Magic xpa
逆引き辞典

OUTPERFORM THE FUTURE™

本マニュアルに記載の内容は、将来予告なしに変更することがあります。これらの情報について MSE（Magic Software Enterprises Ltd.）および MSJ（Magic Software Japan K.K.）は、いかなる責任も負いません。

本マニュアルの内容につきましては、万全を期して作成していますが、万一誤りや不正確な記述があったとしても、MSE および MSJ はいかなる責任、債務も負いません。

MSE および MSJ は、この製品の商業価値や特定の用途に対する適合性の保証を含め、この製品に関する明示的、あるいは黙示的な保証は一切していません。

本マニュアルに記載のソフトウェアは、製品の使用許諾契約書に記載の条件に同意をされたライセンス所有者に対してのみ供給されるものです。同ライセンスの許可する条件のもとでのみ、使用または複製することが許されます。当該ライセンスが特に許可している場合を除いては、いかなる媒体へも複製することはできません。

ライセンス所有者自身の個人使用目的で行う場合を除き、MSE または MSJ の書面による事前の許可なしでは、いかなる条件下でも、本マニュアルのいかなる部分も、電子的、機械的、撮影、録音、その他のいかなる手段によっても、コピー、検索システムへの記憶、電送を行うことはできません。

サードパーティ各社商標の引用は、MSE および MSJ の製品に対するコンパチビリティに関しての情報提供のみを目的としてなされるものです。

本マニュアルにおいて、説明のためにサンプルとして引用されている会社名、製品名、住所、人物は、特に断り書きのないかぎり、すべて架空のものであり、実在のものについて言及するものではありません。

Magic® は Magic Software Japan K.K. の登録商標です。
Magic xpa® は Magic Software Enterprises Ltd. のイスラエルその他の国での商標または登録商標です。
Magic xpa Studio、Magic xpa Client、Magic xpa Enterprise Server および Magic xpa RichClient Server は Magic Software Japan K.K. の商標です。
Pervasive.SQL® は Pervasive Software, Inc. の商標です。
IBM®, iSeries™, xSeries®, DB2® および WebSphere® は、IBM Corporation の商標または登録商標です。
Microsoft® および FrontPage® は、Microsoft Corporation の登録商標です。また、Windows™, WindowsNT™ および ActiveX ™ は Microsoft Corporation の商標です。
Oracle® は Oracle Corporation の登録商標です。
Systinet™ は、Hewlet-Packard Development Company の商標です。

一般に、会社名、製品名は各社の商標または登録商標です。

MSE および MSJ は、本製品の使用またはその使用によってもたらされる結果に関する保証や告知は一切していません。この製品のもたらす結果およびパフォーマンスに関する危険性は、すべてユーザが責任を負うものとします。

この製品を使用した結果、または使用不可能な結果生じた間接的、偶発的、副次的な損害（営利損失、業務中断、業務情報の損失などの損害も含む）に関し、事前に損害の可能性が勧告されていた場合であっても、MSE および MSJ、その管理者、役員、従業員、代理人は、いかなる場合にも一切責任を負いません。

Magic xpa 逆引き辞典

第 10 版　2023 年 4 月 30 日

目次

第 1 章 :操作方法とワークスペース

第 2 章 :プロジェクトとアプリケーション

第 3 章 :モデル

第 4 章 :Magic エンジン

第 5 章 :タスク

第 6 章 :ロジックの拡張

第 7 章 :実行

第 8 章 :サブフォーム

第 9 章 :ツリーコントロール

第 10 章 :さらなるロジックの拡張

第 11 章 :日付と時刻

第 12 章 :GUI の処理

第 13 章 :XML

第 14 章 :COM

第 15 章 :コンポーネント

第 16 章 :動作環境

第 17 章 :データソースの定義

第 18 章 :メインプログラム

第 19 章 :データビュー

第 20 章 : 式

第 21 章 :レポート

第 22 章 :マージ

第 23 章 :メッセージング

第 24 章 :マルチタスク

第 25 章 :ウィンドウインタフェース

第 26 章 :メニュー

第 27 章 :Unicode

第 28 章 :アプリケーションのデバッグ

第 29 章 :イベントとハンドラ

第 30 章 :セキュリティ

第 31 章 :ユーティリティ

第 32 章 :開発環境

第 33 章 :Web サービス（コンシューマ）

第 34 章 :Web サービス（プロバイダ）

第 35 章 :データベース

第 36 章 :ブローカ

第 37 章 :ソース管理

第 38 章 :マルチ言語サポート

第 39 章 :リッチクライアント補足

第 40 章 :.NET 統合

第 41 章 :モバイル開発

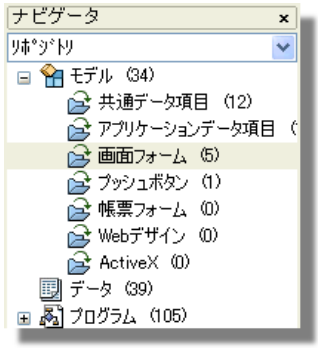
第 42 章 :Java 統合

第 43 章 :Web Client の開発

第 1 章： 操作方法とワークスペース

プロジェクトのオブジェクトを整理するには

Magic xpa のプロジェクトは、大規模な基幹業務用のアプリケーションの場合があります。従って、開発内容を整理しておくことが重要になります。Magic xpa は、開発内容をリポジトリ（モデル、データ、プログラム、ヘルプ、権利、メニューやリソース）に分けることでこのような作業を支援します。



右の図は、これらのリポジトリを示しています。

これらのリポジトリ内で、さらにフォルダを使用して入力項目を分類することができます。右の例では、異なるタイプのモデルを含んだモデルリポジトリ用に 7 つのフォルダが作成されています。

フォルダを作成する

1. 作成したいフォルダの上位にカーソルを置きます（この例では、画面フォーム）。
2. *F4* を押下します（または、コンテキストメニューから行作成を選択するか、プルダウンメニューから 編集 → 行作成 を選択）。
3. フォルダ名を入力します。

ヒント : *各入力項目の右側に表示される数値は、各セクション内の項目数を表しています。例えば、［モデル］セクション内には、現在は 34 のモデルが定義されています。また、この内 12 のモデルは「**共通データ項目**」フォルダ内に定義されています。*

*新しいフォルダを追加すると、フォルダ名として「**新しいフォルダ n**」が表示されます。ここでの数値（n）は、ただの連番を表しています。この例の場合、**新しいフォルダ**は、9 番目に追加されるフォルダとなります。*

フォルダを削除する

必要条件 : フォルダを削除する前に、フォルダ内のオブジェクトが空になっている必要があります。フォルダ内のオブジェクトを別のフォルダに移動してください。また、バージョン管理を使用している場合、最初にこれらをチェックアウトする必要があります。

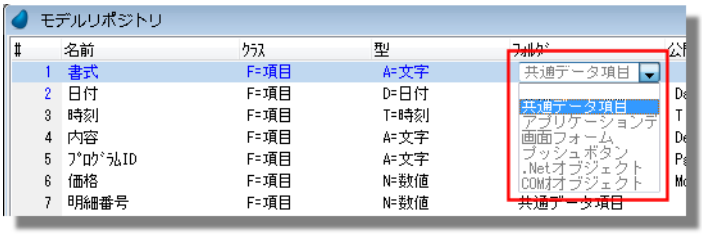
1. 削除したいフォルダ上にカーソルを置きます。
2. *F3* を押下します（または、コンテキストメニューから行削除を選択するか、プルダウンメニューから 編集 → 行削除 を選択）。
3. フォルダの削除の確認ダイアログが表示されたら、はいをクリックします。

フォルダを移動する

1. 移動したいフォルダをクリックします。その際、対応するワークスペースも開きます。
2. フォルダを移動したい位置にドラッグします。
3. フォルダをドロップすると、フォルダのドラッグ＆ドロップの確認ダイアログが表示されます。はいをクリックします。

フォルダにオブジェクトを移動する

既存のオブジェクトをフォルダ内に移動させるには、以下にあげる方法があります。



必要条件： バージョン管理を使用している場合、最初にリポジトリを**チェックアウト**してください。

フォルダカラムを使用してフォルダにオブジェクトを移動する

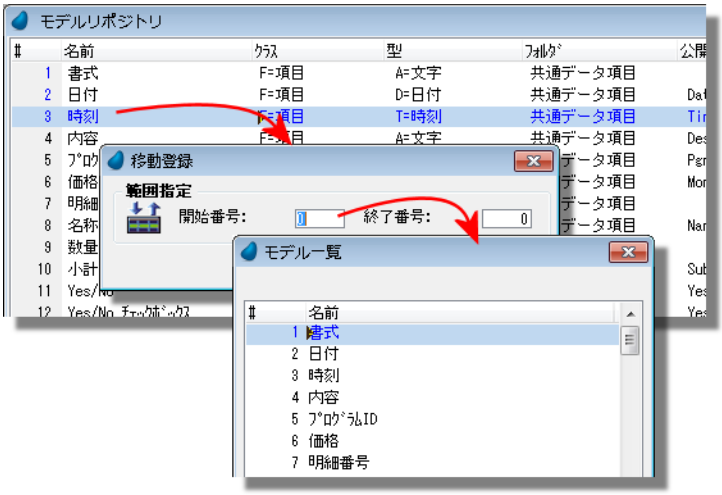
1. 移動したいオブジェクトを選択します。
2. **フォルダ**カラムで、コンボボックスを**クリック**します。
3. 移動するフォルダを選択します。

オブジェクトが現在のフォルダから消え、選択されたフォルダに移動されます。

移動コマンドを使用して移動する

1. オブジェクトを移動させたいフォルダに移動します。
2. ***Ctrl+Shift+M*** を押下します。**移動登録**ダイアログが表示されます。
3. **開始番号**に移動したいオブジェクトグループの最初の項目を選択してください（**ズーム**して、一覧から選択することもできます）。
4. 1 項目のみ移動する場合は、ここで ***Enter*** を押します。
5. 複数の項目を移動する場合は、**終了番号**に移動したいオブジェクトグループの最後の項目を選択して ***Enter*** を押します。

オブジェクトは現在のカーソル位置に表示され、以前の位置からは削除されます。

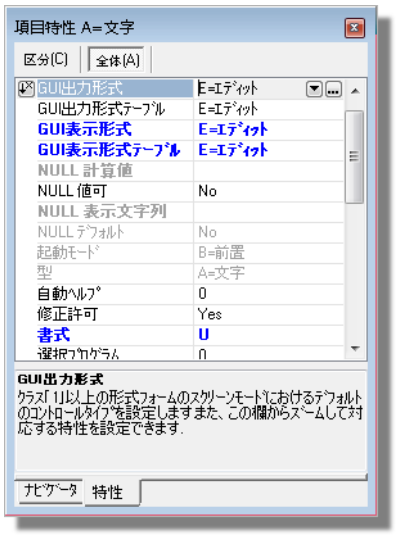


ヒント：*この方法は、連続した複数のオブジェクトを移動する場合に有効です。*

参照： 「入力行を移動するには」（13 ページ）

パレットを分離するには

開発エンジンのパレットは、サイズを変更したり、移動したり、Magic ウィンドウの縁に結合したりすることができ、作業内容に応じて柔軟に対応することができます。また、複数のパレットを 1 つのウィンドウに結合させることでスペースを節約することができます。1 つのパレットを別のパレット側に移動させることでパレットは結合されます。



