class(#Prim_Rbbn.ApplicationMenuFooter)  
Name(#ApplicationMenuFooter) Displayposition(4) Height(20) Left(30)  
Parent(#Ribbon) Tabposition(6) Top(0)
```

Caption

アプリケーション・メニュー・キャプションは、ブルーの[アプリケーション・メニュー]ボタン、主に[ファイル]アイテムに表示されます。

アクセス・キー (#Prim_RBBN.AccessKey) とKeyTipを使うことによりリボンキーボード・ナビゲーションが行なわれるので、& をテキストに埋め込む必要はありません。

Application Menu Content

アプリケーション・メニュー・コンテンツ

(#Prim_RBBN.ApplicationMenuContent) は、ブルーの[アプリケーション]ボタンがクリックされた時に表示されるポップアップの右側に表示されます。

アプリケーション・コンテンツ・ページで何がアクティブかによってコンテンツが全く異なることがあるため、多くの異なるコンテンツ・インスタンスを作る必要のある可能性があります。アプリケーション・メニューと同様に、コンテンツはプレーン・パネルで、デザインと動作は、最終的には開発者に依存します。

```
Define_Com Class(#Prim_rbbn) Name(#Ribbon) Displayposition(1)
Height(140) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(8)
Width(1033)
Define_Com Class(#Prim_Rbbn.ApplicationMenu)
Name(#ApplicationMenu) Caption('File') Displayposition(1) Height(80)
Keytip('F') Left(30) Parent(#Ribbon) Tabposition(8) Top(0) Width(100)
Define_Com Class(#Prim_Rbbn.ApplicationMenuContent)
Name(#ApplicationMenuContent) Displayposition(5) Height(80)
Parent(#Ribbon) Tabposition(7) Top(0)
Define_Com Class(#Prim_Rbbn.ApplicationMenuFooter)
Name(#ApplicationMenuFooter) Displayposition(4) Height(20) Left(30)
Parent(#Ribbon) Tabposition(6) Top(0)
```

Application Menu Footer

アプリケーション・メニュー・フッター

(#Prim_RBBN.ApplicationMenuFooter) は、ブルーの[アプリケーション] ボタンがクリックされた時に表示されるポップアップの下に表示されます。

アプリケーション・メニューやコンテンツと同様、フッターはプレーン・パネルで、デザインと動作は最終的に開発者に依存します。

```
Define_Com Class(#Prim_rbbn) Name(#Ribbon) Displayposition(1)
Height(140) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(8)
Width(1033)
Define_Com Class(#Prim_Rbbn.ApplicationMenu)
Name(#ApplicationMenu) Caption('File') Displayposition(1) Height(80)
Keytip('F') Left(30) Parent(#Ribbon) Tabposition(8) Top(0) Width(100)
Define_Com Class(#Prim_Rbbn.ApplicationMenuContent)
Name(#ApplicationMenuContent) Displayposition(5) Height(80)
Parent(#Ribbon) Tabposition(7) Top(0)
Define_Com Class(#Prim_Rbbn.ApplicationMenuFooter)
Name(#ApplicationMenuFooter) Displayposition(4) Height(20) Left(30)
Parent(#Ribbon) Tabposition(6) Top(0)
```

Contextual Group

コンテキスト・グループは、同じグループに属するシートを集めたり (#Prim_RBBN.Sheet)、単にシートの詳細な情報をタイトル・バーに表示したりするのに使うことができます。

Microsoft Wordでは、コンテキスト・グループをテーブルなどの機能に用いています。ドキュメントでテーブルが選択されると、追加の一組のリボン・シートが表示されます。

Caption

コンテキスト・グループを構成するシートの上に、見出しとしてテキストが表示されます。

Visible

表示プロパティは、そのグループに属するすべてのシートの表示状態を上書きします。

True

このコンテキスト・グループに属するすべてのシートは、そのシートが Visible (False)でない限り表示されます。

False

このコンテキスト・グループのすべてのシートは、条件にかかわらず非表示です。

Ribbon Group

グループはシートのサブセットです。ほとんどのシートには、リボン上のコマンドを関連のサブジェクトにまとめる複数の異なるグループがあります。

コントロールは、グループに属する時のみリボン上に表示されます。リボンのすべてのパーツと同様に、各グループはプレーン・パネルです。デザインは開発者に依存します。

```
Define_Com Class(#Prim_rbbn) Name(#Ribbon) Displayposition(1)
Height(140) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(8)
Width(1033)
Define_Com Class(#Prim_rbbn.Sheet) Name(#Sheet1) Caption('Sheet1')
Displayposition(2) Height(91) Left(0) Parent(#Ribbon) Tabposition(1)
Top(49) Width(1033)
Define_Com Class(#Prim_rbbn.Group) Name(#Sheet1Group1)
Caption('Clipboard') Dialogbutton(True) Displayposition(1) Height(91)
Left(12) Parent(#Sheet1) Tabposition(1) Top(0) Width(100)
Define_Com Class(#Prim_rbbn.Group) Name(#Sheet1Group2)
Caption('Styles') Displayposition(2) Height(91) Left(124) Parent(#Sheet1)
Tabposition(2) Top(0) Width(100)
Define_Com Class(#Prim_rbbn.Group) Name(#Sheet1Group3)
Caption('Editing') Displayposition(3) Height(91) Left(236) Parent(#Sheet1)
Tabposition(3) Top(0) Width(100)
```

DialogButtonクリック・イベント

グループの[ダイアログ]ボタンがクリックされた時に起動します。

DialogButton プロパティ

True

グループの右下隅に[ダイアログ]ボタンを表示します。

False

[ダイアログ]ボタンを非表示にします。

Help Toolbar

リボン・ヘルプ・ツールバーは、リボンの右上隅に表示されます。リボンのすべてのパーツと同様、ヘルプ・ツールバーはプレーン・パネルです。デザインは開発者に依存します。

Image プロパティ

「ベースLANSAグラフィック」(#Prim_FLBX)を参照してください。
グループが折りたたまれた時に表示される32x32のイメージを示します。

Quick Access Toolbar

クイック・アクセス・ツール・バーは、リボンの左側に表示されます。通常は、[保存]など一般に使われるコマンドに使われる小さいイメージ行を表示します。リボンの上下どちらに表示されるかはリボンのQuickAccessToolbarOnTop プロパティによって制御されています。リボンのすべてのパーツと同様、クイック・アクセス・ツールバーはプレーン・パネルです。デザインは開発者に依存します。

Ribbon Sheet

リボン・シートはリボン・グループのコンテナです。(#Prim_RBBN.Group)シートは、リボンを構成するタブ表示のページを定義します。

KeyTip

KeyTipは、キーボード使用中にリボンの特定のコマンドにアクセスするために使われる文字列です。

KeyTipは通常頭文字1字ですが、より複雑なリボンでは2文字を使用することが望まれます。

Contextual Group

「コンテキスト・グループ」を参照してください。

(#Prim_RBBN.ContextualGroup)

コンテキスト・グループが表示された時に表示できるセットにシートをまとめるのに使われます。

ガラス・コントロール

Glass プロパティ

「ガラス」のインスタンスを参照してください。 (#Prim_FORM.Glass)
フォームをガラスに設定することにより、タイトル・バーとすべての枠線が削除され、フォームの背景が半透明になります。

Form Glass

フォームをガラスに設定することにより、タイトル・バーとすべての枠線が削除され、フォームの背景が半透明になります。

標準フォームはタイトル・バー、枠線および100%不透明な背景のための領域を定義しています。コンテンツで最初にアクセス可能なピクセル、左(0)上(0)は、タイトル・バーの下、左枠線の右にあります。

フォームにガラスのインスタンスを与えることにより、開発者はフォームのタイトルバーと枠線を使用できるようになり、0,0は、フォームの左上角に移動します。実際には、枠線やタイトルバーがなく、開発者はすべてのピクセルにアクセスできます。

ガラスはカバーの下でフォームに適用される枠線スタイルです。これはガラスを使用するようにフォームを変更することができるようにするには、FrameStyle プロパティがNoneでなければならないということを意味します。これにより、子コントロールのみ表示される、完全に非表示のフォームとなります。

Glass Style プロパティ

ガラス・スタイルには、可能な値が2つあります。

All

余白の指定値にかかわらず、フォーム全体がガラスになります。

Partial

指定された余白の外の領域のみガラス表示されます。これによりタイトル・バー領域をガラスにできますが、フォームの他の部分は通常通り動作します。

MarginBottom プロパティ

MarginTop, MarginLeftおよびMarginRightプロパティとともに、ガラスの外観を与えられるフォームの領域を記述します。

プロパティで記述される長方形の外の領域がガラスになります。内側の領域は黒く、100%不透明になります。

MarginLeft プロパティ

MarginTop, MarginBottomおよびMarginRightプロパティとともに、ガラスの外観を与えられるフォームの領域を記述します。

プロパティで記述される長方形の外の領域がガラスになります。内側の領域は黒く、100%不透明になります。

MarginRight プロパティ

MarginTop, MarginBottomおよびMarginLeftプロパティとともに、ガラスの外観を与えられるフォームの領域を記述します。

プロパティで記述される長方形の外の領域がガラスになります。内側の領域は黒く、100%不透明になります。

MarginTop プロパティ

MarginBottom, MarginLeftおよびMarginRight プロパティとともに、ガラスの外観を与えられるフォームの領域を記述します。

プロパティで記述される長方形の外の領域がガラスになります。内側の領域は黒く、100%不透明になります。

ShowInTaskBar プロパティ

FormStyleプロパティとともに、ShowInTaskBarは、フォームをWindowsタスクバーにアイテムとして表示するかどうかを決定します。

可能な値は3つあります。

FormStyle

これは省略値で、Formstyleプロパティで決定された通りに動作を行いません。

Yes

Windows タスクバーにこのフォームを表示します。

No

Windows タスクバーにこのフォームを表示しません。
スプラッシュ画面を作成する時に便利です。

TaskBar プロパティ

「TaskBar」のインスタンスを参照してください。

(#Prim_FORM.TaskBar)

TaskBarインスタンスを作成することにより、フォームとWindows タスクバーに表示されたアイテムとのやり取りが可能になります。

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Taskbar(#Taskbar)
Define_Com Class(#Prim_Form.Taskbar) Name(#Taskbar) Hint('My
Application Hint')
```

TaskBar

TaskBarインスタンスを作成することにより、フォームとWindows タスクバーに表示されたアイテムとのやり取りが可能になります。

```
Begin_Com Role(*EXTENDS #PRIM_FORM) Taskbar(#Taskbar)
Define_Com Class(#Prim_Form.Taskbar) Name(#Taskbar) Hint('My
Application Hint')
```

Windowsタスクバーは、アプリケーションの動作に関する追加ランタイム・フィードバックを提供するために使うことができます。タスクバーは、プログレスバーのように動作することができ、オーバーレイされたイメージと共にアプリケーション・アイコンを拡大し、アプリケーションの状態を示すことができます。これは、アプリケーションが最小化されたり、他のアプリケーションの背後に隠されたりしている時に特に便利です。

Hint プロパティ

タスクバー・アイテムの上にマウスを置いた時に表示されるヒントのテキストです。

OverlayImage プロパティ

どんなLANSAビットマップ (Prim_BMP) またはアイコン (#Prim_ICON) にもなることができる「ベース・グラフィック」(#Prim_Flbx) を参照してください。

指定されたイメージは、アプリケーション・アイコンをオーバーレイしてタスクバー・アイテム上の右下角に配置されます。提供されたイメージのサイズにかかわらず、イメージは 16 x 16 で表示されます。

ProgressStyle プロパティ

ProgressValue プロパティとともに、アプリケーションの進捗を示すために使われている時、タスクバー・アイテムの動作を決定します。

ProgressStyleには、可能な値が5つあります。

Error

タスクバー・アイテムは、赤で表示され、アプリケーションにエラーがあることを示します。PercentageValue プロパティの値は表示されます。

Indeterminate

タスクバー・アイテムは、繰り返しのパターンとともにアニメートします。

これは、エンド・タイムが不明で長いプロセスを実行する時に使用できます。

None

進捗状況は表示されず、タスクバー・アイテムが通常通り表示されません。

Normal

タスクバー・アイテムは、プログレスバーのように動作し、ProgressValue プロパティの値を表示します。

Paused

タスクバー・アイテムは、黄色で表示され、アプリケーションが一時停止していることを示します。PercentageValue プロパティの値は表示されます。

Progress Value

ProgressStyleプロパティとともに、アプリケーションの進捗を示すために使われている時、タスクバー・アイテムの動作を決定します。

プログレス値はパーセントで0-100までの値を表示するのに使えます。

アニメーション

アニメーションは、Visual LANSAコントロールを移動、不透明度を変更、サイズを変更させることなどができます。

アニメーション (#Prim_ANIM)は、すべての個々のアニメーション・アクティビティのコンテナとして使用されます。多くの異なるパーツを包含することができ、多くの異なるコントロールを同時にアニメートすることができます。以下のサンプルはボタンを時計回りに回転させます。同時に、2番目のボタンが、高さも変更され半時計回りに回転されます。

```
Define_Com Class(#prim_anim) Name(#Animation)
Define_Com Class(#prim_anim.Rotate) Name(#Rotate1) Angle(180)
Manage(#Button) Parent(#Animation)
Define_Com Class(#prim_anim.Rotate) Name(#Rotate2) Angle(-180)
Manage(#Button2) Parent(#Animation)
Define_Com Class(#prim_anim.Rotate) Name(#Height) Height(50)
Manage(#Button2) Parent(#Animation)
```

別のスレッドで実行され、他の処理が影響なく継続できるため、スプラッシュ画面やビジー処理に最適になります。アニメーションが開始されると、実行中は修正できません。

アニメーションは視覚効果で、アニメーションが完了するまでコンポーネントの状態を修正しません。

```
Define_Com Class(#prim_anim) Name(#Animation)
Define_Com Class(#prim_anim.Rotate) Name(#Rotate) Angle(180)
Duration(5000) Manage(#Button) Parent(#Animation)
```

上記のアニメーションにおいてボタンは180度回転され、完了まで5秒かかります。アニメーション時にボタンがアクティブな状態でクリックできる間、そのRotationプロパティは、アニメーションが完了するまで元の値です。アニメーションは、コントロールに特定のプロパティを設定することにより、コントロールをミラー化できない状態にすることはできません。

アニメーションは、レイアウト・マネージャーの影響を受けません。レイアウト・マネージメントの対象となるコントロールは、通常通りアニメーション開始の前に表示されます。アニメーションが開始されるとアニメーションの命令に従い、アニメーションが終了するとレイアウトの

ルールに従う状態に戻ります。しかし、簡単にするため、アニメーションはレイアウト・マネージャーが使われていない環境での使用に最も適しています。

すべてのVisual LANSACONTROLの機能として、フェードなど、よく使われるアニメーションの使用が可能です。

Ended

終了イベントはアニメーションが完了した時、起動されます。

Starting

開始イベントはアニメーションが開始する直前に起動されます。

Start

Startメソッドはアニメーションを開始または再開します。

アニメーションは一度開始すると変更できないため、アニメーション実行中にStartメソッドを実行すると、アニメーションをリセットしてから再度開始します。

アニメーションがすでに進行中かどうか判断するには、IsAnimatingプロパティを使用します。

Stop

Stopメソッドは、アニメーションを停止しすべてのコントロールをアニメーションの開始前の状態に戻します。

IsAnimating

アニメーションが実行中かどうか決定します。

IsAnimatingは、後続の実行が現在の実行を中断するのを防ぐため、同じアニメーションが繰り返し使われているシナリオで有効です。

```
Evtroutine Handling(#Button.Click)
If (*Not #Animation.isAnimating)
#Animation.Start
Endif
Endroutine
```