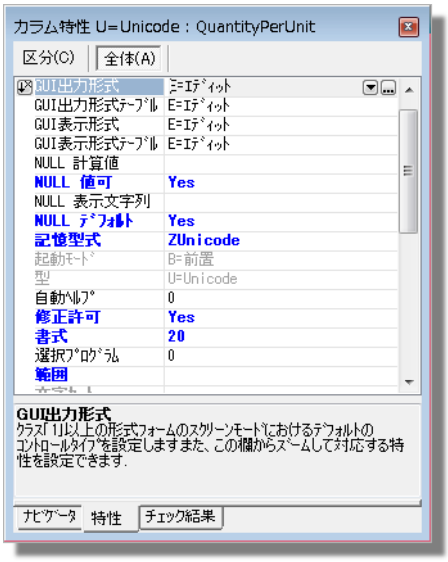
結合されたパレットを分離する

1. パレットにフォーカスを移します。
2. *Ctrl* を押下した状態で、パレットのタイトルバーを任意の方向にドラッグします。
3. *Ctrl* を離します。
4. ３つ以上のパレットが統合されている場合は、同じ操作を繰り返してください。

上記の操作でパレットは分離されます。

複数のペインを一つに統合するには

Magic xpa の各ペインは、作業のやり方に応じてサイズを変更したり、移動したり、MDI の端にくっつけたりと柔軟に表示させることができます。スペースを節約するために、複数のペインを 1 つのウィンドウに結合させることもできます。あるペインを別のペインに被せるように移動することで結合させることができます。

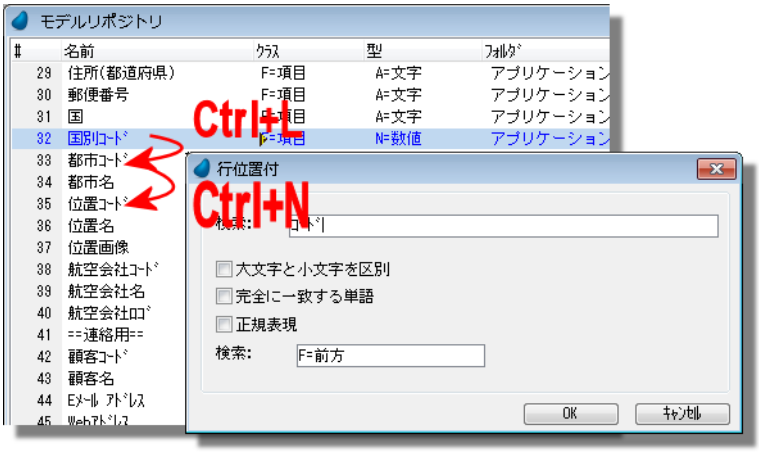


結合されたペインを分離する

1. ペインにフォーカスを移動します。
2. ***Ctrl*** を押下しながら、ペインのタイトルバーを任意の方向にドラッグします。
3. ***Ctrl*** を離します。
4. ペインが 3 つ以上の結合されている場合は、上記の操作を繰り返します。

ペインが分離されます。

開発用テーブルの行位置付けを行うには



リポジトリまたは、プログラム内で特定の行を見つけたい場合があります。このような場合は、位置付け機能を利用することで実現することができます。位置付け機能には、いくつかのオプションがあり、これによって柔軟に対応することができます。デフォルトでは、現在の行から以降の行に対するテキストをもとに位置付けることができます。

例えば、上の例のように、N コードを指定することで、製品コードにもスタジオコードにも位置付けることができます。入力された文字列によって行内に設定されたいる文字列を検索します。**大文字と小文字を区別**というチェックボックスを**チェック**すると、文字列の大文字と小文字を別の文字として検索します。

正規表現のチェックボックスをチェックした場合、複雑なマスク指定を行うこともできます。『リファレンスヘルプ』には、正規表現で使用できる書式の説明があります。

行に位置付ける

1. ***Ctrl+L*** を押下します（または、**編集→クイックアクセス→位置付**）。
2. **行位置付**ダイアログボックスに検索したい文字列を入力します。
3. ***Enter*** を押下します（または、OK を**クリック**）。

入力した文字列をもとに検索処理が実行され、カーソルが最初の行に移動します。***Ctrl+N***（または、**編集→ クイックアクセス→次に位置付**）によって次の検索対象に位置付けられます。***Ctrl+Shift+N***（または、**編集→ クイックアクセス→前に位置付**）によって前の検索対象に位置付けられます。

ヒント :行位置付 *は、現在オープンされているテーブルの行に対して実行されます。このため、特定のフォルダ内で実行した場合は、そのフォルダ内の行のみを検索対象とします。リポジトリ全体を検索したい場合は、リポジトリ全体を開いた状態で実行してください。*

参照 : 「行番号を指定して移動するには」（6 ページ）

行番号を指定して移動するには

Magic 内の全ての項目は、連番を持っています。この番号は、覚えておく必要はありません。使用したい項目を見つける手段として**ズーム**や**位置付け機能**を利用することができます。しかし、行番号を知っていた場合、その番号の行に移動することができます。

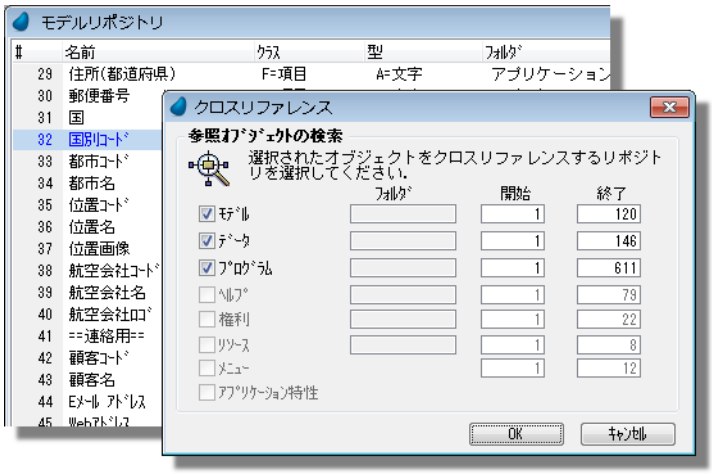
行に移動する

1. アクセスしたいリポジトリにフォーカスを移します。
2. ***Ctrl+J*** を押下します（または、**編集 → クイックアクセス →行ジャンプ**）。
3. 位置付けたい行番号を入力します。
4. ***Enter*** を押下します。

指定した行に位置付けられます。

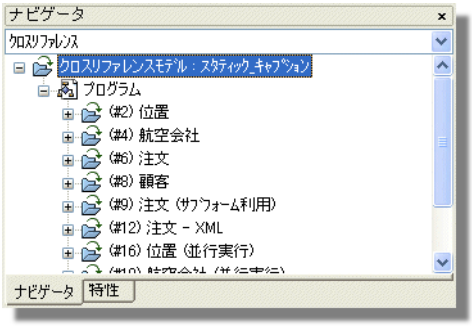
オブジェクトがどこで使用されているかを確認するには

全てのプログラミングにおける主要な問題の 1 つが「もしこの項目を変更したら、他のどの項目に影響するのか?」ということです。1 つの簡単な変更によって、意図しない影響を引き起こす場合があります。Magic xpa には、クロスリファレンスという特定のオブジェクトを参照しているオブジェクトを検索する機能があります。



クロスリファレンスを使用する

1. 検索対象の項目にカーソルを移動します。
2. *Ctrl+F* を押下します（または、編集 → 検索と置換 → クロスリファレンス）。
3. クロスリファレンスダイアログが開きます。デフォルトでは、参照可能な全てのリポジトリをチェックするように設定されています。必要に応じてチェック対象を絞ることができます。
4. *Enter* を押下します（または OK をクリック）。



クロスリファレンスを実行すると、指定されたオブジェクトを使用している全てのオブジェクトが表示されます。

この表示は以下のように利用できます。

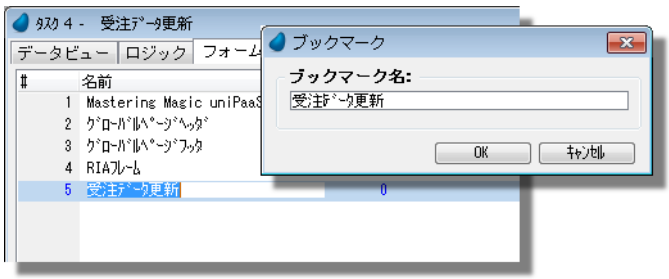
オブジェクト名をクリックすることで、そのオブジェクトに移動することができます。1 つのオブジェクトを複数のオブジェクトが参照しているような場合に有効です。

この表示内容は削除することもできます。参照項目の修正を行った後で、*F3* を押下して（または 編集→行削除：Magic xpa の通常の行削除と同じです）表示を削除してください。

クロスリファレンスの結果をテキストファイルに保存したり、印刷することもできます。（編集→検索と置換→検索結果の保存 と 検索結果の印刷 ）

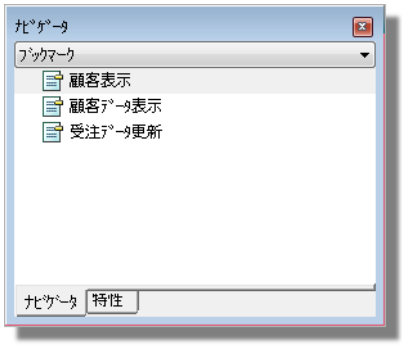
よく使用するオブジェクトにブックマークを設定するには

プログラミングやテストの際に同じオブジェクトに何回も繰り返しアクセスする場合があります。これらのオブジェクトは、タスクツリー内のオブジェクトの場合もあります。これらのオブジェクトに対して、**ブックマーク**という機能を利用することでマークすることができます。ブックマークが定義されたオブジェクトは、**ブックマーク**ペイン（**ナビゲータ→ブックマーク**）に表示されます。



現在の場所にブックマークを定義する

1. ***Ctrl+Shift+B*** を押下します（または、**オプション→ブックマーク**）。
2. **ブックマーク**ダイアログが表示されます。ブックマークとして登録したい名前を入力します。
3. ***Enter*** を押下します（または OK を**クリック**）。