True

アニメーションは現在進行中です。

False

アニメーションは現在停止しています。

Animation Item

すべての個々のアニメーションのベース・クラスです。

Duration

アニメーションの完了にかかるミリ秒数です。
省略値は1000です。

StartTime

アニメーションの開始待ちのミリ秒数です。

アニメーションの各パーツは個別に実行されるので、各パーツが開始する際、まとめるのはユーザーです。

Height アニメーション

Heightアニメーション (#Prim_Anim.Height)は、制御対象のコントロールの現在の高さからHeightプロパティで指定された高さへの変更をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)
Define_Com Class(#prim_anim.Height) Name(#Step1) Height(100)
Manage(#Button) Parent(#Animation)
```

Height

アニメーションが完了する時の制御対象のコントロールの高さです。

Manage

アニメーションが影響を与える「コントロール」(#Prim_CTL)を参照してください。

Move アニメーション

Moveアニメーション (#Prim_Anim.Move)は、制御対象のコントロールのLeftおよびTopプロパティに指定されたピクセル数の移動をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)  
Define_Com Class(#prim_anim.Move) Name(#Step1) Left(100)  
Manage(#Button) Parent(#Animation) Top(-50)
```

Left

コントロールのLeftプロパティに追加するピクセル数です。

正数の場合、コントロールを右に移動させ、Leftプロパティを増加させます。

Top

コントロールのTopプロパティに追加するピクセル数です。

正数の場合、コントロールを下に移動させ、Topプロパティを増加させます。

MoveFrom アニメーション

MoveFromアニメーション (#Prim_Anim.MoveFrom) は、制御対象のコントロールの、LeftおよびTopプロパティに指定する位置から現在の位置への移動をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)
```

```
Define_Com Class(#prim_anim.MoveFrom) Name(#Step1) Left(100)
```

```
Manage(#Button) Parent(#Animation) Top(100)
```

Left

Topプロパティとともに、アニメーションの開始位置を定義します。

Top

Leftプロパティとともに、アニメーションの開始位置を定義します。

MoveTo アニメーション

MoveToアニメーション (#Prim_Anim.MoveTo) は、制御対象のコントロールの、現在の位置からLeftおよびTopプロパティに指定する位置への移動をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)
Define_Com Class(#prim_anim.MoveTo) Name(#Step1) Left(100)
Manage(#Button) Parent(#Animation) Top(100)
```

Left

Topプロパティとともに、アニメーションの終了位置を定義します。

Top

Leftプロパティとともに、アニメーションの終了位置を定義します。

Opacity アニメーション

Opacity アニメーション (#Prim_Anim.Opacity) は、コントロールの現在の不透明度から Opacity プロパティに指定された値への不透明度の変更をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)  
Define_Com Class(#prim_anim.Opacity) Name(#Step1) Manage(#Button)  
Opacity(0) Parent(#Animation)
```

Opacity

アニメーションが完了する時の制御対象のコントロールの不透明度です。

Rotate アニメーション

Rotate アニメーション (#Prim_Anim.Rotate)は、コントロールの現在の回転から Angle プロパティに指定された値への回転の変更をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)  
Define_Com Class(#prim_anim.Rotate) Angle(360) Name(#Step1)  
Manage(#Button) Parent(#Animation)
```

Angle

コントロールを回転させる角度です。

正数は時計回りに、負数は反時計回りにコントロールを回転させます。

OriginLeft

OriginTopプロパティとともに、コントロールが回転する位置を決定します。

OriginLeftはパーセントです。

OriginTop

OriginLeftプロパティとともに、コントロールが回転する位置を決定します。

OriginTopはパーセントです。

Scale アニメーション

Scaleアニメーション (#Prim_Anim.Rotate)は、コントロールの現在の目盛りからScaleHeightおよびScaleWidthプロパティに指定された値へのスケールの変更をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)  
Define_Com Class(#prim_anim.Scale) Name(#Step1) Manage(#Button)  
Parent(#Animation) ScaleHeight(125) Scalewidth(125)
```

Origin Left

OriginTopプロパティとともに、コントロールのスケールの開始位置を決定します。

OriginLeftはパーセントです。

OriginTop

OriginLeftプロパティとともに、コントロールのスケールの開始位置を決定します。

OriginTopはパーセントです。

ScaleHeight Left

アニメーションが完了する時の制御対象のコントロールのスケールの高さです。

ScaleHeightはパーセントです。

ScaleWidth Left

アニメーションが完了する時の制御対象のコントロールのスケールの幅です。

ScaleWidthはパーセントです。

Skew アニメーション

Skewアニメーション (#Prim_Anim.Skew) は、コントロールの現在の傾斜からAngleLeftおよびAngleTopプロパティに指定された値への傾斜の変更をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)
Define_Com Class(#prim_anim.Skew) AngleLeft(45) AngleTop(45)
Name(#Step1) Manage(#Button) Parent(#Animation)
```

Angle Left

制御対象のコントロールに適用される横方向の傾斜の角度を定義します。

Angle Top

制御対象のコントロールに適用される縦方向の傾斜の角度を定義します。

Origin Left

OriginTopプロパティとともに、コントロールが傾斜する位置を決定します。

OriginLeftはパーセントです。

Origin Top

OriginLeftプロパティとともに、コントロールが傾斜する位置を決定します。

OriginLeftはパーセントです。

Visibility アニメーション

Visibility アニメーション (#Prim_Anim.Visibility) は制御対象のコントロールの visible プロパティの変更をアニメートします。

Opacity アニメーションと同様、コントロールはアニメーション時に次第に明るくなったり暗くなったりします。

```
Define_Com Class(#prim_anim) Name(#Animation)
```

```
Define_Com Class(#prim_anim.Visibility) Name(#Step1) Manage(#Button)
```

```
Parent(#Animation)
```

Visible

True

コントロールのvisibleプロパティはtrueで不透明度は100です。

False

コントロールのvisibleプロパティはfalseで不透明度は0です。

Width アニメーション

Widthアニメーション (#Prim_Anim.Width) は、制御対象のコントロールの現在の幅からWidthプロパティで指定された幅への変更をアニメートします。

```
Define_Com Class(#prim_anim) Name(#Animation)  
Define_Com Class(#prim_anim.Width) Name(#Step1) Manage(#Button)  
Parent(#Animation) Width(100)
```

Width

アニメーションが完了する時の制御対象のコントロールの幅です。

Transition アニメーション

Transitionは、FromとToのパラメータに指定された、あるコントロールから別のコントロールへの変更をアニメートします。通常はリストのアイテムが選択され、異なるパネルがアクティブ化されたアプリケーションで使用されます。

Transitionは、2つのコントロールのイメージを選び、それらをアニメーション時に混合することにより動作します。一番よい結果を得るためには、2つのコントロールは同じ物理的範囲を占めなければなりません。アニメーションの終了時に、FromコントロールのvisibleプロパティはfalseになりToコントロールのvisibleプロパティはtrueになります。

```
Define_Com Class(#prim_anim) Name(#Animation)
Define_Com Class(#prim_anim.Transition) Name(#Step1) Duration(2000)
From(#Panel1) Parent(#Animation) To(#Panel2) Transitiontype(Wave)
```

From

「コントロール」(#Prim_CTRL)を参照してください。

Fromは、アニメーションの完了時にvisibleプロパティがfalseになるコントロールを指定します。

To

「コントロール」(#Prim_CTRL)を参照してください。

Toは、アニメーションの完了時にvisibleプロパティがtrueになるコントロールを指定します。

TransitionType

複数の切り替えの使用が可能です。
デフォルトはFadeです。

コントロールと複合

Origin

Originセクター内で指定された変数は、その上でイベントが最初に起動されたコントロールへの参照を含みます。

Handled

Handledセクターにより、ユーザーはルーティング・イベントが親チェーンに沿ってこれ以上先に進まないように停止することができます。省略値はFalseです。

True

イベントはこのイベント・ルーチン内で処理されており、親にルーティングされません。

False

イベントはこのイベント・ルーチン内で完全には処理されておらず、親にルーティングされます。

FadeIn メソッド

DirectX のみ

FadeInは、指定された期間、コントロールのOpacityプロパティを現在の値から100に切り替えるアニメーション・メソッドです。アニメーションが完了すると、VisibleプロパティはTrueになります。

すべてのアニメーションのように別のスレッド内で実行され、ほかの処理が継続できます。

Delay

アニメーションの処理を遅延するミリ秒数です。省略値は0です。

Duration

アニメーションの完了にかかるミリ秒数です。省略値は250です。

FadeOut メソッド

DirectX のみ

FadeOutは、指定された期間、コントロールのOpacityプロパティを現在の値から0に切り替えるアニメーション・メソッドです。アニメーションが完了すると、VisibleプロパティはFalseになります。

すべてのアニメーションのように別のスレッドで実行され、ほかの処理が継続できます。

Delay

アニメーションの処理を遅延するミリ秒数です。省略値は0です。

Duration

アニメーションの完了にかかるミリ秒数です。省略値は250です。

Scale メソッド

DirectX のみ

Scaleは、指定された期間、コントロールのScaleWidthとScaleHeightプロパティを現在の値から指定された値に切り替えるアニメーション・メソッドです。 アニメーションが完了した時、コントロールのScaleWidthとScaleHeightプロパティは、指定された値と一致します。

すべてのアニメーションのように別のスレッドで実行され、ほかの処理が継続できます。

ScaleWidth

アニメーションの対象ScaleWidth値です。省略値は100です。

ScaleHeight

アニメーションの対象ScaleHeight値です。省略値は100です。

Delay

アニメーションの処理を遅延するミリ秒数です。省略値は0です。

Duration

アニメーションの完了にかかるミリ秒数です。省略値は250です。

MouseEnter イベント

DirectX のみ

MouseEnterは、マウスがコントロールの外枠に入ると起動されます。

MouseHover イベント

DirectX のみ

MouseHoverは、マウスがコントロール上で既定の時間静止すると起動されます。イベントは一度起動され、マウスが再度入ったコントロールを離れた時だけ再度起動されます。

MouseLeaveイベント

DirectXのみ

MouseLeaveは、マウスがコントロールの外枠を離れると起動されます。マウスが子コントロールの範囲に入ると、親の上では起動されません。

Hint Popup プロパティ

DirectX のみ

Hint Popupは、省略値のポップアップ・テキスト・ボックスの代わりに PopupPanel (#prim_ppnl) インスタンスが使用できるようにします。これにより、開発者は、ヒントのコンテンツとフォーマットを完全に制御できるようになります。

```
Define_Com Class(#prim_phbn) Name(#Button) Caption('Click')  
Hintpopup(#ButtonHintPanel)  
Define_Com Class(#Prim_ppnl) Name(#ButtonHintPanel)  
Content(#MyHintPanelContent)
```

Hint Title プロパティ

DirectX のみ

指定されると、ヒント・タイトルはヒント・ウィンドウの上部に強調されたテキストで表示されます。ヒントの残りの部分は、通常通り下部に表示されます。

IsAnimating プロパティ

DirectX のみ

コントロールをアニメーションにするかどうかを決定するのに使用されます。

通常、アニメーションが再開されないようにするのに使用されます。

True

コントロールはアニメーションにされていません。

False

コントロールはアニメーションにされています。

MouseOver プロパティ

DirectX のみ

マウスがコントロールのすぐ上にあるかどうかを決定します。マウスが子コントロールの上にある場合、MouseOverはFalseになります。

コントロールの外観を変更するためにMouseEnter/Leaveコードが実行されているが、他のファクタもコントロールの状態に影響を与えている場合は、MouseOverが便利です。

True

マウスはコントロールのすぐ上にあります。

False

マウスはコントロールのすぐ上にはありません。

MouseOverStyle プロパティ

DirectX のみ

マウスがコントロールの物理的範囲の中に入るとコントロールに適用されるスタイル (#Prim_vs.Style) を示します。マウスがコントロールを離れると、スタイルは消去されます。

MouseOverStyleは、多くのMouseEnterおよび対応するMouseLeaveイベントのコード化を不要にし、かわりに単純な宣言を可能にします。

MouseOverStyles プロパティ

DirectX のみ

マウスがコントロールの物理的範囲の中に入るとコントロールに適用されるスタイル (#Prim_vs.Style) のコレクションです。マウスがコントロールを離れると、スタイルは消去されます。

MouseOverStylesは、より複雑なプログラム上の外観の変更のコード化を可能にします。開発者は、1つの宣言型MouseOverStyleに依存するより自由に、必要な数だけスタイル・レイヤーを追加できます。

Opacity プロパティ

DirectX のみ

Opacityは、背景との相互作用の観点から、コントロールの外観を示します。

省略値は100：完全に不透明です。背景はまったく見えません。

値が減少するにつれ、コントロールを通り抜けて背景がより見えるようになります。値が0の時、コントロールは完全に透過的になり、背景のみが見えます。

注：不透過度が0のコントロールはユーザーには見えませんが、通常通り機能し続け、表示され、有効になっています。

Popup プロパティ

DirectX のみ

Popupは、通常の右クリックのポップアップ・メニュー (Prim_pmnu) の代わりに、PopupPanel (#prim_ppnl) インスタンスの使用を可能にします。これにより、開発者は、ポップアップのコンテンツとフォーマットを完全に制御できるようになります。

```
Define_Com Class(#prim_trvw) Name(#Tree) Popup(#TreePopupPanel)
Define_Com Class(#Prim_ppnl) Name(#TreePopupPanel)
Content(#MyTreePopupPanel)
```

Rotation プロパティ

DirectX のみ

Rotationは、コントロールが与えられた原点の周囲を、RotationOriginLeftとRotationOriginTopプロパティで指定されたように回転できるようにします。

回転は度数で表され、0から359までの値域を持ちます。

回転は、スケールや傾斜のように視覚効果で、コントロールの根本的なサイズおよび位置は変更しません。左、上、高さおよび幅プロパティは、その値を保持します。回転されたコントロールは、親コントロールの範囲に入るように縮められます。

RotationOriginLeft プロパティ

DirectX のみ

RotationOriginTopとともに、Rotationプロパティが適用された場合、コントロールがその周囲を回転する仮想上の点の座標を指定します。

ScaleOriginLeftはパーセントで表され、省略値は50です。

RotationOriginTop プロパティ

DirectX のみ

RotationOriginTopとともに、Rotationプロパティが適用された場合、コントロールがその周囲を回転する仮想上の点の座標を指定します。

RotationOriginTopはパーセントで表され、省略値は50です。

ScaleHeight プロパティ

DirectX のみ

ScaleWidthとともに、ScaleOriginLeftとScaleOriginTopプロパティで指定される仮想上の点に基づいてコントロールの表示サイズを指定します。

ScaleHeightはパーセントで表されます。省略値は100です。

回転や傾斜のように、スケールは視覚効果で、コントロールの根本的なサイズおよび位置は変更しません。左、上、高さおよび幅プロパティは、その値を保持します。スケールされたコントロールは、親コントロールの範囲に入るように縮められます。

ScaleWidth プロパティ

DirectX のみ

ScaleHeightとともに、ScaleOriginLeftとScaleOriginTopプロパティで指定される仮想上の点に基づいてコントロールの表示サイズを指定します。

ScaleWidthはパーセントで表されます。省略値は100です。

回転や傾斜のように、スケールは視覚効果で、コントロールの根本的なサイズや位置は変更しません。左、上、高さおよび幅プロパティは、その値を保持します。スケールされたコントロールは、親コントロールの範囲に入るように縮められます。