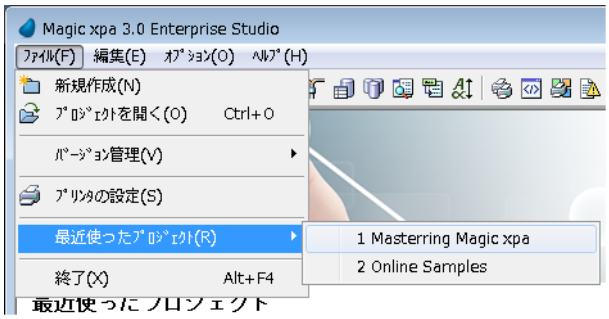
定義されたブックマークが**ブックマーク**ペイン（**ナビゲータ→ブックマーク**）に表示されます。このブックマーク名を**クリック**すると対応するオブジェクトにカーソルが移動します。

コンテキストメニューの**ツリー編集**を使用することでブックマークの名前を変更することができます。また、***F3***（**編集→行削除**）を使用して削除することができます。

ヒント : *登録できるブックマークの最大値は、**動作環境**ダイアログで指定できます。最大値を変更するには、**ブックマークの最大登録数（オプション→設定→動作環境→動作設定）**の設定値を変更します。*

最近使用したプロジェクトを迅速に開くには

複数のプロジェクトで開発を行うことがあります。通常、プロジェクトを開くには *Ctrl+O* を押下します（ファイル→プロジェクトを開く）。しかし、今まで作業していたプロジェクトを簡単に開くこともできます。



最近使用したプロジェクトを開く

1. プルダウンメニューからファイル→最近使ったプロジェクトを選択します。
2. 開きたいプロジェクトをクリックします。

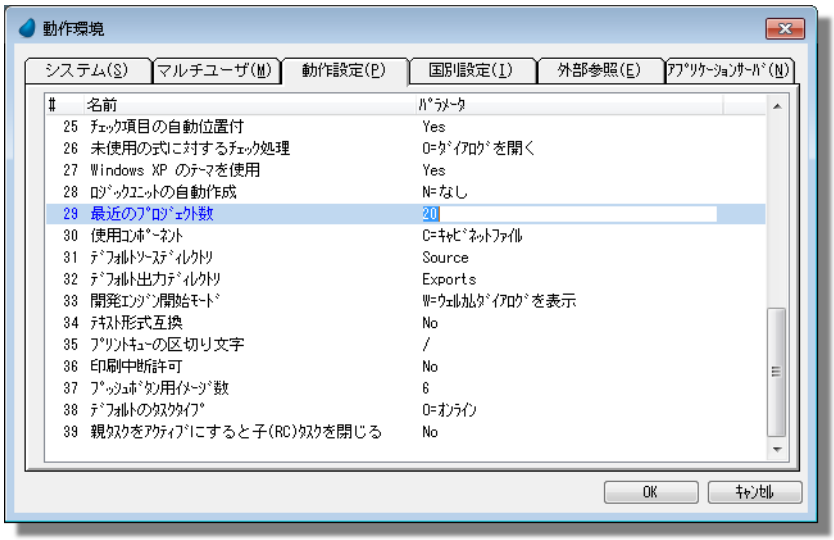
現在作業中のプロジェクトは保存され、選択されたプロジェクトが開きます。

注： プロジェクト名の一覧表示をスクロールするとステータスバーにプロジェクトファイル名がフルパスで表示されます。同じ名前や似たような名前のプロジェクトをいくつか使用している場合は、この表示を参照してください。

参照： 「プロジェクトを切り替えるには」（17 ページ）
「最近開いたプロジェクトの表示数を変更するには」（10 ページ）

最近開いたプロジェクトの表示数を変更するには

最近開いたプロジェクトの表示数を変更することができます。デフォルトは *4* ですが、最大 *99* まで表示させることができます。



最近開いたプロジェクトの数を設定する

1. プルダウンメニューから**オプション→設定→動作環境→動作設定**を選択します。
2. **最近のプロジェクト数**の値を変更します。

大きな値を設定すると、表示されるプロジェクト数が増えます。

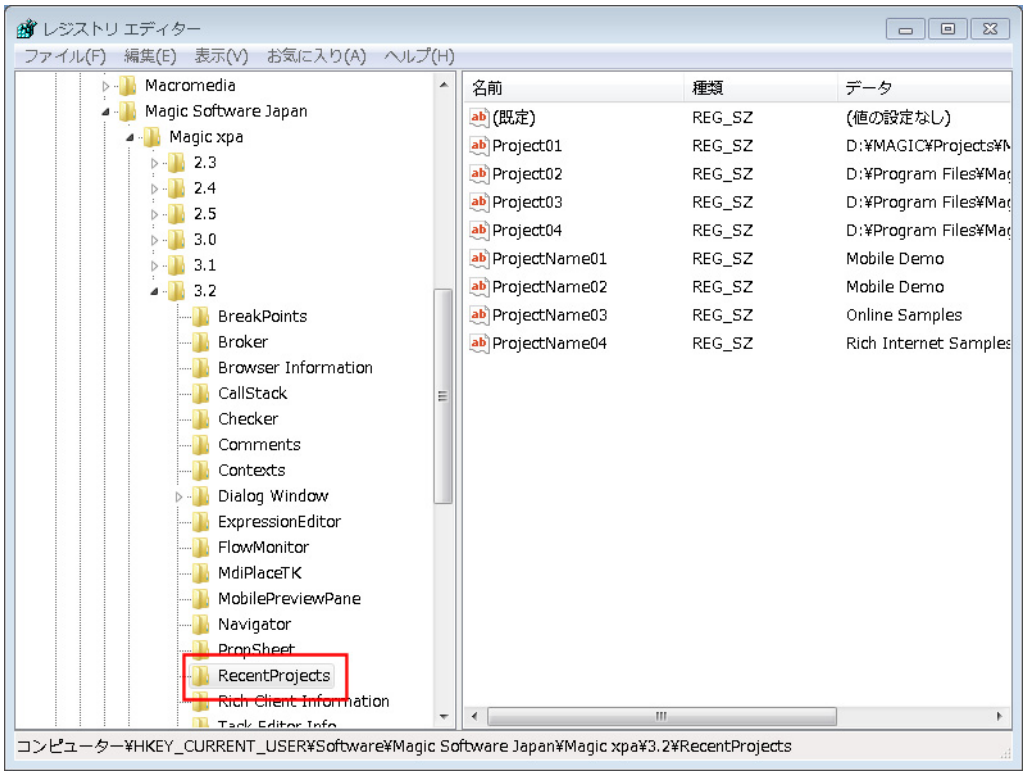
Magic.ini ファイルの **NumberOfRecentProjects** の設定値を変更することで表示数を変えることもできます。

最近開いたプロジェクトリストを変更するには

ウェルカムダイアログ、または**ファイル**メニューから最近開いたプロジェクトを選択する時に、表示されるプロジェクトのリストを変更することができます。

これは、Windows のレジストリーで行います。

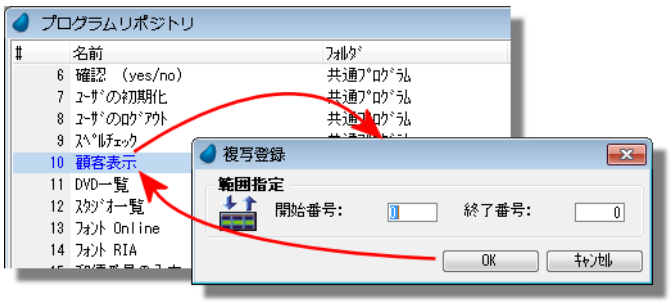
1. Windows の**スタート**メニューから、ファイル名を指定して実行を選択してください。
2. ファイル名を指定して実行ボックスで、**regedit** をタイプ入力し、***Enter*** を押下してください。
3. RecentProjects のエントリを次のキーから探してください：HKEY_CURRENT_USER > Software > Magic Software Japan > Magic xpa > < バージョン番号 > > RecentProjects.



入力行を複写するには

Magic xpa で新しいオブジェクトを作成する必要がある場合、類似したオブジェクトをコピーして利用することで開発時間を短縮することができます。 1 つのオブジェクトをコピーする場合は、この操作を繰り返す必要があります。

1,*Ctrl+Shift+R* を押下する



3,11 行目にコピーされた行が追加される

2, 行番号を入力して OK をクリックする

必要条件： バージョン管理を使用している場合、最初にリポジトリをチェックアウトしてください。

複写機能を使用する

1. 新規にオブジェクトを作成したい行にカーソルを移動します。
2. *Ctrl+Shift+R* を押下します（または、**編集→登録→複写登録**）。
3. ダイアログボックスが表示されたら、複写したい行番号を入力します。行番号があらかじめ分かっている場合は、そのまま入力します。分からない場合は、**開始番号**からズームして**項目一覧**を表示して選択します。
4. **終了番号**のデフォルトは、**開始番号**と同じ値になります。1 つだけコピーする場合はこのまま *Enter* を押下します。連続した複数の行をコピーする場合は、複写行グループの最後の番号を入力します。
5. *Enter* を押下します（または OK をクリック）。

選択された項目は、現在行のすぐ下にコピーされます。

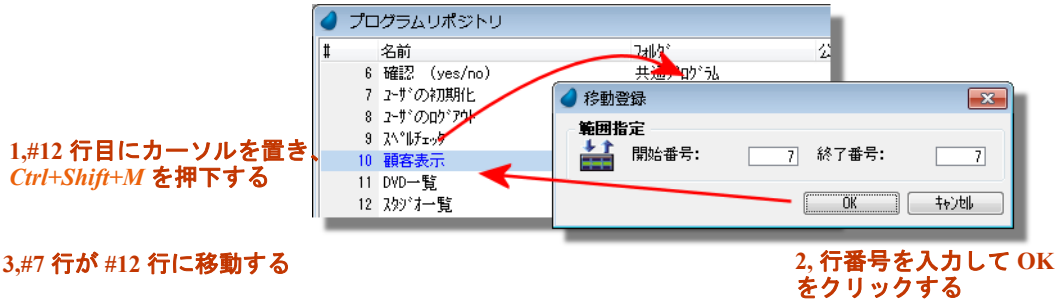
複数のタスクにわたってデータビューの行のコピー & ペーストを行うことができます。（サポートバージョン：4.6）

ヒント：*コピーした項目は、すぐ名前を変更するようにしてください。そうしないと、コピー元の項目との区別が付かなくなります。また、バージョン管理を使用している場合、最初にリポジトリをチェックアウトしてください。チェックアウトしない場合はこの機能は利用できません。*

注： リポジトリ内でのカット＆ペースト操作は、テキストのみに対して行うことができます。リポジトリ内の項目名のみが対象で、オブジェクト自身のコピーは行われません。

入力行を移動するには

作業内容を整理しておくために、リポジトリ内の項目を入れ替える必要がある場合があります。例えば、アルファベット順であったり、機能にもとづいたグループに分けようにします。（メインプログラムを除いた）どのような項目もリポジトリ中で移動させることができます。



必要条件： バージョン管理を使用している場合、最初にリポジトリがチェックアウトされていることを確認してください。

入力行を移動する

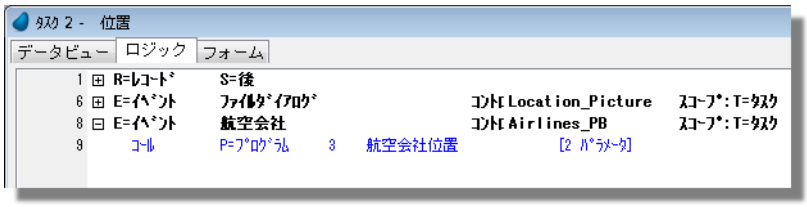
1. 項目を移動したい行にカーソルを移動します。
2. *Ctrl+Shift+M* を押下します（または、編集→登録→移動登録）。
3. ダイアログボックスが表示されたら、**開始番号**に移動する項目の行番号を入力します。番号を知っているのであれば、直接入力します。分からない場合は、ここからズームして一覧から選択します。
4. **終了番号**のデフォルトは**開始番号**と同じになるため、1 行だけ移動する場合は *Enter* を押下するだけです。複数の連続した行を移動する場合は、移動する行の最終番号を入力します。
5. *Enter* を押下します（または、OK をクリック）。
6. 選択された項目は、現在の位置に移動します。

ヒント : *Magic xpa は、オブジェクトを名前ではなく番号で参照しているため、移動オプションを使用して移動することは何の問題もありません。Magic xpa は新しい位置を示すため参照情報を変更し、背後で様々なオブジェクトとの関連性を保持するために内部番号を使用しています。*

ヒント : DSOURCE または PROG *リテラルを使用しない場合は、問題がでる場合があります。このようなプログラムを使用している可能性がある場合は、（特に* **DbDel()** *関数などを）テキスト検索機能を利用してチェックし、修正するようにしてください。*

入力行を他の行の内容に置き換えるには

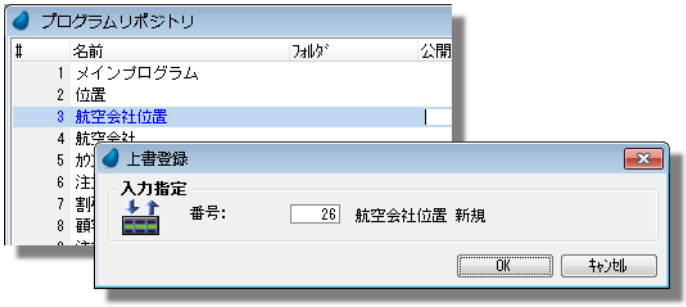
Magic xpa における項目は、これらの連番にもとづいて参照されます。例えば、以下のようなプログラムを呼び出すようにしたとします。



プログラム **#3** の航空会社位置プログラムをプログラム **#26** に移動した場合、何の問題も起きません。Magic xpa は、自動的に起動先を **#26** のプログラムに変更します。

しかし、作成したばかりの新バージョンの航空会社位置プログラムを呼び出すようにしたいと考えた場合、プログラム **#3** を新しいプログラムに置き換える必要があります。

行を置き換える



1. 差し替えたいプログラムの行にカーソルを移動します（この場合は、行 **#3**）。
2. ***Ctrl+Shift+O*** を押下します（または、**編集→登録→上書登録**）。
3. 置き換える項目の行番号（この場合 **#26**）を入力するか、**ズーム**して一覧から選択してください。
4. ***Enter*** を押下します（または OK を**クリック**）。

上記の操作によって、行 **#11** は新しいプログラム（この場合、航空会社位置 新規）に置き換えられます。

ヒント： *上記の操作によって、現在作業中のオブジェクトをそのままにした状態で必要に応じて実際に利用するオブジェクトを変更することができます。複写機能を使用して作業中のオブジェクトをコピーした場合は、後で混乱しないようにコピーされたオブジェクトの名前を変更しておきます。必要であれば、このような方法でいくつかの作業用コピーを取っておき、容易にそれらを比較し、各バージョンに切り替えるために上書き機能を使用することができます。この場合は、開発中のバージョンに戻すため、ロールバック用のオブジェクトも準備しておく必要があります。*

ペイン間の移動を行うには

作業中は、通常 1 つ以上のパレットがオープンされます。これらのパレット間の移動は、複数の方法で行うことができます。

- 使用するパレットを**クリック**します。
- ***Ctrl+Tab*** を使用してパレット間でフォーカスを移動します。
- パレットを表示させるには、以下のキーを使用します。

  Alt+F1（**表示→ナビゲータ**）

  Alt+F2（**表示→特性シート**、***Alt+Enter*** も可）

  Alt+F3（**表示→チェック結果**）

Ctrl+Tab は、***Alt*** とは少し動作が異なります。***Ctrl+Tab*** の場合、結合されたパレットは 1 つのウィンドウとして動作しますが、***Alt*** は他のウィンドウによって隠されていてもパレットが開きます。また、**タスク特性**ダイアログや**タスクエディタ**のようにタブを持つウィンドウを開いた場合、***Ctrl+Tab*** は、タブの切替として動作します。

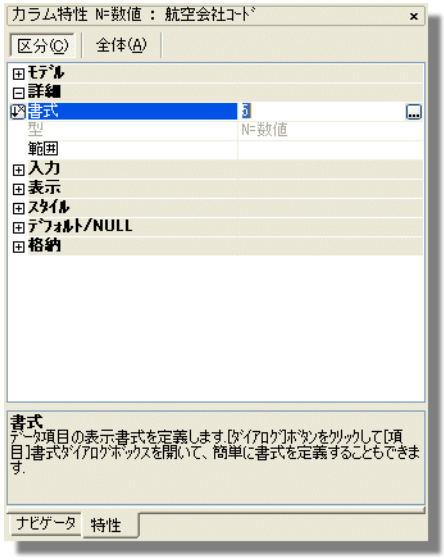
参照： 「パレットを分離するには」（3 ページ）

1 つのセクションを表示している特性シートを保持するには

特性シートが開いた場合、デフォルトではすべてのセクションは展開されて表示され、全ての有効な特性値を参照することができます。しかし、よりコンパクトに表示させたい場合があるかもしれません。

特性シートの表示を変更する

1. 現在のプロジェクトを閉じます。
2. **単一拡張パレット**特性（**オプション→設定→動作環境→動作設定**）を選択します。
3. **単一拡張パレット**特性を *Yes* に設定します。



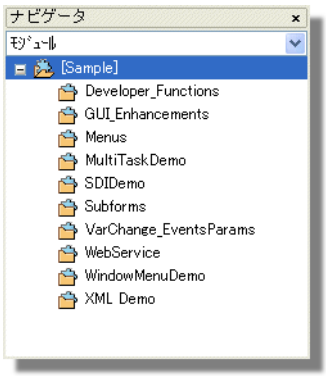
この設定によって、1 つのセクションを開くと、以前に開いていたセクションが縮小表示されます。また、新しい項目をクリックすると、すべてのセクションは縮小表示されます。

全体タブで表示されたいる特性は、この設定に影響しません。

プロジェクトを切り替えるには

複数のプロジェクトが、必ずしも互いにコンポーネントという関係でなくても、グループとして一緒に作業することがあるかもしれません。

ナビゲータペインの*モジュール*を選択することで、これらのグループに簡単にアクセスすることができます。



以下の例では、Sample は 10 のモジュールを持っています。モジュールがオープンされている場合、アイコン名に括弧が設定されます。各モジュールは、Magic xpa のプロジェクトを表しており、これらはどこにでも位置付けることができます。

プロジェクトを切り替える

1. ナビゲータペインに移動します。
2. *モジュール*を選択します。
3. 開きたいモジュールにカーソルを移動します。
4. *ダブルクリック*するか、*F5*（ズーム）を押下します。

そのモジュールに対応するプロジェクトが自動的にオープンされ、現在のプロジェクトがクローズされます。その際、保存されていない変更内容が保存されます。

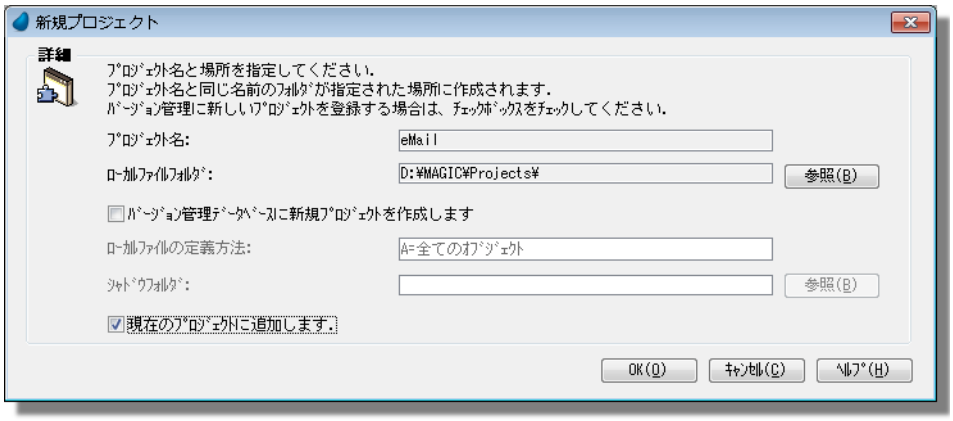
モジュールを追加する

必要条件： バージョン管理を使用している場合、最初にリポジトリをチェックアウトする必要があります。

1. プルダウンメニューからプロジェクト→モジュール追加を選択します。
2. プロジェクトを開くダイアログが表示されます。追加したいプロジェクトファイルを選択します。

追加されたプロジェクトは、モジュールペインに表示されます。

また、プロジェクトを新規作成する場合、**現在のプロジェクトに追加します。**のチェックボックスをチェックすることで自動的に現在のプロジェクトのモジュールペインに追加されます。