ScaleOriginLeft プロパティ

DirectX のみ

ScaleOriginTopとともに、ScaleWidthとScaleHeightプロパティが適用される際、コントロールがそこからスケールを開始する仮想上の点の座標を指定します。

ScaleOriginLeftはパーセントで表され、省略値は50です。

ScaleOriginTop プロパティ

DirectX のみ

ScaleOriginLeftとともに、ScaleWidthとScaleHeightプロパティが適用される際、コントロールがそこからスケールを開始する仮想上の点の座標を指定します。

ScaleOriginLeftはパーセントで表され、省略値は50です。

SkewLeft プロパティ

DirectX のみ

SkewTopとともに、SkewOriginLeftとSkewOriginTopプロパティで指定される仮想上の点の周囲を回転したかのようにコントロールの外観を定義します。

SkewLeftは、度数で表されます。省略値は0です。

回転やスケールのように、傾斜は視覚効果で、コントロールの根本的なサイズや位置は変更しません。左、上、高さおよび幅プロパティは、その値を保持します。傾斜されたコントロールは、親コントロールの範囲に入るように縮められます。

SkewTop プロパティ

DirectX のみ

SkewLeftとともに、SkewOriginLeftとSkewOriginTopプロパティで指定される仮想上の点の周囲を回転したかのようにコントロールの外観を定義します。

SkewTopは、度数で表されます。省略値は0です。

回転やスケールのように、傾斜は視覚効果で、コントロールの根本的なサイズや位置は変更しません。左、上、高さおよび幅プロパティは、その値を保持します。傾斜されたコントロールは、親コントロールの範囲に入るように縮められます。

SkewOriginLeft プロパティ

DirectX のみ

SkewOriginTopとともに、SkewLeftとSkewTopプロパティを使用して、コントロールがその周囲で傾けられる仮想上の点を指定します。

SkewOriginLeftはパーセントで表されます。省略値は50です。

SkewOriginTop プロパティ

DirectX のみ

SkewOriginLeftとともに、SkewLeftとSkewTopプロパティを使用して、コントロールがその周囲で傾けられる仮想上の点を指定します。

SkewOriginTopはパーセントで表されます。省略値は50です。

Style プロパティ

DirectX のみ

コントロールに適用されるStyle (#Prim_vs.Style) を示します。Styleプロパティが設定されると、コントロールに以前適用されたすべてのスタイルが削除されます。

親コントロールからのスタイルは、isualStyleOfParentプロパティの値によって継承されます。

Styles プロパティ

DirectX のみ

コントロールに適用されるスタイル (#Prim_vs.Style) のコレクションです。

Stylesは、より複雑なプログラム上の外観の変更のコード化を可能にします。開発者は、1つの宣言型Styleプロパティに依存するより自由に、必要な数だけスタイル・レイヤーを追加できます。

Styles コレクション

DirectX のみ

コントロールに適用されるスタイル (#Prim_Vcol<#Prim_vs.Style>) のコレクションです。

Styles コレクションは、実行時、コントロールへの複数のスタイルの追加や、コントロールからの複数のスタイルの削除を可能にします。

Style

Stylesコレクションに追加されるスタイル (#Prim_VS.Style) への参照です。

コレクションに追加されるスタイルがすでにコレクションに存在する場合、Add要求は無視されます。

Add メソッド

Stylesコレクションにスタイルを追加します。

コレクションに追加されるスタイルがすでにコレクションに存在する場合、Add要求は無視されます。

Remove メソッド

Stylesコレクションからスタイルを削除します。

コレクションから削除されるスタイルがコレクションに存在しない場合、Remove要求は無視されます。

RemoveAll メソッド

Stylesコレクションからすべてのスタイルを削除します。

MouseOverPart プロパティ

DirectX のみ

マウスがコントロールまたは子コントロールのすぐ上にあるかどうかを決定します。実際には、このプロパティは、マウスがコントロールの物理的範囲の中にあるかどうかを指定します。

コントロールの外観を変更するためにMouseEnter/Leaveコードが実行されているが、他のファクタもコントロールの状態に影響を与えている場合、MouseOverPartが便利です。

True

マウスは、コントロールまたはその子コントロールの内の一つのすぐ上にあります。

False

マウスは、コントロールのすぐ上にもそのすべての子コントロールのすぐ上にもありません。

PrivateStyle プロパティ

DirectX のみ

以下のコントロールにのみ適用されるスタイル (#Prim_vs.Style) を示します。

通常スタイルは、VisualStyleOfParent プロパティにより子コントロールに継承されます。PrivateStyleは、子コントロールの外観に影響を与えずに複合コントロールが独自のスタイルを持つことを可能にします。

例えば、グループ・ボックスに太字のテキストが必要な場合、通常太字を複合ディレクトリに適用すると標題とコンテンツの両方が太字になります。これをプライベート・スタイルとして指定することにより、グループ・ボックスは、スタイル継承チェーンの外へ出ることができま

PrivateStyles プロパティ

DirectX のみ

このコントロールにのみ適用されるスタイル (#Prim_vs.Style) のコレクションです。

通常、スタイルは、VisualStyleOfParent プロパティにより子コントロールに継承されます。PrivateStyle は、子コントロールの外観に影響を与えずに複合コントロールが独自のスタイルを持つことを可能にします。

実行時、Styles コレクションは、コントロールへの複数のスタイルの追加やコントロールからの複数のスタイルの削除を可能にします。

Transition メソッド

DirectX のみ

Transitionは、あるコンポーネントから別のコンポーネントへの変更をアニメートするメソッドです。

ToおよびFromパラメータ内で指定されるコントロールは、それぞれ Visible (True) から Visible (False) に切り替えられます。アニメーションのスタイルは、TransitionTypeプロパティに依存します。

一番よい結果を得るために、2つのコントロールは同じ物理的範囲を占めなければなりません。

すべてのアニメーションのように別のスレッドで実行され、ほかの処理が継続できます。

メソッドが再度呼び出された時にアニメーションがまだ実行中の場合、アニメーションはリセットされ、再度開始されます。

To

コントロールへの参照です (#Prim_Ctrl)。

コントロールは、複合コントロールに属する必要があります。アニメーションが終了すると、コントロールは Visible (True) になります。

From

コントロールへの参照です (#Prim_Ctrl)。

コントロールは、複合コントロールに属する必要があります。アニメーションが終了すると、コントロールは Visible (False) になります。

TransitionType

遷移タイプは多数あります。省略値はfadeです。

Delay

アニメーションの処理を遅延するミリ秒数です。省略値は0です。

Duration

アニメーションの完了にかかるミリ秒数です。省略値は250です。

RenderStyle

RDNR X_RUN引数とともに、フォーム (#Prim_FORM) とパネル (#Prim_PANL) のRenderStyleプロパティは、アプリケーションの一部または全体がどのように表示されるかを決定します。

RDNR実行時引数には、可能な値が3つあります：W, M, X。それぞれ、Win32、Win32とDirectX, そしてDirectXを指します。DirectX内で表示するすべてのアプリケーションの実行時引数は、MまたはXでなければなりません。

値がWの場合、指定されるプロパティの値にかかわらず、DirectXは使われません。

フォーム (#Prim_Form) には、可能な値が3つあります。

ApplicationRDNR実行時引数値が使われます。これは省略値です。

DirectX

RDNR実行時引数がMまたはXの場合、フォームはDirectXを使います。

Win32

実行時の設定に関わらず、フォームは Win32 になります。

パネル (#Prim_Panl) には、可能な値が2つあります。

パネルを特別にWin32にするオプションはありません。パネルがDirectXになると、すべての子パネルはDirectXでなければなりません。

DirectX

RDNR実行時引数がMまたはXの場合、パネルはDirectXを使用します。

Parent

パネルは、親のRenderStyleに応じて表示されます。これは省略値です。

フォームのアプリケーションとパネルの親の省略値は、実行時の設定を1つ変更することにより、DirectXのアプリケーション全体での実装を可能にします。

新しいDirectXコントロールやスタイルを組み込むため、数枚のパネルなど部分的にDirectXのみを有効にしたい場合、実行時の設定をMに変更してDirectXが許可されるようにし、その後必要に応じてフォームやパネルをDirectXに変更します。

スタイル (Prim_Vs.Style)

スタイルは、コントロールの外観を変更するのに使用されます。ユーザーは、スタイルを使用してテキストの色、背景色、フォント、枠線その他のビジュアル機能を定義することができます。

相互に排他的なビジュアル・スタイルとは違い、スタイル上で指定された機能以外に影響を与えずに、複数のスタイルをコントロールに追加することができます。スタイルの同じ機能が、2つの異なるスタイル・インスタンスを使って適用されると、最後に適用されたインスタンスが優先されます。

子コントロールには、FacenameやTextColorのような前景機能のみが採用されます。

BackgroundBrush プロパティ

DirectX のみ

コントロールの背景に適用される ブラシ (#Prim_vs.Brush) インスタンスへの参照です。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

BackgroundBrushとNormBackColorの両方が指定された場合、BackgroundBrushが優先されます。

BackgroundBrushプロパティは、以下の例のように使えます。

```
Define_Com Class(#PRIM_VS.Style) Name(#BackGround)
Backgroundbrush(#Backgroundbrush)
Define_Com Class(#PRIM_VS.LinearBrush) Name(#Backgroundbrush)
Colors(#BackgroundbrushColors)
Define_Com Class(#Prim_Vs.BrushColors) Name(#BackgroundbrushColors)
Define_Com Class(#PRIM_VS.BrushColor)
Name(#BackgroundbrushColor1) Color(Silver)
Parent(#BackgroundbrushColors)
Define_Com Class(#PRIM_VS.BrushColor)
Name(#BackgroundbrushColor1) At(100) Color(White)
Parent(#BackgroundbrushColors)
```

BorderBottom プロパティ

DirectX のみ

BorderBottomは、BorderTop、BorderLeft、BorderRightとともにコントロールの周囲に表示される枠線の太さを指定します。

枠線は、適用されるとコントロールの一部を消費します。これは、MouseOverの上および左の枠線にあるスタイルを適用する際、特に関係します。上および左の枠線は、効率良く0,0座標を移動させ、その結果、子コントロールの画面上の位置を動かします。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

BorderBrush プロパティ

DirectX のみ

コントロールの枠線に適用されるブラシ (#Prim_vs.Brush) インスタンスへの参照です。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

BorderLeft プロパティ

DirectX のみ

BorderLeftは、BorderTop、BorderBottom、BorderRightとともにコントロールの周囲に表示される枠線の太さを指定します。

枠線は、適用されるとコントロールの一部を消費します。これは、MouseOverの上および左の枠線にあるスタイルを適用する際、特に関係します。上および左の枠線は、効率良く0,0座標を移動させ、その結果、子コントロールの画面上の位置を動かします。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

BorderRight プロパティ

DirectX のみ

BorderRightは、BorderTop、BorderLeft、BorderBottomとともにコントロールの周囲に表示される枠線の太さを指定します。

枠線は、適用されるとコントロールの一部を消費します。これは、MouseOverの上および左の枠線にあるスタイルを適用する際、特に関係します。上および左の枠線は、効率良く0,0座標を移動させ、その結果、子コントロールの画面上の位置を動かします。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

BorderTop プロパティ

DirectX のみ

BorderTopは、BorderBottom、BorderLeft、BrderRightとともにコントロールの周囲に表示される枠線の太さを指定します。

枠線は、適用されるとコントロールの一部を消費します。これは、MouseOverの上および左の枠線に、あるスタイルを適用する際、特に関係します。上および左の枠線は、効率良く0,0座標を移動させ、その結果、子コントロールの画面上の位置を動かします。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

CornerBottomLeft プロパティ

DirectX のみ

CornerBottomLeftは、CornerTopLeft、CornerTopRight、CornerBottomRightとともにコントロールの角の半径を定義します。省略値では、すべての角は直角でその半径は0です。

0,0座標の位置を変更する枠線とは異なり、角は視覚効果です。適用されると、コントロールの背景は縮小され、コントロールは丸く表示されます。しかし、前景機能は縮小されず、テキストは角に向かって広がります。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

CornerBottomRight プロパティ

DirectX のみ

CornerBottomRightは、CornerTopLeft、CornerTopRight、CornerBottomLeftとともにコントロールの角の半径を定義します。省略値では、すべての角は直角でその半径は0です。

0,0座標の位置を変更する枠線とは異なり、角は視覚効果です。適用されると、コントロールの背景は縮小され、コントロールは丸く表示されます。しかし、前景機能は縮小されず、テキストは角に向かって広がります。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

CrnerTopLeft プロパティ

DirectX のみ

CornerTopLeftは、CornerBottomLeft、CornerBottomRight、CornerTopRightとともにコントロールの角の半径を定義します。省略値では、すべての角は直角でその半径は0です。

0,0座標の位置を変更する枠線とは異なり、角は視覚効果です。適用されると、コントロールの背景は縮小され、コントロールは丸く表示されます。しかし、前景機能は縮小されず、テキストは角に向かって広がります。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

CornerTopRight プロパティ

DirectX のみ

CornerTopRightは、CornerBottomLeft、CornerBottomRight、CornerTopLeftとともにコントロールの角の半径を定義します。省略値では、すべての角は直角でその半径は0です。

0,0座標の位置を変更する枠線とは異なり、角は視覚効果です。適用されると、コントロールの背景は縮小され、コントロールは丸く表示されます。しかし、前景機能は縮小されず、テキストは角に向かって広がります。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

Effect プロパティ

DirectX のみ

効果 (#PRIM_VS.Effect) インスタンスへの参照です。

Effectは、スタイル上で指定される特徴を、ぼかしたり影をつけたりして増補し、3D効果をよりよく表現するために使用されます。

スタイルの背景機能は、子コントロールには継承されません。したがって、フォームに適用された背景は、そのまま子コントロールに繰り返されることはありません。

ForegroundBrush プロパティ

DirectX のみ

コントロールの前景に適用されるブラシ (#Prim_vs.Brush) インスタンスへの参照です。

スタイルの前景機能は、子コントロールに継承されます。したがって、スタイルによってフォームに適用されたフォントが、すべての子コントロールで一貫して使用されます。

ForegroundBrushとTextColorの両方が指定された場合、ForegroundBrushが優先されます。

MaskBrush

選択的に透明性を適用するためにコントロールに適用されるブラシ (#Prim_vs.Brush) インスタンスへの参照です。したがって、コントロール、特にイメージがフェードアウトされ、背景に溶け込みます。

すべての色を透過的な色にグラデーションさせる線状ブラシの場合、コントロールは、次第に透過的になります。ビジュアル ブラシを使用する場合、ビジュアル ブラシで使用されるイメージの透過度に基いて、透明性が適用されます。

ブラシ

Solid、Linear、Radial、Image すべてのBrushクラスのベース・クラスです。

ブラシはスタイル (#Prim_VS.Style) で使用され、前景と背景の外観を定義します。

詳細は、各Brushクラスを参照してください。

BrushColors

BrushColorsは、BrushColor (Prim_Vs.BrushColor) インスタンスのコレクターです。この抽象化コレクターは、同じ一連の色が複数のブラシで使用できるようにします。

BrushColorsは、BrushColorインスタンスを使用して、ある色から別の色への遷移を記述します。一般的に、上のシルバーから下の白への単純なグラデーションの遷移の場合もありますが、色の数と遷移の性質は、使用されるブラシの特定のクラスの構成およびカラー・インスタンスの定義に依存します。