必要条件： バージョン管理を使用している場合、最初にリポジトリがチェックアウトされているかどうかを確認してください。

モジュールを削除する

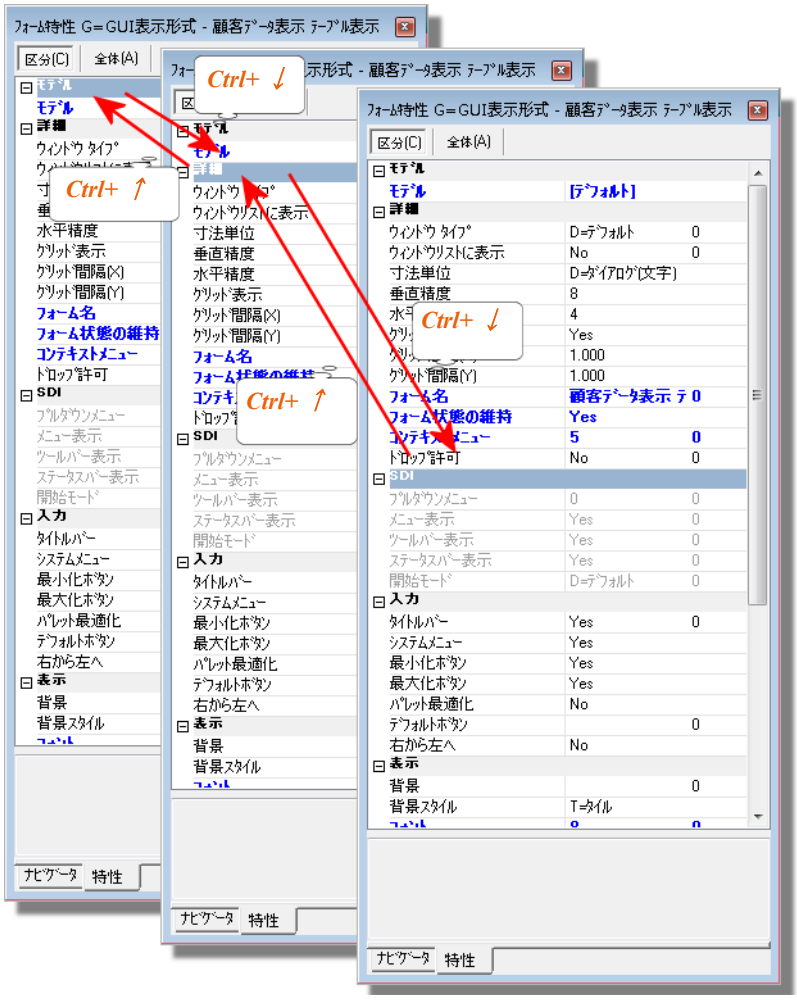
必要条件： モジュールを削除する前に、モジュールの上位にカーソルを置く必要があります。現在開いているモジュールや、モジュールの上位のオブジェクトを削除することはできません。また、バージョン管理を使用している場合、最初にリポジトリをチェックアウトとする必要があります。

1. 削除したいモジュールのノード上にカーソルを置きます。
2. *F3* を押下します（または、編集→行削除）。
3. 削除確認ダイアログではいをクリックします。

ノードは、現在のモジュールツリーから削除されるます。ただし、プロジェクトは削除されません。再度、モジュールとして追加することができます。

参照： 「最近使用したプロジェクトを迅速に開くには」（9 ページ）

特性シートのセクション表示を切り替えるには



特性シートのセクション間を移動する：

特性シートのセクション間を移動するには、以下の３つの方法があります。

- マウスで該当セクションをクリックします。
- キーボードで *Ctrl+ ↑*（上方向のセクションに移動）、または *Ctrl+ ↓*（下方向のセクションに移動）を押下します。
- 編集→ツリー→前のセクション（上方向のセクションに移動）、または編集→ツリー→次のセクション（下方向のセクションに移動）

特性シートのノードを拡張表示させる：

特性シートのノード表示を拡張表示させるには、以下の３つの方法があります。

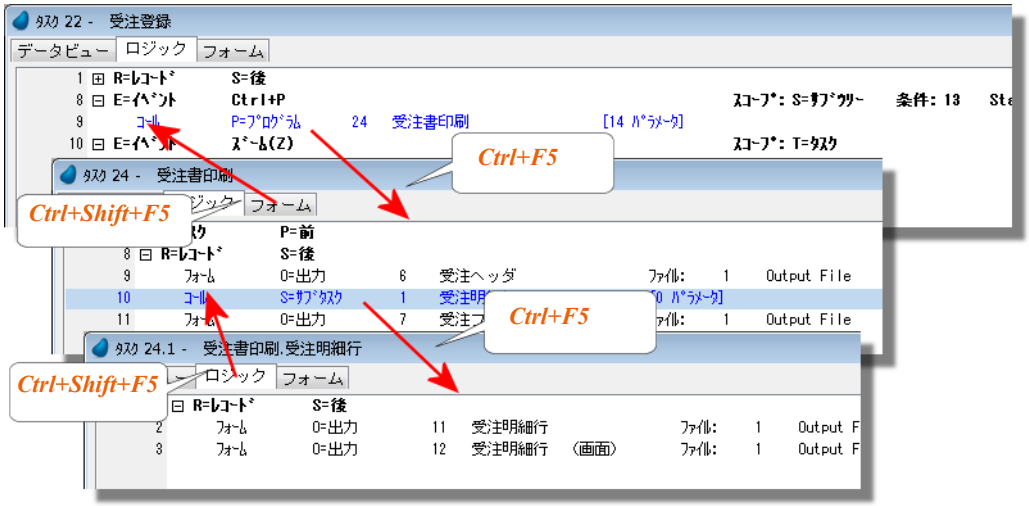
- マウスで該当するセクション名の前に表示されている⊟アイコンをクリックします。
- キーボードで *Ctrl+Plus*（+）を押下します。
- 編集→ツリー→サブツリー展開

特性シートのノードを縮小表示させる：

特性シートのノード表示を縮小表示させるには、以下の３つの方法があります。

- マウスで該当するセクション名の前に表示されている⊞アイコンをクリックします。
- キーボードで *Ctrl+Minus*（-）を押下します。
- 編集→ツリー→サブツリー縮小

呼び出されるプログラムに移動するには



プログラムの編集中に、呼び出されるプログラムまたはサブタスクに簡単に移動するための、ショートカットを使用することができます。

1. カーソルを**コール**処理コマンドの行に移動します。
2. ***Ctrl+F5*** を押下するか、コンテキストメニューから**オプション→オブジェクトに移動**を選択します。呼び出すプログラムまたは、タスクが開きます。

編集処理の終了時に、呼び出し元のプログラムまたは、タスクに戻りたい場合は、***Ctrl+Shift+F5*** を押下するか、**オプション→オブジェクトから戻る**を選択します。呼び出されたプログラムまたはタスクが閉じ、前のプログラムまたはタスクに戻ります。

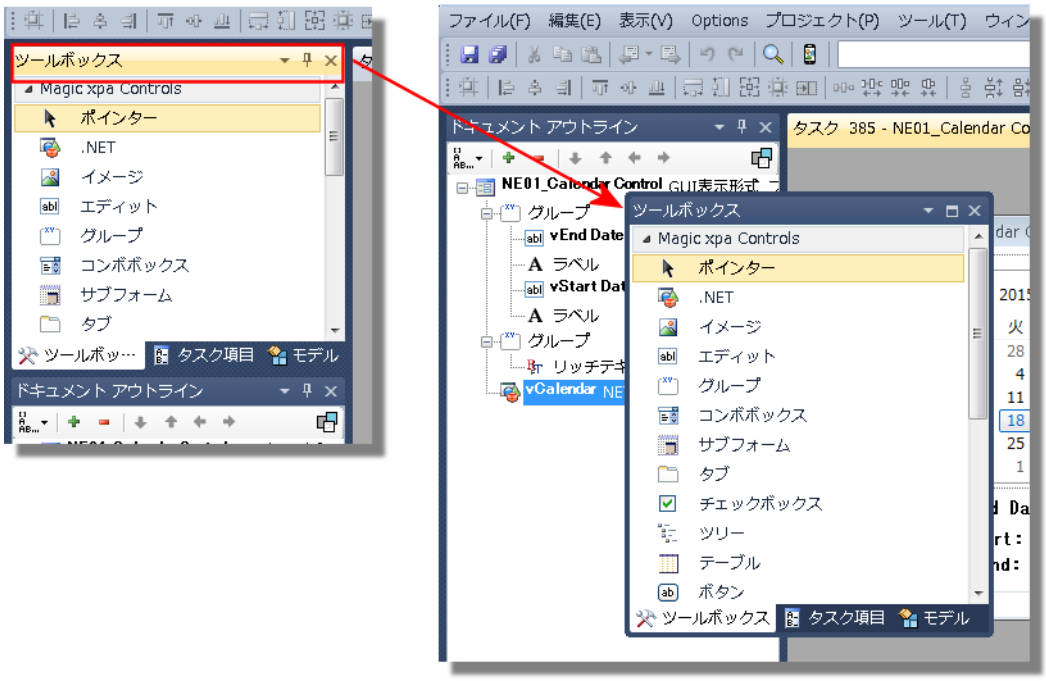
注： この操作は。普通のズーム処理とは少し異なります。*コール*処理コマンド上での様々なフィールドから *F5* を押下した場合、プログラムまたはタスクを選択するためのウインドウを開いたり、パラメータを設定したり、条件の設定を行います。オブジェクトに移動は、これとは異なり、新しくプログラムやタスクを開くことができます。

フォームデザイナ上でペインウィンドウを移動するには

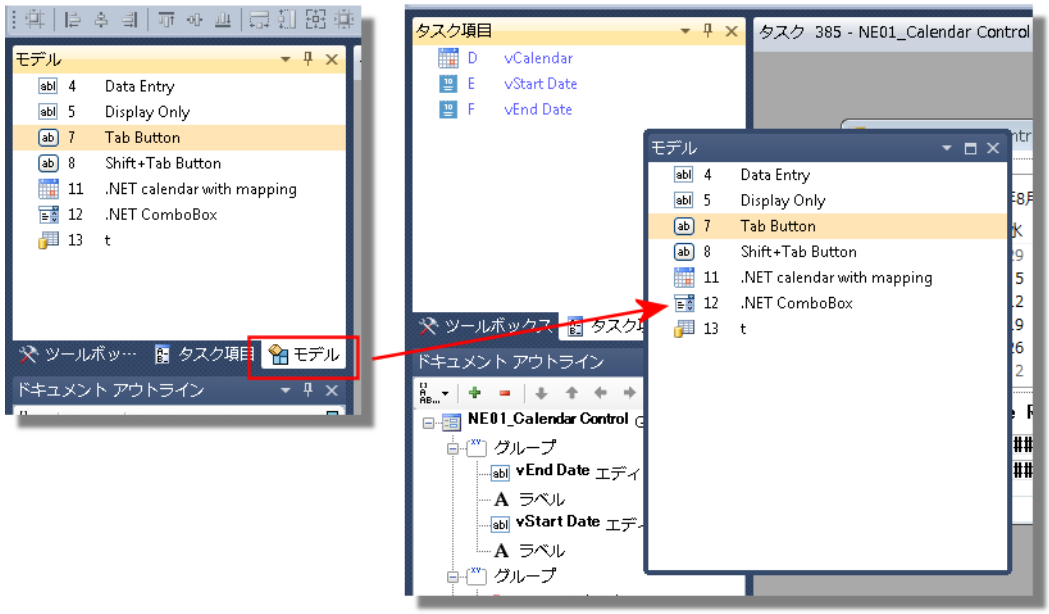
Magic xpa 3.0 より表示用フォームは、フォームデザイナで編集するようになりました、これは、Visual Studio のようなインタフェースを持ち、操作用のペインウィンドウをドッキングさせたりフローティングにすることができます。