以下の例では、青から赤に色が変わります。2番目の色 (#Color2) に "At(100)" を使用することで表されます。

```
Define_Com Class(#Prim_Vs.BrushColors) Name(#Colors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color1) Color(0:0:255)
Parent(#Colors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color2) At(100)
Color(255:0:0) Parent(#Colors)
```

遷移は一樣なので、青のチャンネルは255から0へ減少し、赤のチャンネルは0から255に増加します。その結果、中間点において、色は128:128、紫になります。

BrushColor

色をBrushColors (#Prim_VS.BrushColors) インスタンスの一部として定義するのに使用されます。

At プロパティ

Atは、Colorプロパティで指定された色としてグラデーション色が表示される、仮想線上のポイントを決定します。

Atはパーセントで表されます。

```
Define_Com Class(#Prim_Vs.BrushColors) Name(#Colors)
```

```
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color1) Color(Silver)
```

```
Parent(#Colors)
```

```
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color2) At(100)
```

```
Color(White) Parent(#Colors)
```

Color プロパティ

Atプロパティで決定されるポイントにおけるブラシの色を示します。色は赤の255:0:0のように任意の有効なRGB値、または赤、青、黄または透過的な色など、一連の定義済みの指定色のいずれかです。

MaskBrushにブラシ色が使用されている場合、指定された色にかかわらず、Colorプロパティは透過的または不透過的として評価されます。

以下は、白から透過的な色への変更の例です。遷移が起こると、背景は、徐々により見えるようになります。

```
Define_Com Class(#Prim_Vs.BrushColors) Name(#Colors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color1) Color(White)
Parent(#Colors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#Color2) At(100)
Color(Transparent) Parent(#Colors)
```

Parent プロパティ

各カラー・インスタンスの収集に使用される BrushColors (#Prim_Vs.BrushColors) への参照です。

グラデーション ブラシ

1色以上の色を使ったグラデーション・カラーを定義するのに使用されるブラシです。

Colors プロパティ

BrushColors (#Prim_VS.BrushColors) インスタンスへの参照です。
ブラシに使用される色を定義します。

Opacity プロパティ

Opacityは、背景との相互作用の観点から、グラデーションの外観を示します。

省略値は100：完全に不透明です。グラデーションの背後には何も見えません。

値が減少するにつれ、コントロールを通り抜けて、背景がより見えるようになります。値が0の場合、コントロールは完全に透過的になり、背景のみが見えます。

コントロール上で指定されると、不透過度は、コントロール全体、つまり前景と背景の両方に影響を与えます。ブラシ上で指定されると、不透過度は前景と背景のどちらにも使用可能で、半透過的な背景にも使用できますが、テキストは完全に不透過的です。

Spread プロパティ

ブラシ色の範囲、または特定のブラシの論理的始まりと終わりを越えたブラシの外観の定義に使用されます。

Pad

ブラシ色の始まりの前または終わりの後のスペースは、それぞれ最初または最後の色として指定された色で塗られます。

したがって、以下の例では、赤のグラデーションは全体の20%の位置まで始まらないので、最初の20パーセントは赤に塗られます。同様に、青の後の20%が青に塗られます。

```
Define_Com Class(#Prim_Vs.LinearBrush) Name(#LinearBrush)
Colors(#LinearBrushColors)
Define_Com Class(#Prim_Vs.BrushColors) Name(#LinearBrushColors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#LinearBrushColor1)
At(20) Color(Red) Parent(#LinearBrushColors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#LinearBrushColor2)
At(80) Color(Blue) Parent(#LinearBrushColors)
```

Reflect

ブラシ色の始まりの前または終わりの後のスペースは、ブラシのリフレクションで塗られます。

Repeat

ブラシ色の始まりまたは終わりの後のスペースは、ブラシの繰り返しで塗られます。

線状ブラシ

線状ブラシは、Colors (#Prim_vs.BrushColors) プロパティで指定された2色の間で遷移するグラデーション色の作成に使用されます。

線状ブラシには、グラデーションがたどるパスを定義する仮想線を作成するために始まりと終わりの座標があり、垂直、水平または斜めのグラデーションを可能にします。

EndLeft プロパティ

EndTop、StartLeft、StartTop プロパティとともに使用され、ブラシ評価のパスと範囲を定義します。EndLeftはパーセントで表されます。

StartLeft(0) StartTop(0) EndLeft(100) EndTop(100) と定義されるブラシは、左上から右下へ対角線上に移動するように表示されます。

StartLeft(0) StartTop(0) EndLeft(0) EndTop(100) と定義されるブラシは、上から下へ垂直に移動するように表示されます。

StartLeft(0) StartTop(0) EndLeft(0) EndTop(50) と定義されるブラシは、コントロールの上から真ん中まで垂直に移動するように表示されます。

Spread プロパティは、始まりと終わりの座標の範囲を超えた外観を定義します。

EndTop プロパティ

EndLeft、StartLeft、StartTop プロパティとともに使用され、ブラシのパスと範囲を定義します。EndTop はパーセントで表されます。

StartLeft(0) StartTop(0) EndLeft(100) EndTop(100) と定義されるブラシは、左上から右下へ対角線上に移動するように表示されます。

StartLeft(0) StartTop(0) EndLeft(0) EndTop(100) と定義されるブラシは、上から下へ垂直に移動するように表示されます。

StartLeft(0) StartTop(0) EndLeft(0) EndTop(50) と定義されるブラシは、コントロールの上から半分下へ垂直に移動するように表示されます。

Spread プロパティは、始まりと終わりの座標の範囲を超えた外観を定義します。

StartLeft プロパティ

EndTop、EndLeft、StartTop プロパティとともに使用され、ブラシのパスと範囲を定義します。StartLeftはパーセントで表されます。

StartLeft(0) StartTop(0) EndLeft(100) EndTop(100) と定義されるブラシは、左上から右下へ対角線上に移動するように表示されます。

StartLeft(0) StartTop(0) EndLeft(0) EndTop(100) と定義されるブラシは、上から下へ垂直に移動するように表示されます。

StartLeft(0) StartTop(0) EndLeft(0) EndTop(50) と定義されるブラシは、コントロールの上から半分下へ垂直に移動するように表示されます。

Spread プロパティは、始まりと終わりの座標の範囲を超えた外観を定義します。

StartTop プロパティ

EndTop、EndLeft、StartLeft プロパティとともに使用され、ブラシのパスと範囲を定義します。StartTopはパーセントで表されます。

StartLeft(0) StartTop(0) EndLeft(100) EndTop(100) と定義されるブラシは、左上から右下へ対角線上に移動するように表示されます。

StartLeft(0) StartTop(0) EndLeft(0) EndTop(100) と定義されるブラシは、上から下へ垂直に移動するように表示されます。

StartLeft(0) StartTop(0) EndLeft(0) EndTop(50) と定義されるブラシは、コントロールの上から半分下へ垂直に移動するように表示されます。

Spread プロパティは、始まりと終わりの座標の範囲を超えた外観を定義します。

放射状ブラシ

放射状ブラシは、Colors (#Prim_vs.BrushColors) プロパティ内で指定された2色の間で遷移するグラデーション色の作成に使用されます。

放射状ブラシには、Origin、CenterおよびRadiusの座標があります。放射状ブラシは、線状ブラシ (#Prim_VS.Linear Brush) と同様、CenterからRadiusの範囲に沿って放射状に広がりますが、線状ブラシとは違い、円形パターンを作ります。

OriginとCenter座標が同じなので、ブラシは常にビューのすぐ前にあるように表示されます。しかし、値を変更すると形が変わり、ブラシは傾いて見えます。単純に例えると、懐中電灯の光線を見ているようなものです。放射状ブラシは、3Dの形を2Dで見るようなものと考えられます。Originは懐中電灯の位置、Centerは懐中電灯が何かを照らす時の光の中心、そしてRadiusは幅を表します。原点と中心に十分な差がある場合、光線の三角の側面のみしか見えず、基点を見ることはできません。

以下の例では、0では赤だった単純な放射状ブラシが、中央に赤を、外側に青を生成しながら青に変化します。

```
Define_Com Class(#Prim_Vs.RadialBrush) Name(#RadialBrush)
Colors(#RadialBrushColors)
Define_Com Class(#Prim_Vs.BrushColors) Name(#RadialBrushColors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#RadialBrushColor1)
Color(Red) Parent(#RadialBrushColors)
Define_Com Class(#Prim_Vs.BrushColor) Name(#RadialBrushColor2)
At(100) Color(Blue) Parent(#RadialBrushColors)
```

CenterLeft プロパティ

CenterTopプロパティとともに、光線の基点の出発点と中間点を定義します。ブラシ色はこのポイントから広がります。

CenterLeftはパーセントで表されます。

CenterLeftとCenterTopの省略値は、どちらも50です。

CenterTop プロパティ

CenterLeftプロパティとともに、光線の基点の出発点と中間点を定義します。ブラシ色はこのポイントから広がります。

CenterTopはパーセントで表されます。

CenterLeftとCenterTopの省略値は、どちらも50です。

OriginLeft プロパティ

OriginTopプロパティとともに、ソースの位置を定義します。OriginとCenterが分散すると、ブラシの形は円形から平らな形に変わります。

OriginLeftはパーセントで表されます。

OriginLeftとOriginTopの省略値は、どちらも50です。

OriginTop プロパティ

OriginLeftプロパティとともに、ソースの位置を定義します。OriginとCenterが分散すると、ブラシの形は円形から平らな形に変わります。

OriginTopはパーセントで表されます。

OriginLeftとOriginTopの省略値は、どちらも50です。

RadiusLeft プロパティ

RadiusTopプロパティとともに、ブラシの範囲を定義します。

RadiusLeftとRadiusTopの値が等しいと、中央と原点の値に関係なく、ブラシは円形になります。異なる値を使用すると、ブラシはつぶれて表示されます。

RadiusLeftはパーセントで表されます。

RadiusLeftとRadiusTopの省略値は、どちらも50です。

RadiusTop プロパティ

RadiusLeft プロパティとともに、ブラシの範囲を定義します。

RadiusLeft と RadiusTop の値が等しいと、中央と原点の値に関係なく、ブラシは円形になります。異なる値を使用すると、ブラシはつぶれて表示されます。

RadiusTop はパーセントで表されます。

RadiusLeft と RadiusTop の省略値は、どちらも50です。

単色ブラシ

単色ブラシは、実際は1色の線状ブラシ (#Prim_VS.LinearBrush) です。複数色 (#Prim_VS.BrushColors) または1色 (#Prim_VS.BrushColor) の作成を不要にし、1つのステートメントでの簡単な定義を可能にします。

Color プロパティ

ブラシの色を示します。

色は、赤の255:0:0のように任意の有効なRGB値、または赤、青、黄または透過色のように、一連の定義済みの指定色のいずれかが可能です。

Opacity プロパティ

Opacityは、背景との相互作用の観点から、グラデーションの外観を示します。

省略値は100：完全に不透明です。グラデーションの背後には何も見えません。

値が減少するにつれ、コントロールを通り抜けて背景がより見えるようになります。値が0の場合、コントロールは完全に透過的になり、背景のみが見えます。

コントロール上で指定されると、不透過度は、コントロール全体、つまり前景と背景の両方に影響を与えます。ブラシ上で指定されると、不透過度は前景と背景のどちらにも使用され、半透過的な背景も可能にしますが、テキストは完全に不透過的です。

イメージ ブラシ

イメージ ブラシは、コントロールに色ではなくイメージを適用するのに使用されます。

主として、アプリケーションの透かしや背景として使用されます。

Alignment プロパティ

Alignmentは、使用可能なスペースを埋めない場合のイメージの位置を決定します。反対に、スペースに対してイメージが大きすぎる場合、Alignmentはイメージのどの部分を表示するかを決定します。

可能な値は、TopLeft、TopCenter、TopRight、CenterLeft、Center、CenterRight、BottomLeft、BottomCenterおよびBottomRightです。

Height プロパティ

イメージの高さを、Unitsプロパティに応じて、ピクセル値またはパーセンテージで決定します。タイル表示する場合は、各タイルのサイズを示します。

Image プロパティ

ベーシック LANSА グラフィック・コンポーネント (#PRIM_FLBX) インスタンスへの参照です。

これは、登録されているビットマップ (#Prim_Bmp) やアイコン (#Prim_Icon)、または実行時に作成されたビットマップです。

Left プロパティ

イメージの表示位置のコントロールの左端からの距離を、特定のピクセル数、またはUnitsプロパティで指定されるように、パーセントで表わされたコントロールの幅で指定します。

Opacity プロパティ

Opacityは、背景との相互作用の観点から、グラデーションの外観を示します。

省略値は100：完全に不透明です。イメージの背後には何も見えません。

値が減少するにつれ、コントロールを通り抜けて背景がより見えるようになります。値が0の場合、コントロールは完全に透明になり、背景のみが見えます。

コントロール上で指定されると、不透過度は、コントロール全体、つまり前景と背景の両方に影響を与えます。ブラシ上で指定されると、不透過度は前景と背景のどちらにも使用され、半透過的な背景も可能にしますが、テキストは完全に不透過的です。

Sizing プロパティ

コントロールまたは分割された部分の使用可能なコンテナに収まるよう、どのようにイメージを拡大したりつぶしたりするかを記述します。省略値はNoneです。

None

イメージは、決してサイズ変更されません。

Best Fit

縦横比は保持したまま、使用可能なスペースに収まるようにサイズ変更されます。

Fit Both

イメージは、水平および垂直のスペースにぴったりはまるようにサイズ変更されます。縦横比は保持されません。その結果、イメージは拡大されたように見えることがあります。

Cropped

イメージは、縦横比は保持したまま、使用可能なスペースに収まるようにサイズ変更されます。水平または垂直どちらかがコントロールに収まると、それ以上サイズは変更されません。その結果、いくつかのイメージが表示されなくなることがあります。

Tile プロパティ

イメージが、コントロール内でどのように繰り返されるかを記述します。

省略値はNoneです。他のすべての列挙型の値は、イメージを垂直または水平に繰り返します。

None

イメージは、決して分割されません。

Tile

イメージは、垂直にも水平にも繰り返されます。

MirrorHorizontal

イメージは分割され、垂直軸方向に反転されます。

MirrorVertical

イメージは分割され、水平軸方向に反転されます。

MirrorBoth

イメージは分割され、水平および垂直軸方向に反転されます。

Top プロパティ

イメージの表示位置のコントロールの上端からの距離を、特定のピクセル数、またはUnitsプロパティで指定されるように、パーセントで表わされたコントロールの幅で指定します。

Units プロパティ

イメージのサイズと位置がどのように計測されるかを示します。省略値はPercentageです。

Percentage

すべての高さ、幅、左および上の値は、コントロールのパーセンテージで計測されます。

Pixels

すべての高さ、幅、左および上の値は、特定のピクセル数で計測されます。

Width プロパティ

イメージの幅を、Unitsプロパティに応じて、ピクセル値またはパーセンテージで決定します。タイル表示する場合は、各タイルのサイズを示します。

ビジュアル ブラシ

イメージ ブラシは、コントロールに色ではなくイメージを適用するのに使用されます。静的イメージを使用するイメージ ブラシとは違い、ビジュアル ブラシは別のコントロールを使用します。

通常、ドラッグ・アンド・ドロップ中のイメージとして使用されます。

Control プロパティ

Control (#Prim_CTRL) インスタンスへの参照です。

見えるイメージは実際のコントロールそのものなので、コントロールは実現化される必要があります。

イメージはソース・イメージのスナップショットやコピーではなく、イメージへの参照です。これは、コントロールへの変更にイメージが対応することを意味します。

効果

DirectX のみ

効果は、スタイル上で指定される特徴を増強するために使用されます。効果には、以下の2つのタイプがあります。ぼかし (#Prim_VS.Blur) または、ドロップシャドウ (#Prim_VS.DropShadow) です。

ぼかし効果

その名の通り、ぼかし効果は、コントロールをぼかすために使用されます。

KernelType プロパティ

ぼかしには、以下の2つのタイプがあります。

Gaussian

より滑らかで質の高いぼかしになります。しかし、実行するには、より多くの処理が必要です。

Box

より単純で質の低いぼかしになります。多くの処理は必要ありません。

Radius プロパティ

ぼかしの程度を決定します。

ドロップシャドウ効果

ドロップシャドウは、コントロールが、親コントロールの上に浮いているように見せるために使用されます。Windows フォームのAeroテーマのタイトルバーに見られる光彩のような効果を、主としてテキストの背後に作成するのにも使用されます。

スケールや回転のように、ドロップシャドウは視覚効果で、コントロールの根本的なサイズは変更しません。

ドロップシャドウ付きのコントロールは、親コントロールの範囲に入るように縮められます。

BlurRadius プロパティ

シャドウに適用されるぼかしの量を示します。

値がゼロに近いほど、シャドウのぼかしは少なくなります。

Color プロパティ

ドロップシャドウの色を示します。

省略値はシルバーです。

Direction プロパティ

ShadowDepthプロパティとともに、ソース・コントロールに関するドロップシャドウの位置を示します。

Directionは、度数で計測されます。

値が0というのは、9時の位置に影を落とす光源があるのと同じです。

Opacity プロパティ

Opacityは、背景との相互作用の観点から、影の外観を示します。省略値は100：完全に不透明です。影の背後には何も見えません。値が減少するにつれ、影を通りぬけて、背景がより見えるようになります。値が0の場合、影は完全に透明になり、背景のみが見えます。

ShadowDepth プロパティ

Direction プロパティとともに、ソース・コントロールに関わるドロップシャドウの位置を示します。

値が0の場合、影はコントロールのすぐ後ろになります。

ユーザー定義コントロール

タイル (Prim_Tile)

DirectX のみ

タイルは、ユーザー定義コントロールと言われる基本コントロールに関連する、一連のリストの一つです。

タイルは、各デザイン・パネルを、フロー・レイアウト・マネージャーに相当するグリッドにまとめます。

ユーザー定義コントロール (以下UDC) は、通常のLANSAリスト・コマンド、ADD_ENTRY, UPD_ENTRYを使って操作できます。リストにエントリーが追加されると、デザインのインスタンスが作成され、フィールドが受け渡され、対応するリスト・アイテムが作成されます。UDCは、リスト内でのアイテムの位置、選択、フォーカス、展開/折りたたみなどを制御し、UDCのタイプ固有のインターフェース上で発行された、一連の定義済みのメソッドを通じて、各アイテムのデザインと通信します。Prim_TileのUDCは、#Prim_Tile.iTileDesignです。

すべてのUDCはパラメータ化されたタイプとして使われ、各エントリーが追加されるたびに作成されるデザインのクラスを定義します。これはDEFINE_COM上で以下のように指定されます。

```
Define_Com Class(#Prim_Tile<#MyTileDesign>) Name(#Tile)
```

ツリー・ビュー (#Prim_TRVW) やリスト・ビュー (#Prim_LTVW) の単純なアイテム作成のオーバーヘッドと比較した再利用可能パーツ・インスタンス作成のオーバーヘッドを考えると、UDCは、数千単位のアイテムのために作られてはいません。大量のシナリオには、他のテクニックを推奨します。

Items プロパティ

タイトルに現在入っているすべてのアイテムのコレクションです。

SelectionMode プロパティ

タイトルが1つまたは複数の選択を許可するかどうかを定義します。

Single

一つのアイテムのみ選択できます。

Multiple

複数のアイテムが選択できます。

タイル・イベント・アイテム

DirectX のみ

Tileアイテム (#Prim_Tile.TileItem) のインスタンスへの参照です。
イベント上で提供されるアイテムは、現在処理中のアイテムへの参照です。

Add メソッド

通常のLANSAリスト・コマンドを使うのとはちがい、Addメソッドは異なるデザインのクラスの追加を可能にします。

Addメソッドを使う場合、*ListFields機能を使ってフィールド値をデザインに引き渡すことはできません。ユーザーはプロセスを制御しており、プログラムを通じ、なんらかのデータを新しいデザイン・インスタンスに引き渡す必要があります。

異なるタイプのデザインを追加できるので、複雑なUI要件の処理をはるかに簡単にします。

Result パラメータ

Addメソッドで作成されたアイテム (#Prim_Tile.TileItem) への参照です。

DesignType パラメータ

作成されるデザイン・インスタンスのクラスです。

指定されるクラスは、タイルのDEFINE_COM上でパラメータ化されたタイプとして指定されたクラスを継承しなければなりません。

DeleteAll メソッド

その名の通り、このメソッドはリストのすべてのアイテムを削除します。

これは `Clr_list` コマンドを使うのと同じです。

FindItem メソッド

FindItemは、各アイテムの デザイン・インターフェース上のOnFindメソッドを呼び出します。

各タイル・アイテムは、テストされる値を受け取り、ブール値の結果を返しなが、順に呼び出されます。正数の結果が返されると、Findは終了します。値を返したアイテムから再開することができます。

```
Mthroutine Name(Search)
```

```
Define_Com Class(#Prim_Tile.TileItem) Name(#FoundItem)
```

```
Reference(*dynamic)
```

```
Begin_Loop
```

```
#FoundItem <= #Tree.FindItem( #Search #FoundItem )
```

```
Leave If(#FoundItem *Is *null)
```

```
End_Loop
```

```
Endroutine
```

Result パラメータ

正数の結果を返す最初のアイテム (#Prim_Tile.TileItem) への参照です。

Key パラメータ

検索される値またはオブジェクトです。

Keyは、どんなタイプの値またはオブジェクトも含むことができるバリエーションです。

Item パラメータ

その後から検索を開始するアイテム (#Prim_Tile.TileItem) への参照です。

StartItemが指定されないと、検索は最初から開始されます。

iTileDesign インターフェース

iTileDesignは、タイル・コントロール (#Prim_Tile) のデザインとして使われる再利用可能パーツにより実行されなければなりません。インターフェースには、それを通じてタイルがデザイン・インスタンスと通信し、インスタンスにSelection, Focusなどの状態の変更を知らせることができる一連のメソッドが含まれます。

以下のソースは、Prim_tileの適切なデザインを実行している再利用可能パーツのBegin_comと、ADD_ENTRYを使ってアイテムを作成する場合にデザインが受け取るフィールドを定義する*Listfields機能を示します。

```
Begin_Com Role(*EXTENDS #Prim_Panl3 *implements  
#Prim_Tile.iTileDesign *ListFields #ListFields)
```

* Fields received as on Add_entry

```
Group_By Name(#ListFields) Fields(#Empno #Surname #Givenname  
#Deptment #Section #Deptdesc #Secdesc)
```

OnAdd メソッド

OnAddは、デザイン・インスタンスがADD_ENTRY経由で、またはADDメソッドを使ってリストに最初に追加されると、実行されます。

ADD_ENTRY使用時、Begin_Com Roleパラメータの*ListFields機能に指定されるフィールドは、このメソッドの実行の前に追加されます。

タイル・アイテム

デザインが追加された場合に作成される、対応するタイル・アイテム (#Prim_Tile.TileItem) への参照です。

OnDelete メソッド

OnDeleteは、対応するアイテムが消去されるプロセスにある時に実行されます。

デザイン・インスタンスは、デザインへの参照が他にないと仮定し、削除の一部として破棄されます。

タイル・アイテム

対応する、削除されるタイル・アイテム (#Prim_Tile.TileItem) への参照です。

OnDisplaying メソッド

OnDisplayingは、デザインがユーザー定義コントロールの表示部分に入ろうとする時に実行されます。

これにより、ユーザーはデザインを作成時に処理するのではなく、必要になるまで遅らせることができます。UDCに多くのアイテムがある場合や、処理が比較的長く続く場合に便利です。

タイル・アイテム

これから表示されるデザインに対応するタイル・アイテム
(#Prim_Tile.TileItem) への参照です。

OnFind メソッド

OnFindは、ユーザーがデザインした親コントロール上でFindメソッドが使われる場合に実行されます。

各アイテムは、テストされる値を受け取り、ブール値の結果を返しなが
ら順に呼び出されます。正数の結果が返されると、Findは終了します。
値を返したアイテムから再開することができます。

```
Mthroutine Name(OnFind) Options(*Redefine)
```

```
#Result := #Surname.Contains( #Key )
```

```
Endroutine
```

Result パラメータ

Resultを設定し、Findが成功したかどうかを示します。

Key パラメータ

検索される値またはオブジェクトです。

Keyは、どんなタイプの値またはオブジェクトも含むことができるバリエーションです。

TileItem パラメータ

デザインに対応するタイル・アイテム (#Prim_Tile.TileItem) への参照です。

OnItemGotFocus メソッド

デザイン・インスタンスがフォーカス・インスタンスになる場合に実行されます。

リストやツリーと同様、あるアイテムは、ユーザーがデザインのカーソルをクリックするか、デザインの境界内にカーソルを置くとフォーカスになります。

アイテムがフォーカスを取得したか失った場合、自動的なビジュアル・フィードバックはありません。スタイルの追加/削除の責任はユーザーにあります。

TileItem パラメータ

デザインに対応するタイル・アイテム (#Prim_Tile.TileItem) への参照です。

OnItemLostFocus メソッド

デザイン・インスタンスがフォーカスを失おうとしている時に実行されます。

アイテムがフォーカスを取得したか失った場合、自動的なビジュアル・フィードバックはありません。スタイルの追加/削除の責任はユーザーにあります。

TileItem パラメータ

デザインに対応するタイル・アイテム (#Prim_Tile.TileItem) への参照です。

OnItemGotSelection メソッド

デザイン・インスタンスが選択された場合に実行されます。

アイテムは、リストやツリーと同様に、ユーザーがデザインのカーソルをクリックするか、デザインの境界内にカーソルを置くと選択されます。

アイテムが選択を取得したか失った場合、自動的なビジュアル・フィードバックはありません。スタイルの追加/削除の責任はユーザーにあります。

TileItem パラメータ

デザインに対応するタイル・アイテム (`#Prim_Tile.TileItem`) への参照です。

OnItemLostSelection メソッド

デザイン・インスタンスが選択を失おうとしている時に実行されます。アイテムが選択を取得したか失った場合、自動的なビジュアル・フィードバックはありません。スタイルの追加/削除の責任はユーザーにあります。

TileItem パラメータ

デザインに対応するタイル・アイテム (#Prim_Tile.TileItem) への参照です。

OnUpdate メソッド

OnUpdateは、UPD_ENTRYを使って対応するアイテムをアップデートする場合に実行されます。

タイル・アイテム

対応する、アップデートされるタイル・アイテム (#Prim_Tile.TileItem)
への参照です。

Tile プロパティ

親タイル (#Prim_Tile) コントロールへの参照です。

タイル・アイテム (Prim_Tile.TileItem)

タイル・アイテムは、タイル (Prim_Tile) に、ADD_ENTRY や Tile Addメソッドなどのエントリが追加される場合に作成されます。

追加される各アイテムについて、対応するユーザー定義のDesignインスタンスが作成され、コントロールの表示部分を提供します。

Design プロパティ

タイル・デザイン (#Prim_Tile.iTileDesign) への参照です。

エントリーがタイルに追加された時に作成された、対応するDesignインスタンスへのアクセスを提供します。

Tile プロパティ

親Tile (#Prim_Tile) コントロールへの参照です。

カルーセル (Prim_Caro)

DirectX のみ

カルーセルは、ユーザー定義コントロールと言われる基本コントロールに関連する、一連のリストの一つです。

カルーセルは、主にイメージのデザイン・パネルを、線状または楕円状で、1つの中央のデザインパネルのみが完全に見えるシーケンスにまとめます。

UDCは、通常のLANSAリスト・コマンド、ADD_ENTRY, UPD_ENTRY を使って操作できます。リストにエントリーが追加されると、デザインのインスタンスが作成され、フィールドが受け渡され、対応するリスト・アイテムが作成されます。UDCは、リスト内でのアイテムの位置、選択、フォーカス、展開/折りたたみなどを制御し、UDCのタイプ固有のインターフェース上で発行された、一連の定義済みのメソッドを通じて、各アイテムのデザインと通信します。Prim_CaroのUDCは、#Prim_Caro.iCarouselDesignです。

すべてのUDCはパラメータ化されたタイプとして使われ、各エントリーが追加されるたびに作成されるデザインのクラスを定義します。これは、DEFINE_COM上で以下のように指定されます。

```
Define_Com Class(#Prim_Caro<#MyCarouselDesign>) Name(#Carousel)
```

ツリー・ビュー (#Prim_TRVW) やリスト・ビュー (#Prim_LTVW) の単純なアイテム作成のオーバーヘッドと比較した再利用可能パーツ・インスタンス作成のオーバーヘッドを考えると、UDCは、数千単位のアイテムのために作られてはいません。大量のシナリオには、他のテクニックを推奨します。

CarouselStyle プロパティ

カルーセルは、以下の2つの方法で変更できます。

Linear

アイテムは、左から右に等間隔で画面を横断して整理されます。

コントロールは、コントロールに収まる数のアイテムのみ表示します。

Elliptical

デザインは楕円状に整理されます。

楕円は、画面に収まるようにサイズ調整され、すべてのアイテムが表示されます。

Items プロパティ

カルーセルに現在入っているすべてのアイテムのコレクションです。

NavigationStyle プロパティ

Buttons

省略値のナビゲーション・ボタンを表示します。

None

省略値のナビゲーション・ボタンを非表示にします。

Carousel イベント・アイテム

DirectX のみ

カルーセル・アイテム (#Prim_Caro.CarouselItem) のインスタンスへの参照です。

イベントで提供されるアイテムは、現在処理中のアイテムへの参照です。

Add メソッド

通常のLANSAリスト・コマンドを使うのとはちがい、Addメソッドは異なるデザインのクラスの追加を可能にします。

Addメソッドを使う場合、*ListFields機能を使ってフィールド値をデザインに引き渡すことはできません。ユーザーはプロセスを制御しており、プログラムを通じ、なんらかのデータを新しいデザイン・インスタンスに引き渡す必要があります。

異なるタイプのデザインを追加できるので、複雑なUI要件の処理をはるかに簡単にします。

Result パラメータ

Addメソッドで作成されたアイテム (#Prim_Caro.CarouselItem) への参照です。

DesignType パラメータ

作成されるデザイン・インスタンスのクラスです。

指定されるクラスは、カルーセル DEFINE_COM上でパラメータ化されたタイプとして指定されたクラスを継承しなければなりません。

DeleteAll メソッド

その名の通り、このメソッドはリストのすべてのアイテムを削除します。

これは `Clr_list` コマンドを使うのと同じです。

FindItem メソッド

FindItemは、各アイテムの デザイン・インターフェース上のOnFindメソッドを呼び出します。

各アイテムは、テストされる値を受け取り、ブール値の結果を返しなが
ら順に呼び出されます。正数の結果が返されると、Findは終了します。
値を返したアイテムから再開することができます。

```
Mthroutine Name(Search)
Define_Com Class(#Prim_Caro.CarouselItem) Name(#FoundItem)
Reference(*dynamic)
Begin_Loop
#FoundItem <= #Tree.FindItem( #Search #FoundItem )
Leave If(#FoundItem *Is *null)
End_Loop
Endroutine
```