ドッキングペインをフローティングする

まとめてフローティングするには、ペインの上部のタイトル部分をクリックしてドラッグします。



特定のタブだけフローティングするには、タブ部分をクリックしてドラッグします。



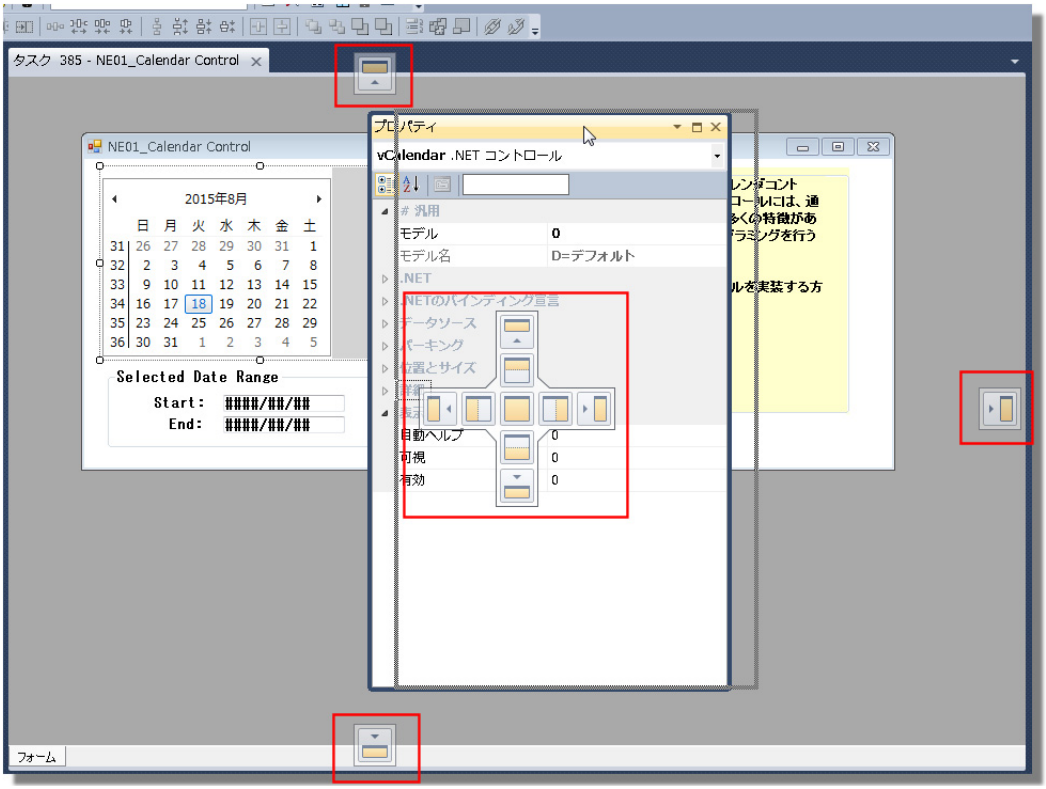
注： 移動したいタブを開いた状態で、タイトルかタブ上で右クリックしてコンテキストメニューを開きフローティングを選択しても同じように操作できます。

フォームデザイナ上でペインウィンドウをドッキングするには

Magic xpa 3.0 より表示用フォームは、フォームデザイナで編集するようになりました、これは、Visual Studio のようなインタフェースを持ち、操作用のペインウィンドウをドッキングさせたりフローティングにすることができます。

フローティングペインをウィンドウの端にドッキングする

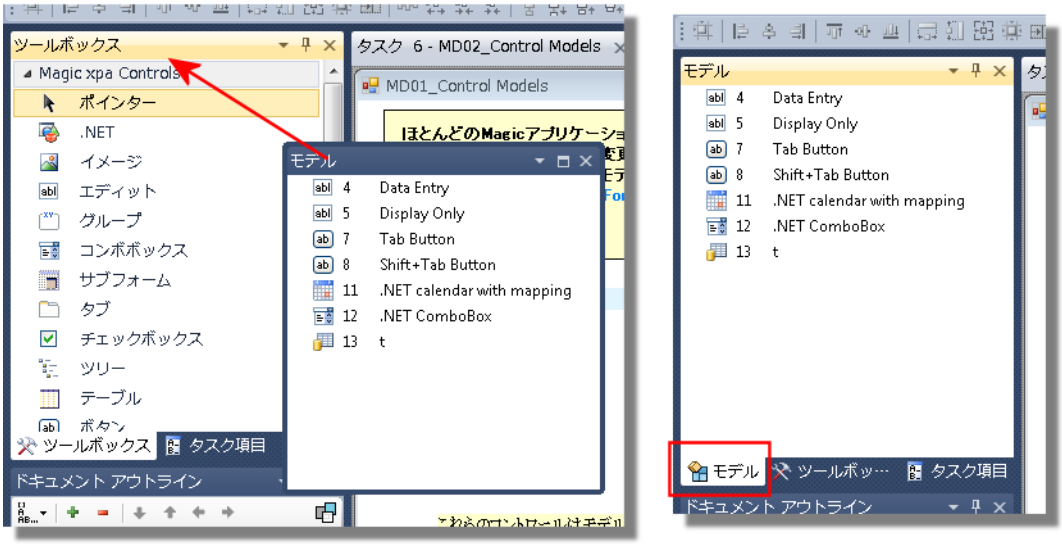
フローティングペインをクリックしてドラッグすると矢印の入った「ボックス」がいくつか表示されます。これはマウスのドラッグ中にしか表示されないもの でマウスのボタンを離すと消えます。これらのボックスは、そこにドラッグするとその位置にドッキングすることを示すためのしるしとなります。



ドラッグしたまま任意のウィンドウの端にドロップするとペインがドッキングされます。

フローティングペインを別のペインにドッキングする

フローティングペインをクリックして別のペインにドラッグすると一つのウィンドウとしてドッキングされます。



また、ドッキングしたいウィンドウのタイトル上で右クリックしてコンテキストメニューを開きドッキングを選択すると、デフォルトの位置にドッキングされます。

フォームデザイナでツールバーをカスタマイズするには

フォームデザイナでは、フォームを編集するためのコマンドがツールバーにアイコン表示されます。このツールバーは、カスタマイズすることができます。

レイアウトツールバーを表示させるには

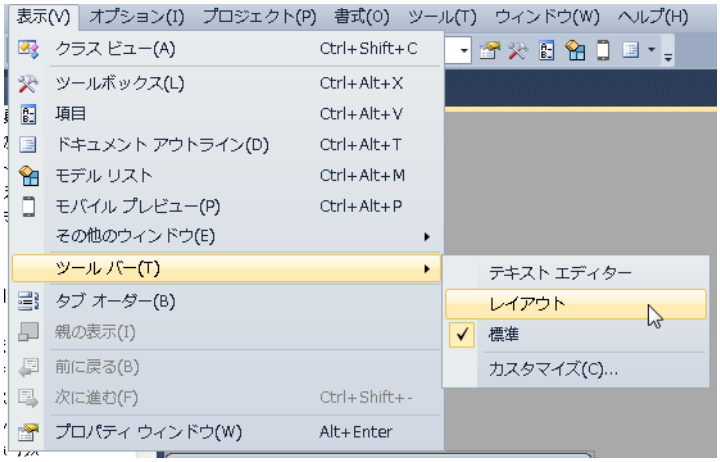
Magic xpa Studio Ver3 をインストールした直後は、フォームデザイナには、標準ツールバーしか表示されない状態です。

フォームの編集を行うには、プルダウンメニューを使用して行うこともできますが、レイアウトツールバーを表示させた方が便利です。

レイアウトツールバーを表示させるには以下のようにします。

プルダウンメニューで表示させる

表示メニューから**ツールバー**を選択するとツールバーに表示させたい項目を選択できるメニューが表示されます。ここから、**レイアウト**を選択してチェックマークを表示させます。



ツールバー上で表示させる

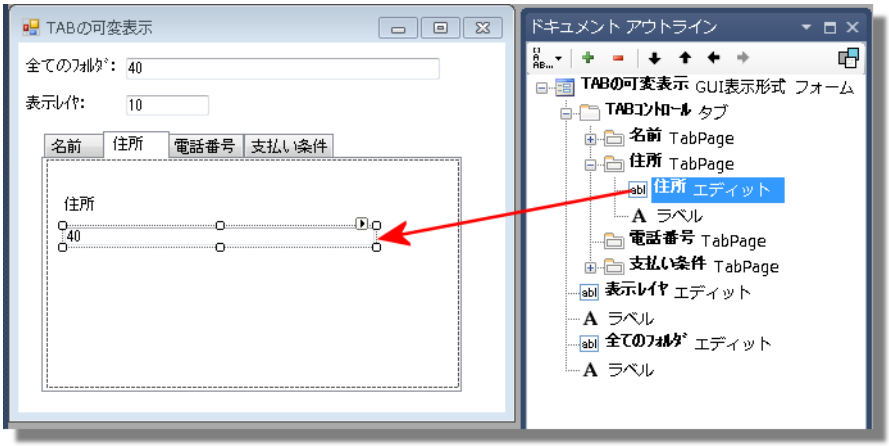
標準ツールバーの右側のアイコンが表示されていない場所で右クリックするとコンテキストメニューが表示されます。ここから、**レイアウト**を選択してチェックマークを表示させます。

フォームデザイナ上で特定のコントロールを選択するには

タブコントロール上のコントロールなどは、フォームデザイナ上で隠れている場合があります。このようなコントロールを編集する場合は、タブを切り替えて表示させる必要がありますが、ドキュメントアウトラインペインを使用することでことで簡単に特定のコントロールを指定することができます。

ドキュメントアウトラインでコントロールを指定する

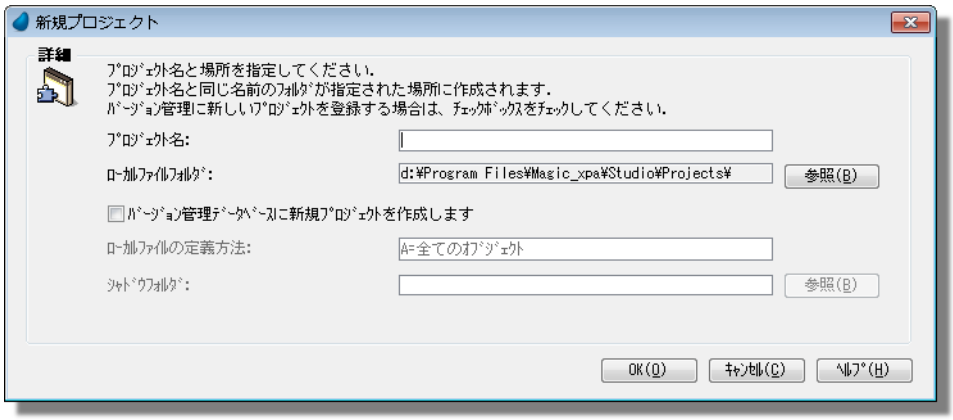
ドキュメントアウトラインペインには、フォーム上のコントロールが表示されています。コントロールは配置内容に基づいてツリー表示されています。表示されているコントロール名のノードをクリックするとフォーム上のコントロールが選択状態になります。