Result パラメータ

正数の結果を返す最初のアイテム (#Prim_Caro.CarouselItem) への参照です。

Key パラメータ

検索される値またはオブジェクトです。

Keyは、どんなタイプの値またはオブジェクトも含むことができるバリエーションです。

Item パラメータ

その後から検索を開始するアイテム (#Prim_Caro.CarouselItem) への参照です。

StartItemが指定されないと、検索は最初から開始されます。

FirstItem メソッド

カルーセルの中で、フォーカスアイテムになる最初のアイテムを設定します。

Animate パラメータ

True

現在のアイテムと最初のアイテムの間にあるアイテムの表示の移動をアニメートします。

False

他のアイテムを表示せずに最初のアイテムにジャンプします。

LastItem メソッド

カルーセルの中で、フォーカスアイテムになる最後のアイテムを設定します。

Animate パラメータ

True

現在のアイテムと最後のアイテムの間にあるいくつかのアイテムの表示の移動をアニメートします。

False

他のアイテムを表示せずに最後のアイテムにジャンプします。

MoveToItem メソッド

Positionパラメータで指定されるアイテムに移動します。

Positionパラメータ

アクティブアイテムになるアイテムを指定します。

Animate パラメータ

True

現在のアイテムとターゲットアイテムの間にあるいくつかのアイテムの表示の移動をアニメートします。

False

他のアイテムを表示せずにターゲットアイテムにジャンプします。

NextItem メソッド

カルーセルの中で、次にフォーカスアイテムになるアイテムを設定します。

Animate パラメータ

True

次のアイテムへの移動をアニメートします。

False

アニメートせずに次のアイテムにジャンプします。

PrevItem メソッド

カルーセルの中で、フォーカスアイテムになる、前のアイテムを設定します。

Animate パラメータ

True

前のアイテムへの移動をアニメートします。

False

アニメートせずに前のアイテムにジャンプします。

iCarouselDesign インターフェース

iCarouselDesignは、カルーセル・コントロール (#Prim_Caro) のデザインとして使われる再利用可能パーツにより実行されなければなりません。インターフェースには、それを通じて、カルーセルがデザイン・インスタンスと通信し、Selection、Focusなどの状態の変更を知らせることができる一連のメソッドが含まれます。

以下のソースは、Prim_Caroの適切なデザインを実行している再利用可能パーツのBegin_comと、ADD_ENTRYを使ってアイテムを作成する場合にデザインが受け取るフィールドを定義する*Listfields機能を示します。

```
Begin_Com Role(*EXTENDS #Prim_Panl *implements  
#Prim_Caro.iCarouselDesign *ListFields #ListFields)
```

* Fields received as on Add_entry

```
Group_By Name(#ListFields) Fields(#Empno #Surname #Givenname  
#Deptment #Section #Deptdesc #Secdesc)
```

CarouselItem パラメータ

デザインが追加された時に作成される、対応するカルーセル・アイテム (#Prim_Tile.CarouselItem) への参照です。

CarouselItem パラメータ

対応する、削除されるカルーセル・アイテム (#Prim_Caro.CarouselItem) への参照です。

CarouselItem パラメータ

これから表示されるデザインに対応するカルーセル・アイテム (#Prim_Caro.CarouselItem) への参照です。

Result パラメータ

Resultを設定し、Findが成功したかどうかを示します。

Key パラメータ

検索される値またはオブジェクトです。

Keyは、どんなタイプの値またはオブジェクトも含むことができるバリエーションです。

CarouselItem パラメータ

デザインに対応するカルーセル・アイテム (#Prim_Caro.CarouselItem) への参照です。

CarouselItem パラメータ

デザインに対応するカルーセル・アイテム (#Prim_Caro.CarouselItem) への参照です。

CarouselItem パラメータ

デザインに対応するカルーセル・アイテム (#Prim_Caro.CarouselItem) への参照です。

CarouselItem パラメータ

対応する、更新されるカルーセル・アイテム (#Prim_Caro.CarouselItem) への参照です。

Carousel Items (Prim_Carousel.iCarouselItems)

カルーセルに現在入っているすべてのアイテムのコレクションです (#Prim_Caro)。

Carousel プロパティ

親カルーセル (#Prim_Caro) コントロールへの参照です。

カルーセル・アイテム (Prim_Carousel.iCarouselItem)

カルーセル・アイテムは、カルーセル (Prim_Caro) に、ADD_ENTRY や Carousel Addメソッドなどのエントリーが追加される場合に作成されま
す。

追加される各アイテムについて、対応するユーザー定義のデザイン・イ
ンスタンスが作成され、コントロールの表示部分を提供します。

ブック (Prim_Book)

DirectX のみ

ブックは、ユーザー定義コントロールと言われる基本コントロールに関連する、一連のリストの一つです。

ブックは、各デザイン・パネルを本のページのようにまとめます。

UDCは、通常のLANSAリスト・コマンド、ADD_ENTRY, UPD_ENTRYを使って操作できます。リストにエントリーが追加されると、デザインのインスタンスが作成され、フィールドが受け渡され、対応するリスト・アイテムが作成されます。UDCは、リスト内でのアイテムの位置、選択、フォーカス、展開/折りたたみなどを制御し、UDCのタイプ固有のインターフェース上で発行された、一連の定義済みのメソッドを通じて、各アイテムのデザインと通信します。Prim_BookのUDCは、#Prim_Book.iBookDesignです。

すべてのUDCはパラメータ化されたタイプとして使われ、各エントリーが追加されるたびに作成されるデザインのクラスを定義します。これは、DEFINE_COM上で以下のように指定されます。

```
Define_Com Class(#Prim_Book<#MyBookDesign>) Name(#Book)
```

ツリー・ビュー (#Prim_TRVW) やリスト・ビュー (#Prim_LTVW) の単純なアイテム作成のオーバーヘッドと比較した再利用可能パーツ・インスタンス作成のオーバーヘッドを考えると、UDCは、数千単位のアイテムのために作られてはいません。大量のシナリオには、他のテクニックを推奨します。

Items プロパティ

ブックに現在入っているすべてのアイテムのコレクションです。

NavigationStyle プロパティ

Buttons

省略値のナビゲーション・ボタンを表示します。

None

省略値のナビゲーション・ボタンを非表示にします。

Book イベント・アイテム

DirectX のみ

ブック・アイテム (#Prim_Book.BookItem) のインスタンスへの参照です。

イベントで提供されるアイテムは、現在処理中のアイテムへの参照です。

Add メソッド

通常のLANSAリスト・コマンドを使うのとはちがい、Addメソッドは異なるデザインのクラスの追加を可能にします。

Addメソッドを使う場合、*ListFields機能を使ってフィールド値をデザインに引き渡すことはできません。ユーザーはプロセスを制御しており、プログラムを通じ、なんらかのデータを新しいデザイン・インスタンスに引き渡す必要があります。

異なるタイプのデザインを追加できるので、複雑なUI要件の処理をはるかに簡単にします。

Result パラメータ

Addメソッドで作成されたアイテム (#Prim_Book.BookItem) への参照です。

DesignType パラメータ

作成されるデザイン・インスタンスのクラスです。

指定されるクラスは、ブックのDEFINE_COMでパラメータ化されたタイプとして指定されたクラスを継承しなければなりません。

DeleteAll メソッド

その名の通り、このメソッドはリストのすべてのアイテムを削除します。

これは `Clr_list` コマンドを使うのと同じです。

FindItem メソッド

FindItemは、各アイテムの デザイン・インターフェース上のOnFindメソッドを呼び出します。

各アイテムは、テストされる値を受け取り、ブール値の結果を返しなが
ら順に呼び出されます。正数の結果が返されると、Findは終了します。
値を返したアイテムから再開することができます。

```
Mthroutine Name(Search)
Define_Com Class(#Prim_Book.BookItem) Name(#FoundItem)
Reference(*dynamic)
Begin_Loop
#FoundItem <= #Tree.FindItem( #Search #FoundItem )
Leave If(#FoundItem *Is *null)
End_Loop
Endroutine
```

Result パラメータ

正数の結果を返す最初のアイテム (#Prim_Book.BookItem) への参照です。

Key パラメータ

検索される値またはオブジェクトです。

Keyは、どんなタイプの値またはオブジェクトも含むことができるバリエーションです。

Item パラメータ

その後から検索を開始するアイテム (#Prim_Book.BookItem) への参照です。

StartItemが指定されないと、検索は最初から開始されます。

FirstItem メソッド

ブックの中で、フォーカスアイテムになる最初のアイテムを設定します。

Animate パラメータ

True

現在のアイテムと最初のアイテムの間にあるアイテムの表示の移動をアニメートします。

False

他のアイテムを表示せずに最初のアイテムにジャンプします。

LastItem メソッド

ブックの中で、フォーカスアイテムになる最後のアイテムを設定します。

Animate パラメータ

True

現在のアイテムと最後のアイテムの間にあるいくつかのアイテムの表示の移動をアニメートします。

False

他のアイテムを表示せずに最後のアイテムにジャンプします。

MoveToItem メソッド

Positionパラメータで指定されるアイテムに移動します。

Position パラメータ

アクティブアイテムになるアイテムを指定します。

Animate パラメータ

True

現在のアイテムとターゲットアイテムの間にあるいくつかのアイテムの表示の移動をアニメートします。

False

他のアイテムを表示せずにターゲットアイテムにジャンプします。

NextItem メソッド

ブックの中で、次にフォーカスアイテムになるアイテムを設定します。

Animate パラメータ

True

次のアイテムへの移動をアニメートします。

False

アニメートせずに次のアイテムにジャンプします。

PrevItem メソッド

ブックの中で、フォーカスアイテムになる、前のアイテムを設定します。

Animate パラメータ

True

前のアイテムへの移動をアニメートします。

False

アニメートせずに前のアイテムにジャンプします。

iBookDesign インターフェース

iBookDesignは、ブック・コントロール (#Prim_Book) のデザインとして使われる再利用可能パーツにより実行されなければなりません。インターフェースには、それを通じて、ブックがデザイン・インスタンスと通信し、Selection、Focusなどの状態の変更を知らせることができる一連のメソッドが含まれます。

以下のソースは、Prim_Bookの適切なデザインを実行している再利用可能パーツのBegin_comと、ADD_ENTRYを使ってアイテムを作成する場合にデザインが受け取るフィールドを定義する*Listfields機能を示します。

```
Begin_Com Role(*EXTENDS #Prim_Panl3 *implements  
#Prim_Book.iBookDesign *ListFields #ListFields)
```

* Fields received as on Add_entry

```
Group_By Name(#ListFields) Fields(#Empno #Surname #Givenname  
#Deptment #Section #Deptdesc #Secdesc)
```

BookItem パラメータ

デザインが追加された場合に作成される、対応するBookItem (#Prim_Tile.TileItem) への参照です。

BookItem パラメータ

対応する、削除されるブック・アイテム (#Prim_Book.BookItem) への参照です。

BookItem パラメータ

これから表示されるデザインに対応するブック・アイテム
(#Prim_Book.BookItem) への参照です。

Result パラメータ

Resultを設定し、Findが成功したかどうかを示します。

Key パラメータ

検索される値またはオブジェクトです。

キーは、どんなタイプの値またはオブジェクトも含むことができるバリアントです。

BookItemパラメータ

デザインに対応するブック・アイテム (#Prim_Book.BookItem) への参照です。

BookItem Parameter

デザインに対応するブック・アイテム (#Prim_Book.BookItem) への参照です。

BookItem パラメータ

デザインに対応するブック・アイテム (`#Prim_Book.BookItem`) への参照です。

BookItem パラメータ

デザインに対応するブック・アイテム (#Prim_Book.BookItem) への参照です。

Book プロパティ

親ブック (#Prim_Book) コントロールへの参照です。

ブック・アイテム (Prim_Book.BookItem)

Book Itemは、ブック (Prim_Book) に、ADD_ENTRY や Book Add メソッドなどのエントリーが追加される場合に作成されます。

追加される各アイテムについて、対応するユーザー定義のDesignインスタンスが作成され、コントロールの表示部分を提供します。

ツリー (Prim_Tree)

DirectX のみ

ツリーは、ユーザー定義コントロールと言われる基本コントロールに関連する、一連のリストの一つです。

ツリーは、各デザインパネルを、ツリー・ビューによく似た階層構造にまとめます。

UDCは、通常のLANSAリスト・コマンド、ADD_ENTRY, UPD_ENTRY を使って操作できます。リストにエントリーが追加されると、デザインのインスタンスが作成され、フィールドが受け渡され、対応するリスト・アイテムが作成されます。UDCは、リスト内でのアイテムの位置、選択、フォーカス、展開/折りたたみなどを制御し、UDCのタイプ固有のインターフェース上で発行された、一連の定義済みのメソッドを通じて、各アイテムのデザインと通信します。Prim_TreeのUDCは、#Prim_Tree.iTreeDesignです。

すべてのUDCはパラメータ化されたタイプとして使われ、各エントリーが追加されるたびに作成されるデザインのクラスを定義します。これは、DEFINE_COM上で以下のように指定されます。

```
Define_Com Class(#Prim_Tre<#MyTreeDesign>) Name(#Tree)
```

ツリー・ビュー (#Prim_TRVW) やリスト・ビュー (#Prim_LTVW) の単純なアイテム作成のオーバーヘッドと比較した再利用可能パーツ・インスタンス作成のオーバーヘッドを考えると、UDCは、数千単位のアイテムのために作られてはいません。大量のシナリオには、他のテクニックを推奨します。

Items プロパティ

ツリーに現在入っているすべてのアイテムのコレクションです。

Selection Style プロパティ

All

位置に関係なくツリーのすべてのアイテムが選択できます。

Single

どんな時でも1つのアイテムのみ選択できます。

SameParent

親が同じアイテムである場合、複数のアイテムが選択できます。

SameLevel

同じ親を持つアイテムを親に持つ場合、複数のアイテムが選択できます。

Treeイベント・アイテム

DirectX のみ

ツリー・アイテム (#Prim_Tree.TreeItem) のインスタンスへの参照です。イベントで提供されるアイテムは、現在処理中のアイテムへの参照です。

ItemCollapsed イベント

あるツリーのアイテムが折りたたまれました。すべての子ノードは、ユーザーには見えなくなります。

これは、ツリー・デザイン・インターフェース (#Prim_Tree.iTreeDesign) のOnItemCollapsedメソッドに相当します。

ItemExpanding イベント

あるツリーのアイテムが展開されています。すべての子ノードが、展開しているアイテムの直下に表示されます。

これは、ツリー・デザイン・インターフェース

(#Prim_Tree.iTreeDesign) のOnItemExpandingメソッドに相当します。

ItemGotParent イベント

ツリーのParentItemプロパティは、ツリーの他のアイテムに割り当てられているかnull値です。

これは、ツリー・デザイン・インターフェース (#Prim_Tree.iTreeDesign) のOnItemGotParentメソッドに相当します。

Add メソッド

通常のLANSAリスト・コマンドを使うのとはちがい、Addメソッドは異なるデザインのクラスの追加を可能にします。

Addメソッドを使う場合、*ListFields機能を使ってフィールド値をデザインに引き渡すことはできません。ユーザーはプロセスを制御しており、プログラムを通じ、なんらかのデータを新しいデザイン・インスタンスに引き渡す必要があります。

異なるタイプのデザインを追加できるので、複雑なUI要件の処理をはるかに簡単にします。

Result パラメータ

Addメソッドで作成されたアイテム (`#Prim_Tree.TreeItem`) への参照です。

DesignType パラメータ

作成されるデザイン・インスタンスのクラスです。

指定されるクラスは、ツリーのDEFINE_COMでパラメータ化されたタイプとして指定されたクラスを継承しなければなりません。

DeleteAll メソッド

その名の通り、このメソッドはリストのすべてのアイテムを削除します。

これは `Clr_list` コマンドを使うのと同じです。

FindItem メソッド

FindItemは、各アイテムの Design Interface上のOnFindメソッドを呼び出します。

各ツリー・アイテムは、テストされる値を受け取り、ブール値の結果を返しながらか、順に呼び出されます。正数の結果が返されると、Findは終了します。値を返したアイテムから再開することができます。

```
Mthroutine Name(Search)
Define_Com Class(#Prim_Tree.TreeItem) Name(#FoundItem)
Reference(*dynamic)
Begin_Loop
#FoundItem <= #Tree.FindItem( #Search #FoundItem )
Leave If(#FoundItem *Is *null)
End_Loop
Endroutine
```

Result パラメータ

正数の結果を返す最初のアイテム (`#Prim_Tree.TreeItem`) への参照です。

Key パラメータ

検索される値またはオブジェクトです。

Keyは、どんなタイプの値またはオブジェクトも含むことができるバリエーションです。

Item パラメータ

その後から検索を開始するアイテム (`#Prim_Tree.TreeItem`) への参照です。

`StartItem`が指定されないと、検索は最初から開始されます。

iTreeDesign インターフェース

iTreeDesignは、ツリー・コントロール (#Prim_Tree) のデザインとして使われる再利用可能パーツにより実行されなければなりません。インターフェースには、それを通じて、ツリーがデザイン・インスタンスと通信し、Selection、Focusなどの状態の変更を知らせることができる一連のメソッドが含まれます。

以下のソースは、Prim_treeの適切なデザインを実行している再利用可能パーツのBegin_comと、ADD_ENTRYを使ってアイテムを作成する場合にデザインが受け取るフィールドを定義する*Listfields機能を示します。

```
Begin_Com Role(*EXTENDS #Prim_Panl3 *implements  
#Prim_Tree.iTreeDesign *ListFields #ListFields)
```

* Fields received as on Add_entry

```
Group_By Name(#ListFields) Fields(#Empno #Surname #Givenname  
#Deptment #Section #Deptdesc #Secdesc)
```

Tree Item パラメータ

デザインが追加された場合に作成される、対応するツリー・アイテム (#Prim_Tree.TreeItem) への参照です。

Tree Item パラメータ

対応する、削除されるツリー・アイテム (`#Prim_Tree.TreeItem`) への参照です。

Tree Item パラメータ

これから表示されるデザインに対応するツリー・アイテム (#Prim_Tree.TreeItem) への参照です。

Result パラメータ

Resultを設定し、Findが成功したかどうかを示します。

Key パラメータ

検索される値またはオブジェクトです。

Keyは、どんなタイプの値またはオブジェクトも含むことができるバリエーションです。

TreeItem パラメータ

デザインに対応するツリー・アイテム (`#Prim_Tree.TreeItem`) への参照です。

OnItemCollapsed メソッド

対応するアイテムがCollapseメソッドを使うか、ExpandedプロパティをFalseに設定することにより折りたたまれる場合に実行されます。

このメソッドは、ユーザーが折りたたまれたアイテムを表示するためにデザインの外見を変更できるようにします。例は、適切な折りたたみのイメージです。

TreeItem パラメータ

デザインに対応するツリー・アイテム (`#Prim_Tree.TreeItem`) への参照です。

OnItemExpanding メソッド

対応するアイテムがExpandメソッドを使うか、ExpandedプロパティをTrueに設定することにより展開される場合に実行されます。

このメソッドは、ユーザーが折りたたまれたアイテムを表示するためにデザインの外見を変更できるようにします。例は、適切な展開のメッセージです。

TreeItem パラメータ

デザインに対応するツリー・アイテム (`#Prim_Tree.TreeItem`) への参照です。

OnItemGotParent メソッド

対応するアイテムの親がParentItem Property.Expandの設定により変更される場合に実行されます。

このメソッドは、インデントを増やすために左マージンを増加させるなど、ユーザーがデザインの外見を変更して、アイテムの親が変更されたことを示すことができるようにします。

TreeItem パラメータ

デザインに対応するツリー・アイテム (`#Prim_Tree.TreeItem`) への参照です。

Tree プロパティ

親ツリー (#Prim_Tile) コントロールへの参照です。

ツリー・アイテム (Prim_Tree.TreeItem)

ツリー・アイテムは、ツリー (#Prim_Tree) にADD_ENTRY や Tree Add
メソッドなどのエントリーが追加される時に作成されます。

追加される各アイテムについて、対応するユーザー定義のDesignインスタ
ンスが作成され、コントロールの表示部分を提供します。

Design プロパティ

ツリー・デザイン (#Prim_Tile.iTileDesign) への参照です。

エントリーがツリーに追加された時に作成された、対応するDesignインスタンスへのアクセスを提供します。

Expanded プロパティ

アイテムが展開されます。すべての子アイテムが表示されます。

FocusedStyle プロパティ

アイテムがフォーカスアイテムになる時に、デザインに適用されるスタイル (#Prim_Vs.Style) への参照です。

アイテムがフォーカスを失うとスタイルは削除されます。

```
Mthroutine Name(OnAdd) Option(*Redefine)
#TreeItem.FocusStyle <= # MyStyles<Focus>
Endroutine
```

FocusedStyle プロパティ

アイテムがフォーカスアイテムになる時に、デザインに適用されるスタイル (#Prim_Vs.Style) のコレクションです。

アイテムがフォーカスを失うとスタイルは削除されます。

MarginBottom プロパティ

アイテムの下に適用されるマージンのサイズをピクセル数で定義します。

MarginLeft プロパティ

アイテムの左に適用されるマージンのサイズをピクセル数で定義します。

MarginRight プロパティ

アイテムの右に適用されるマージンのサイズをピクセル数で定義します。

MarginLeft プロパティ

アイテムの上部に適用されるマージンのサイズをピクセル数で定義します。

MouseOverStyle プロパティ

DirectX のみ

マウスがコントロールの物理的境界の中に入るとコントロールに適用されるスタイル (#Prim_vs.Style) を示します。マウスがコントロールを離れると、スタイルは消去されます。

MouseOverStyleは、多くのMouseEnterおよび対応するMouseLeaveイベントのコード化を不要にし、かわりに単純な宣言を可能にします。

MouseOverStyles プロパティ

DirectX のみ

マウスがコントロールの物理的境界の中に入るとコントロールに適用されるスタイル (#Prim_vs.Style) のコレクションです。マウスがコントロールを離れると、スタイルは消去されます。

MouseOverStylesは、プログラム上でのより複雑な外観の変更のコード化を可能にします。開発者は、1つの宣言型MouseOverStyleに依存するより自由に、必要な数のスタイル・レイヤーを追加できます。

ParentItem プロパティ

ツリー・アイテムへの参照です (#Prim_Tree.TreeItem)。

ParentItem プロパティを設定することにより、アイテムをツリー内で複雑な階層にすることができます。

ツリー・アイテムは、異なるツリーのアイテムを親にすることはできません。

SelectedStyle プロパティ

アイテムが選択された時に、デザインに適用されるスタイル (#Prim_Vs.Style) への参照です。

アイテムが選択を失うとスタイルは削除されます。

```
Mthroutine Name(OnAdd) Option(*Redefine)
#TreeItem.SelectedStyle <= # MyStyles<Focus>
Endroutine
```

SelectedStyle プロパティ

アイテムが選択された時に、デザインに適用されるスタイル (#Prim_Vs.Style) のコレクションです。

アイテムが選択を失うとスタイルは削除されます。

Style プロパティ

アイテムに適用されるスタイル (#Prim_vs.Style) を示します。Style プロパティが設定されると、アイテムに以前適用されたすべてのスタイルが削除されます。

Styles プロパティ

コントロールに適用されるスタイル (#Prim_vs.Style) のコレクションです。

Stylesは、プログラム上でのより複雑な外観の変更のコード化を可能にします。開発者は、1つの宣言型Styleプロパティに依存するより自由に、必要な数のスタイル・レイヤーを追加できます。

Tree プロパティ

親ツリー (#Prim_Tile) コントロールへの参照です。

System Application コントロール

Focus Control Changed イベント

FocusControlChangedは、フォーカスが別のコントロールに移動すると起動されます。

これは、すべてのコントロール (#Prim_CTRL) にあるGotFocusイベントと同じです。

Control パラメータ

フォーカス・コントロール (#Prim_CTRL) への参照です。

Appearance プロパティ

Prim_Vs.Appearanceインスタンスへの参照です。

Appearance プロパティは、アプリケーションの起動時でも実行時でも、適用される各コントロールの省略値のスタイルに、中心の位置を提供します。

Calendar プロパティ

すべてのカレンダー (#Prim_MTCL) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

CheckBox プロパティ

すべてのチェックボックス (#Prim_CKBX) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ComboBox プロパティ

すべてのコンボボックス (#Prim_CMBX) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

DateTime プロパティ

すべてのDateTime (#Prim_DAT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Edit プロパティ

すべてのエディットボックス (#Prim_EDIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。これには入力フィールド (#Prim_EVEF) の一部として使われる場合のEditが含まれます。

Grid プロパティ

すべてのグリッド (#Prim_GRID) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

GridFocused プロパティ

Grid itemインスタンスがフォーカスになり、グリッドがフォーカス・コントロールの場合、すべてのgrid item (#Prim_GDIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

GridFocusedInactive プロパティ

Grid itemインスタンスがフォーカス・アイテムで、グリッドがフォーカス・コントロールではない場合、すべてのgrid item (#Prim_GDIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

GridSelectedInactive プロパティ

Grid itemインスタンスが選択され、グリッドがフォーカス・コントロールではない場合、すべてのgrid item (#Prim_GDIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

GridMouseOver プロパティ

マウスが境界に入るとすべてのgrid item (#Prim_GDIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

GridSelected プロパティ

grid itemインスタンスが選択され、グリッドがフォーカス・コントロールの場合、すべてのgrid item (#Prim_GDIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

GroupBox プロパティ

すべてのグループボックス (#Prim_GPBX) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Image プロパティ

すべてのイメージ (#Prim_IMGE) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Label プロパティ

すべてのラベル (#Prim_LABL) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

List Box プロパティ

すべてのリストボックス (#Prim_LTBX) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListBoxFocused プロパティ

リストボックス項目 (#Prim_LBIT) インスタンスがフォーカスになり、リストボックスがフォーカス・コントロールの場合、すべてのリストボックス項目インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListboxFocusedInactive プロパティ

リストボックス項目 (#Prim_LBIT) インスタンスはフォーカス・アイテムで、リストボックスはフォーカス・コントロールではない場合、すべてのリストボックス項目 (#Prim_LBIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListboxSelectedInactive プロパティ

リストボックス項目 (#Prim_LBIT) インスタンスが選択され、リストボックスがフォーカス・コントロールではない場合、すべてのリストボックス項目 (#Prim_LBIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListBoxMouseOver プロパティ

マウスがリストボックス項目 (#Prim_LBIT) インスタンスの境界に入る時、すべてのリストボックス項目 (#Prim_LBIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListBoxSelected プロパティ

リストボックス項目 (#Prim_LBIT) インスタンスが選択され、グリッドがフォーカス・コントロールの場合、すべてのリストボックス項目 (#Prim_LBIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

List View プロパティ

すべてのリストビュー (#Prim_LTVW) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListViewFocused プロパティ

リストビュー項目 (#Prim_LTVW) インスタンスがフォーカスになり、リストビューがフォーカス・コントロールの場合、すべてのリストビュー項目 (#Prim_LVIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListviewFocusedInactive プロパティ

リストビュー項目 (#Prim_LTVW) インスタンスはフォーカス・アイテムで、リストビューはフォーカス・コントロールではない場合、すべてのリストビュー項目 (#Prim_LVIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListviewSelectedInactive プロパティ

リストビュー項目 (#Prim_LVIT) インスタンスが選択され、リストビューがフォーカス・コントロールではない場合、すべてのリストビュー項目 (#Prim_LVIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListViewMouseOver プロパティ

マウスがリストビュー (#Prim_LTVW) インスタンスの境界に入る時、すべてのリストビュー (#Prim_LTVW) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ListViewSelected プロパティ

リストビュー項目 (#Prim_LTVW) インスタンスが選択され、リストビューがフォーカス・コントロールの場合、すべてのリストビュー項目 (#Prim_LTVW) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Memo プロパティ

すべてのメモボックス (#Prim_MEMO) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Panel プロパティ

すべてのパネル (#Prim_PANL) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Popup プロパティ

すべてのポップアップ・パネル (#Prim_PPNL) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

ProgressBar プロパティ

すべてのプログレスバー (#Prim_PGBR) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

PushButton プロパティ

すべてのプッシュボタン (#Prim_PHBN) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

RadioButton プロパティ

すべてのラジオボタン (#Prim_RDBN) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

SpeedButton プロパティ

すべてのツールバーボタン (#Prim_SPBN) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

SpinEdit プロパティ

すべてのスピンエディットボックス (#Prim_SPDT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Tab プロパティ

すべてのタブフォルダ (#Prim_TAB) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

TrackBar プロパティ

すべてのトラックバー (#Prim_TKBR) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Tree プロパティ

すべてのツリービュー (#Prim_TRVW) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

TreeFocused プロパティ

ツリービュー項目 (#Prim_TVIT) インスタンスがフォーカスになり、ツリーがフォーカス・コントロールの場合、すべてのツリービュー項目 (#Prim_TVIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

TreeviewFocusedInactive プロパティ

ツリービュー項目 (#Prim_TVIT) インスタンスはフォーカス・アイテムで、ツリービューはフォーカス・コントロールではない場合、すべてのツリービュー項目 (#Prim_TVIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

TreeboxSelectedInactive プロパティ

ツリービュー項目 (#Prim_TVIT) インスタンスが選択され、ツリービューがフォーカス・コントロールではない場合、すべてのツリービュー項目 (#Prim_TVIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

TreeMouseOver プロパティ

マウスがツリービュー項目 (#Prim_TVIT) インスタンスの境界に入る時、すべてのツリービュー項目 (#Prim_TVIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

TreeSelected プロパティ

ツリービュー項目 (#Prim_TVIT) インスタンスが選択され、ツリービューがフォーカス・コントロールの場合、すべてのツリービュー項目 (#Prim_TVIT) インスタンスに適用されるスタイル (#Prim_VS) への参照です。

Cursors プロパティ

Windows テーマから採用される標準カーソル (#Prim_CRSR) のコレクションです。

Cursors を適用して、マウスがコントロールの境界に入ったり離れたったりした場合にカーソルを変化させることができます。多くの場合、Visual LANSa は、開発者が変更しなくても正しくカーソルを制御します。

```
#Button.Cursor <= #sys_Appln.Cursors<Hand>
```