［このページは意図的に空白にしています。］

第 2 章： プロジェクトとアプリケーション

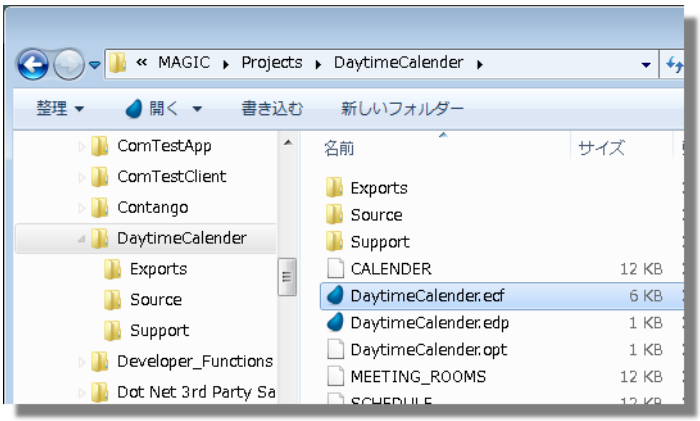
プロジェクトを新規作成するには



プロジェクトを新規作成する

1. プルダウンメニューから、ファイル→新規作成 を選択します。
2. **プロジェクト名**を入力します。
3. **位置**にプロジェクトファイルを作成するディレクトリを入力します。
4. OK をクリックします。

現在プロジェクトを開いている場合は、そのプロジェクトがクローズされ、新規プロジェクトが作成されます。



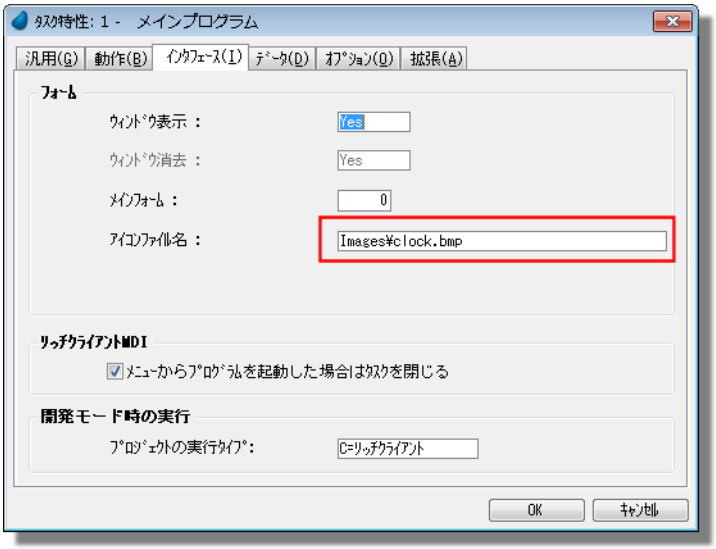
プロジェクトが作成されると、指定されたディレクトリ内にサブディレクトリが作成され、その中に関連するファイルが格納されます。プロジェクト名はディレクトリ名になり、プロジェクトファイルの名前になります。プロジェクトファイルの拡張子は、「.edp」になります。

プロジェクトが既にオープンされている場合、現在のプロジェクトに追加します。というチェックボックスが表示されます。チェックボックスを**チェック**すると作成されたプロジェクトは、既存のプロジェクトのモジュールとして追加されます。

参照： 「既存のプロジェクトをオープンするには」（34 ページ）

アプリケーションのアイコンを設定するには

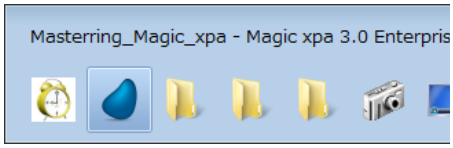
アプリケーションを開発する場合、アプリケーションに独自のアイコンを割り当てることがあります。このような場合は、以下で示す方法で設定することができます。



アプリケーションのアイコンを設定する

1. **メインプログラム**の**タスク特性**（*Ctrl+P*）を開きます。
2. **インタフェース**タブを選択します。
3. **アイコンファイル名**をから**ズーム**して表示させたいアイコンファイルを選択します。

アプリケーションを実行すると、ウィンドウの左上にアイコンが表示されます。また、タスクバーや *Alt+Tab* によってウィンドウを切り替える際にも表示されます。

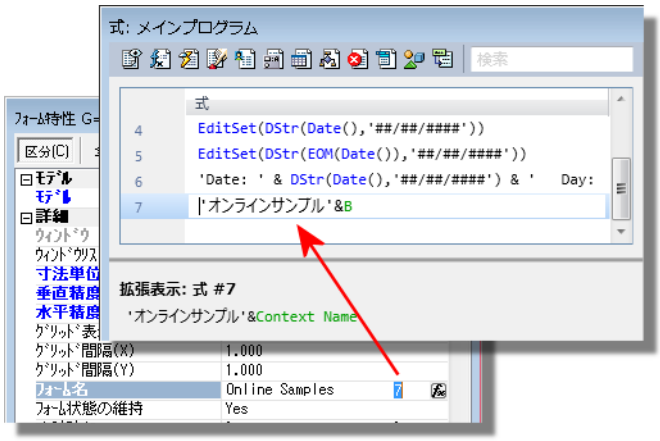
		
ウィンドウタイトルのアイコン	**システムトレイのアイコン**	**アプリケーション切り替え時の表示**

ヒント : *アプリケーションのユーザは、開発時とは異なるパスでセットアップする場合があるため、このような内部ファイルを指定する場合は、固定のパス名を使用しないでください。デフォルトのパスは、作業ディレクトリ（プロジェクトファイルが存在している場所）になるため、アイコンファイルはそこに置くことか、相対的なサブディレクトリを指定してください。*

参照 : 「プロジェクトのディレクトリのファイルを読み書きするには」（31 ページ）

アプリケーションのタイトルを指定するには

ウィンドウに表示されるアイコンの隣に、アプリケーションの名前を示すテキストを表示させることができます。この設定は、メインプログラムのフォーム名として指定します。



アプリケーションのタイトルを設定する

1. メインプログラムのフォームタブを選択します。
2. メインフォームのフォーム特性（*Alt+Enter*）を開きます。
3. **フォーム名**特性に表示させたい名前を設定します。

以下のように式で定義することで、実行時に動的に表示させることもできます。

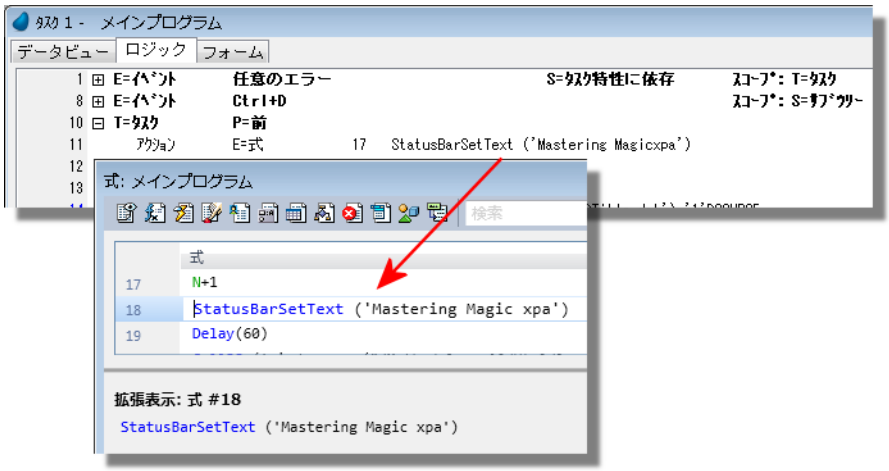
4. **式**でズームして式エディタに表示させたいタイトルを定義します。

xpa Stiudio では、**フォーム名**特性に定義した文字列が表示されますが、実行時は、式が評価され、その結果が表示されます。

ステータスバーにタイトルを表示する

アプリケーション起動時に、ステータスバーにアプリケーションのタイトルを表示させることもできます。

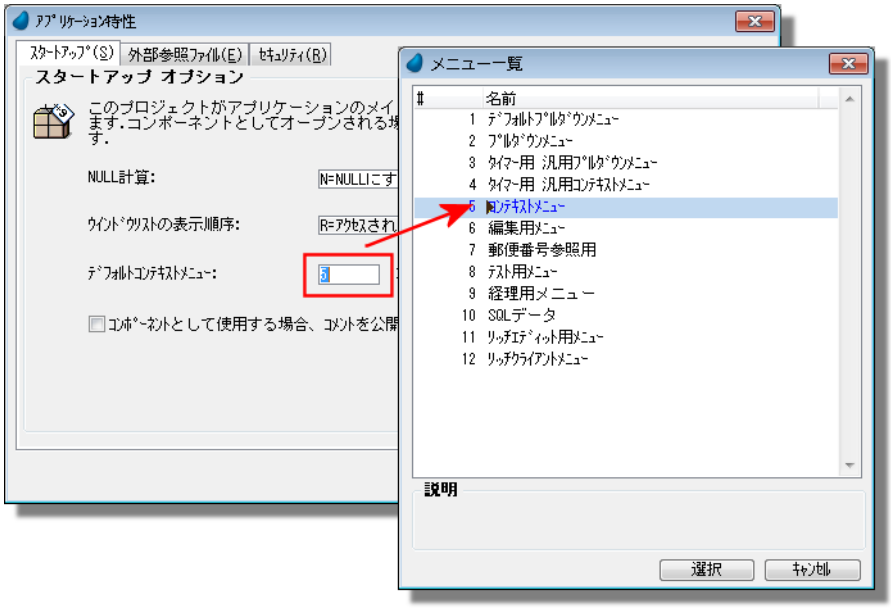
1. メインプログラムのロジックエディタを開きます。
2. タスク前のロジックユニットを作成し、１行追加します。
3. アクション処理コマンドを定義し、以下の式を実行するように定義します。
   StatusBarSetText (*アプリケーション名*)



アプリケーションを実行すると、ステータスバーにタイトルが表示されます。

アプリケーションにコンテキストメニューを設定するには

オンラインタスク用にコンテキストメニューを定義する場合、特定のプログラムに設定する方法とは別に、アプリケーション全体で有効なデフォルトのコンテキストメニューを設定することもできます。

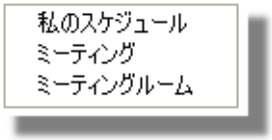


設定するメニューは、あらかじめ作成しておく必要があります。

アプリケーションにコンテキストメニューを設定する

1. アプリケーション特性（ファイル→アプリケーション特性、または *Ctrl+Shift+P*）を開きます。
2. **システムコンテキストメニュー**特性でズーム（*F5*、またはダブルクリック）を実行します。
3. 設定するコンテキストメニューにカーソルを置きます。
4. OK をクリックして、アプリケーション特性に戻ります。
5. OK をクリックして、ダイアログを閉じます。