FocusControl プロパティ

フォーカス・コントロール (#Prim_CTRL) への参照です。

GlassEnabled プロパティ

フォームがガラス (#Prim_Form.Glass) の外観を持つことができる環境でアプリケーションが実行されているかどうかを判断します。

True

フォームはガラスの外観を持つように構成され、アプリケーションは現在、ガラスで実行可能です。

False

フォームは、ガラスの外観を持つように構成されておらず、アプリケーションは現在、ガラスで実行不可能です。

HelpHandler プロパティ

HelpHandler インターフェイス (#Prim_App.iHelpHandler) を実行する再利用可能コンポーネントのインスタンスへの参照です。

開発者は、HelpHandler を指定することにより、F1 のヘルプの省略値の動作を置き換えることができます。

Images プロパティ

Windows テーマから採用される標準イメージ (#Prim_BMP) のコレクションです。

イメージは、特にユーザー・ツリー (#Prim_TREE) のようなユーザー定義のコントロールを構築し、アプリケーションに一貫した外観を持たせるのに便利です。

```
#ExpandCollapse.image <= #sys_appln.Images<ExplorerCollapsed>
```

Operating System プロパティ

アプリケーションを実行しているオペレーティング・システムの名前です。

PartitionShortCharLength プロパティ

PartitionShortCharLevelプロパティに基づいて、現在のpartition short char.lengthへのアクセスを提供します。

PartitionShortCharLevel プロパティ

現在のpartition short char. lengthへのアクセスを提供します。

RenderStyle プロパティ

RenderStyleは、RNDR X_Run引数と同様にアプリケーションの現在の表示エンジンへのアクセスを提供します。

Win32

アプリケーションはWin32表示エンジンを使用しており、DirectX表示はできません。これはRDNR値がWの場合と同じです。

Win32AndDirectX

アプリケーションはWin32表示エンジンを使用していますが、特定のコンポーネントのDirectXの使用をサポートします。これは、RDNR値がMの場合と同じです。

DirectX

アプリケーションはDirectX表示エンジンを使用していますが、特定のコンポーネントのWin32の使用をサポートします。これは、RDNR値がXの場合と同じです。

Style プロパティ

アプリケーション全体に適用されるスタイル (#Prin_VS.Style) への参照です。

指定されるスタイルは、すべてのフォームと再利用可能パーツに採用されます。

Styles プロパティ

アプリケーション全体に適用されるスタイル (#Prin_VS.Style) のコレクションです。

指定されるスタイルは、すべてのフォームと再利用可能パーツに採用されます。

TraceHandler プロパティ

TraceHandlerインターフェイス (`#Prim_App.iTraceHandler`) を実行する再利用可能コンポーネントのインスタンスへの参照です。

開発者は、トレース・ハンドラーを指定することにより、独自のトレース・メカニズムを実行できます。

リボン・コントロール

リボン

リボンは、プログラム機能をフォーム最上部の一連のタブまたはシートにまとめるコマンドバーです。リボンは、従来のメニューバーやツールバーを一つのコントロール・コンセプトにまとめます。

リボンの例は、バージョン13 LANSА IDE および Microsoft Office 2007以降に見られます。

すべてのアプリケーションの環境がリボンに適しているわけではないため、リボンがユーザーの要件にあった最善のソリューションかどうかを慎重に検討してください。

リボンの幅は常に現在使われているコンポーネントの幅と同じで、高さは固定されています。

Minimized

True

リボンは最小化して表示されます。シートのキャプションのみ表示されます。

False

リボン全体が表示されます。

OpenPage

リボンシートへの参照です (#Prim_RBBN.Sheet)。

アクティブ・リボンシートへのアクセスを提供します。

QuickAccessToolbarOnTop

True

QuickAccessツールバーは、リボンの上に表示されます。ガラスの外観のフォームでは、ツールバーはタイトルバー領域に表示されます。

False

クイック・アクセス・ツールバーは、リボンの直下に表示されます。

アクセス・キー

アクセス・キーは、リボンの特定のコントロールに添付されるショートカット・コマンド（F3, Ctrl+Sなど）と簡略アクセスのKeyTipを定義します。

Control

KeyTipおよびショートカットが適用されるコントロールのインスタンス（#Prim_CTRL）を参照してください。

アプリケーションメニュー

アプリケーションメニュー (#Prim_RBBN.ApplicationMenu) は、ブルーの [アプリケーション] ボタンをクリックして表示されるポップアップの左側に表示されます。

特別なデザイン・ルールはないため、アプリケーション・メニューはプレーンパネルです。動作や表示内容はすべて開発者に依存します。

```
Define_Com Class(#Prim_rbbn) Name(#Ribbon) Displayposition(1)
Height(140) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(8)
Width(1033)
Define_Com Class(#Prim_Rbbn.ApplicationMenu)
Name(#ApplicationMenu) Caption('File') Displayposition(1) Height(80)
Keytip('F') Left(30) Parent(#Ribbon) Tabposition(8) Top(0) Width(100)
Define_Com Class(#Prim_Rbbn.ApplicationMenuContent)
Name(#ApplicationMenuContent) Displayposition(5) Height(80)
Parent(#Ribbon) Tabposition(7) Top(0)
Define_Com Class(#Prim_Rbbn.ApplicationMenuFooter)
Name(#ApplicationMenuFooter) Displayposition(4) Height(20) Left(30)
Parent(#Ribbon) Tabposition(6) Top(0)
```

Caption

Application Menu キャプションは、ブルーの [アプリケーション・メニュー] ボタン、主に [ファイル] アイテムに表示されます。

AddressKey (#Prim_RBBN.AccessKey) とKeyTipsを使ってリボン・キーボード・ナビゲーションを実行するので、"&" をテキストに埋め込む必要はありません。

アプリケーションメニュー・コンテンツ

アプリケーションメニュー・コンテンツ

(#Prim_RBBN.ApplicationMenuContent) は、ブルーの [アプリケーション] ボタンをクリックした時に表示されるポップアップの右側に表示されます。

アプリケーション・コンテンツ・ページでアクティブなものによってコンテンツが全く異なることがあるため、多くの異なるコンテンツ・インスタンスを作らなければならない場合があります。アプリケーション・メニューと同様に、コンテンツはプレーン・パネルで、デザインと動作は、最終的に開発者に依存します。

```
Define_Com Class(#Prim_rbbn) Name(#Ribbon) Displayposition(1)
Height(140) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(8)
Width(1033)
Define_Com Class(#Prim_Rbbn.ApplicationMenu)
Name(#ApplicationMenu) Caption('File') Displayposition(1) Height(80)
Keytip('F') Left(30) Parent(#Ribbon) Tabposition(8) Top(0) Width(100)
Define_Com Class(#Prim_Rbbn.ApplicationMenuContent)
Name(#ApplicationMenuContent) Displayposition(5) Height(80)
Parent(#Ribbon) Tabposition(7) Top(0)
Define_Com Class(#Prim_Rbbn.ApplicationMenuFooter)
Name(#ApplicationMenuFooter) Displayposition(4) Height(20) Left(30)
Parent(#Ribbon) Tabposition(6) Top(0)
```

アプリケーションメニュー・フッター

アプリケーションメニュー・フッター

(#Prim_RBBN.ApplicationMenuFooter) は、ブルーの [アプリケーション] ボタンをクリックした時に表示されるポップアップの下部に表示されます。

アプリケーションメニューやコンテンツと同様、フッターはプレーン・パネルで、デザインと動作は最終的に開発者に依存します。

```
Define_Com Class(#Prim_rbbn) Name(#Ribbon) Displayposition(1)
Height(140) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(8)
Width(1033)
Define_Com Class(#Prim_Rbbn.ApplicationMenu)
Name(#ApplicationMenu) Caption('File') Displayposition(1) Height(80)
Keytip('F') Left(30) Parent(#Ribbon) Tabposition(8) Top(0) Width(100)
Define_Com Class(#Prim_Rbbn.ApplicationMenuContent)
Name(#ApplicationMenuContent) Displayposition(5) Height(80)
Parent(#Ribbon) Tabposition(7) Top(0)
Define_Com Class(#Prim_Rbbn.ApplicationMenuFooter)
Name(#ApplicationMenuFooter) Displayposition(4) Height(20) Left(30)
Parent(#Ribbon) Tabposition(6) Top(0)
```

コンテキスト・グループ

コンテキスト・グループを使って、同じグループに属するシートを集めたり (#Prim_RBBN.Sheet)、単にシートの詳細情報をタイトルバーに表示したりできます。

Microsoft Wordでは、コンテキスト・グループをテーブルなどの機能に使っています。ドキュメント内でテーブルを選択すると、追加のリボンシートがー対表示されます。

```
Define_Com Class(#Prim_rbbn.ContextualGroup) Name(#ContextualGroup)
Caption('Contextual Group') Parent(#Ribbon)
Define_Com Class(#Prim_rbbn.Sheet) Name(#Sheet1) Caption('Sheet1')
Contextualgroup(#ContextualGroup) Displayposition(1) Height(91) Left(0)
Parent(#Ribbon) Tabposition(3) Top(49) Width(1033)
Define_Com Class(#Prim_rbbn.Sheet) Name(#Sheet2) Caption('Sheet2')
Contextualgroup(#ContextualGroup) Displayposition(2) Height(91) Left(0)
Parent(#Ribbon) Tabposition(4) Top(49) Width(1033)
```

Caption

コンテキスト・グループを構成するシートの上に、見出しとして表示されるテキストです。

Visible

Visibleプロパティは、そのグループに属するすべてのシートの表示状態を上書きします。

True

このコンテキスト・グループに属するすべてのシートは、そのシートが Visible (False) でない限り表示されます。

False

このコンテキスト・グループのすべてのシートは、条件にかかわらず非表示です。

リボン・グループ

グループは、シートのサブセットです。ほとんどのシートには、リボン上のコマンドを関連するサブジェクトにまとめる、複数の異なるグループがあります。コントロールは、グループに属する時のみリボン上に表示されます。リボンのすべてのパーツと同様に、各グループはプレーン・パネルです。デザインは開発者に依存します。

```
Define_Com Class(#Prim_rbbn) Name(#Ribbon) Displayposition(1)
Height(140) Left(8) Parent(#COM_OWNER) Tabposition(1) Top(8)
Width(1033) Define_Com Class(#Prim_rbbn.Sheet) Name(#Sheet1)
Caption('Sheet1') Displayposition(2) Height(91) Left(0) Parent(#Ribbon)
Tabposition(1) Top(49) Width(1033)
Define_Com Class(#Prim_rbbn.Group) Name(#Sheet1Group1)
Caption('Clipboard') Dialogbutton(True) Displayposition(1) Height(91)
Left(12) Parent(#Sheet1) Tabposition(1) Top(0) Width(100)
Define_Com Class(#Prim_rbbn.Group) Name(#Sheet1Group2)
Caption('Styles') Displayposition(2) Height(91) Left(124) Parent(#Sheet1)
Tabposition(2) Top(0) Width(100)
Define_Com Class(#Prim_rbbn.Group) Name(#Sheet1Group3)
Caption('Editing') Displayposition(3) Height(91) Left(236)
Parent(#Sheet1) Tabposition(3) Top(0) Width(100)
```

DialogButton クリック・イベント

グループの [ダイアログ] ボタンがクリックされた時に起動します。

DialogButton プロパティ

True

グループの右下隅に [ダイアログ] ボタンを表示します。

False

[ダイアログ] ボタンを非表示にします。

ヘルプ・ツールバー

リボン・ヘルプ・ツールバーは、リボンの右上隅に表示されます。リボンのすべてのパーツと同様、ヘルプ・ツールバーはプレーン・パネルです。デザインは開発者に依存します。

Image プロパティ

「ベースLANSAグラフィック」(#Prim_FLBX)を参照してください。
グループが折りたたまれた時に表示される32x32のイメージを示します。

クイック・アクセス・ツールバー

クイック・アクセス・ツールバーは、リボンの左側に表示されます。通常は、[保存] など一般に使われるコマンドに使われる小さいイメージ行を表示します。リボンの上下どちらに表示されるか

はQuickAccessToolbarOnTop プロパティリボンに制御されています。

リボンのすべてのパーツと同様、クイック・アクセス・ツールバーはブレイク・パネルです。デザインは開発者に依存します。

リボン・シート

リボン・シートはリボン・グループ (**#Prim_RBBN.Group**) のコンテナです。シートは、リボンを構成するタブ表示のページを定義します。

KeyTip

KeyTipは、キーボード使用中に、リボンの特定のコマンドにアクセスするために使われる文字列です。

KeyTipは通常頭文字1字ですが、より複雑なリボンでは2文字を使用した方がよい場合があります。

コンテキスト・グループ

「コンテキスト・グループ」を参照してください。

(#Prim_RBBN.ContextualGroup)

シートを、コンテキスト・グループが表示された時に表示できるセットにまとめるのに使われます。

テーブル・レイアウト

DirectX のみ

テーブル・レイアウトは、グリッド・レイアウトやMicrosoft Wordのテーブルと同様に、コントロールを行や列に分割します。

Table Column

テーブル・レイアウトの一部として列幅の定義に使用されます。
以下の例では列が3つあります。最初の列の幅は25ピクセルです。あとの2列は比例的で、残りのスペースを均等に分けます。

```
Define_Com Class(#prim_tblo) Name(#TableLayout)
Define_Com Class(#Prim_tblo.Column) Name(#Column1)
Parent(#TableLayout) Units(Pixels) Width(25)
Define_Com Class(#Prim_tblo.Column) Name(#Column2)
Parent(#TableLayout)
Define_Com Class(#Prim_tblo.Column) Name(#Column3)
Parent(#TableLayout)
```

Units

Width/Heightプロパティとともに、列/行のサイズを決定します。

Pixels

列/行の幅/高さは、Width/Heightプロパティで指定された値と等しい幅/高さのピクセル値です。

Proportion

サイズは、固定サイズの列/行の算出後、残りの使用可能スペースのある比率として定義されます。

以下の例では、Column1は使用可能スペースの40%を使用し、Column2は60%を使用します。

```
Define_Com Class(#Prim_tblo.Column) Name(#Column1)
Parent(#TableLayout) Width(2)
Define_Com Class(#Prim_tblo.Column) Name(#Column2)
Parent(#TableLayout) Width(3)
```

Width

Unitsプロパティとともに、列の幅を決定します。

Widthは数字で、ピクセルの特定の数または使用可能スペースのある比率を示します。

Table Row

テーブル・レイアウトの一部として行の高さの定義に使用されます。以下の例では、行は3つあります。最初の行の幅は、25ピクセルです。あとの2列は比例的で、残りのスペースを均等に分けます。

```
Define_Com Class(#prim_tblo) Name(#TableLayout)
Define_Com Class(#Prim_tblo.Row) Name(#Row1) Parent(#TableLayout)
Units(Pixels) Width(25)
Define_Com Class(#Prim_tblo.Row) Name(#Row2) Parent(#TableLayout)
Define_Com Class(#Prim_tblo.Row) Name(#Row3) Parent(#TableLayout)
```

Height Column

Unitsプロパティとともに列の高さを決定します。

Heightは数字で、ピクセルの特定の数または使用可能スペースのある比率を示します。

Table Item

項目に制御されているコントロールのサイズと位置を決定します。

Column

列 (`#prim_tblo.Column`) への参照です。

`ColumnSpan` プロパティとともに、項目に制御されるコントロールの開始位置と水平方向の範囲を指定します。

ColumnSpan

Columnプロパティとともに、項目に制御されるコントロールの開始位置と水平方向の範囲を指定します。

Row

行 (#prim_tblo.Row) への参照です。

RowSpanプロパティとともに、項目に制御されるコントロールの開始位置と垂直方向の範囲を指定します。

RowSpan

Rowプロパティとともに、項目に制御されるコントロールの開始位置と垂直方向の範囲を指定します。