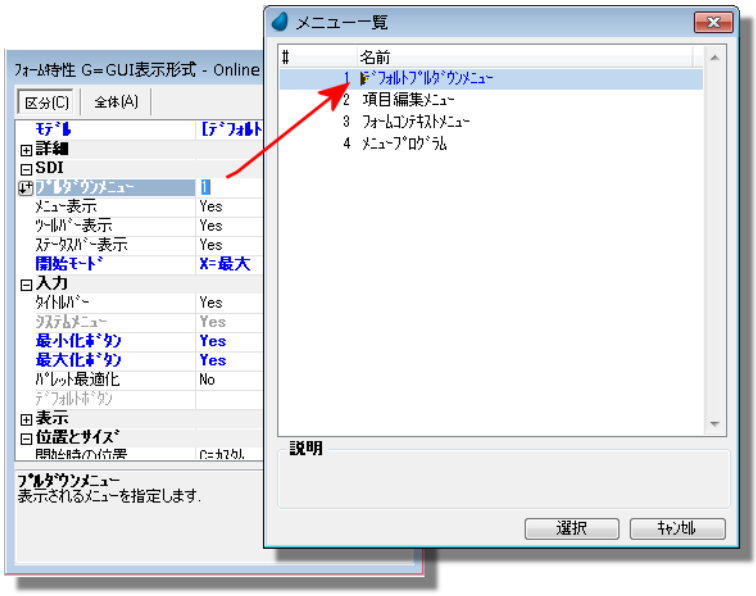
アプリケーションを実行し、ユーザがマウスの右クリックを行うと、設定されたメニューが表示されます。



参照： 第 5 章：「コントロールにデフォルトのコンテキストメニューを設定するには」（135 ページ）
第 5 章：「個別のコントロールにコンテキストメニューを設定するには」（136 ページ）

アプリケーションにプルダウンメニューを設定するには

Magic xpa は通常、デフォルトのプルダウンメニューが表示されますが、ここには基本的な編集用のコマンドのみ表示されています。アプリケーションを使用するユーザ用に、専用のプルダウンメニューを表示させることができます。

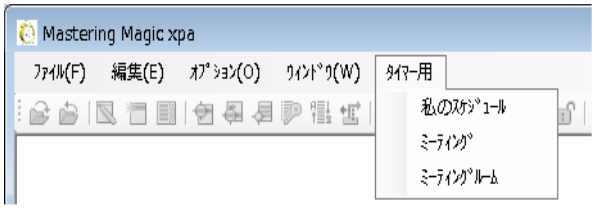


必要条件： 設定するメニューは、あらかじめ作成しておく必要があります。

アプリケーションにプルダウンメニューを設定する

1. メインプログラムのフォームタブを選択します。
2. メインフォームのフォーム特性（*Alt+Enter*）を開きます。
3. **プルダウンメニュー**特性でズーム（*F5*、またはダブルクリック）を実行し、表示させたいメニューを設定します。
4. または、式からズームして実行時に表示させたいメニュー番号として評価される式を定義します。

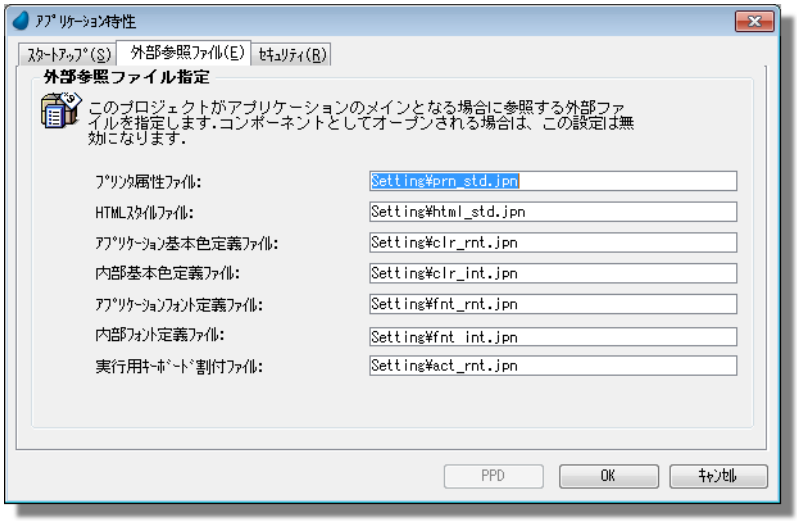
アプリケーションを実行すると、指定されたプルダウンメニューが表示されます。



注： メニューリポジトリ上では、*コンテキスト*と*プルダウン*の違いはありません。必要であれば、両方に同じメニューを使用することができます。

アプリケーション用に基本色、フォント、キーボード割付を設定するには

Magic xpa は、デフォルトではインストール時に転送された（基本色、フォント、キーボード割付）定義ファイルを使用します。しかし、アプリケーション用にこれらのファイルをカスタマイズする必要があります。各アプリケーションは、独自の定義ファイルを使用することができます。これらのファイルは、個別のテキストファイルのため、ユーザによって任意にカスタマイズすることができます。



必要条件： 最初に特定のディレクトリ内に定義ファイルをコピーしておく必要があります。

アプリケーション用の定義ファイルを設定する

1. アプリケーション特性（ファイル→アプリケーション特性、または *Ctrl+Shift+P*）を開きます。
2. 変更したい特性値（基本色、フォント、キーボード割付の各定義ファイル）にカーソルを移動します。
3. ファイル名を入力します。

各ファイル名の入力欄でズームすると、アプリケーション用に定義内容を変更するためのダイアログが表示されます。

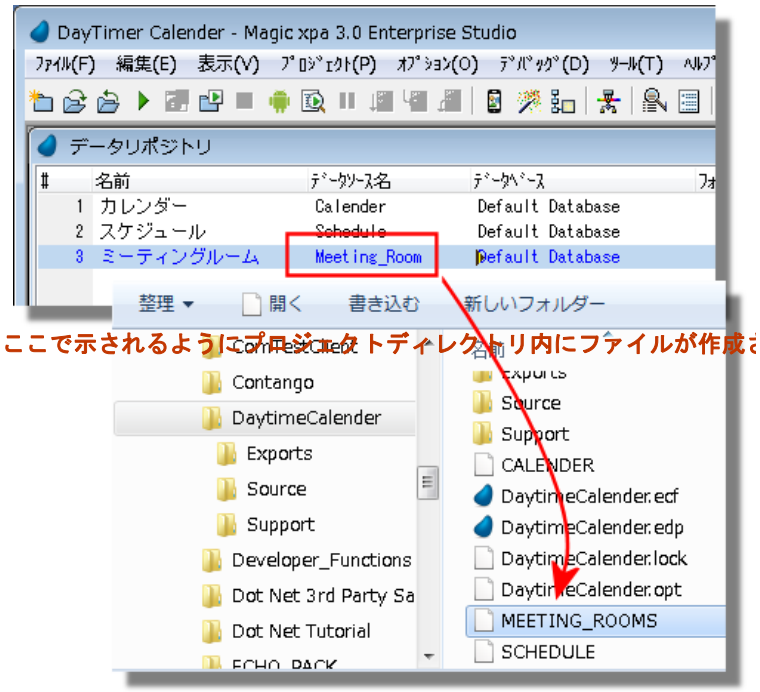
注： フルパスまたは論理名が指定されない場合、Magic エンジンが存在するディレクトリ（**%EngineDir%**）をアクセスします。アプリケーションフォルダ内のファイルをアクセスさせる場合は、**%WorkingDir%** を指定するようにする必要があります。

参照： 「プロジェクトのディレクトリのファイルを読み書きするには」（31 ページ）

プロジェクトのディレクトリのファイルを読み書きするには

プロジェクトの実行中、デフォルトディレクトリはプロジェクトディレクトリ（.edp または .ecf ファイルのが配置されているディレクトリ）です。従って、アプリケーションによってファイルを作成する際、明示的なパスを指定しない場合、プロジェクトディレクトリ内に作成されることになります。

以下のような DayTimerCalender アプリケーションに 3 つの DB テーブルがある場合を例に説明します。



ここで示されるようにプロジェクトディレクトリ内にファイルが作成さ

入出力ファイルなどを使用してプログラムから他のファイルを参照する場合も同じように動作します。プロジェクトディレクトリ以下にファイルを置くことで、相対パスを使用してこれらのファイルにアクセスすることができます。

アプリケーションが複数のプロジェクトによって構成されている場合、プロジェクトディレクトリは一番上位のプロジェクトのプロジェクトディレクトリによって決まります。従って、DayTimerCalender プロジェクトがプロジェクトディレクトリに書き込まれたコンポーネントを呼び出す場合、これらのファイルは DayTimeCalendar ディレクトリ内にあります。

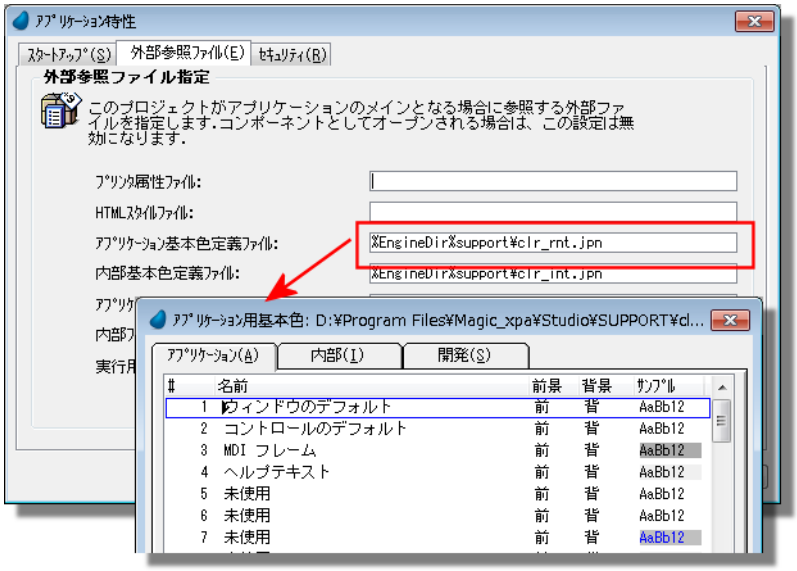
明示的にプロジェクトディレクトリを参照したい場合は、論理名（%WorkingDir%）を使用してください。

Magic ディレクトリのファイルを読み書きするには

*Magic ディレクトリ*は、Magic エンジンが存在するディレクトリを表しています。インストール時のデフォルトディレクトリは、Windows の上の Programe Files ディレクトリの下に作成されます。

Magic xpa は、データソースや入出力ファイルなどのプロジェクト固有のファイルと一般的なエンジンファイルを区別します。例えば、**動作環境**ダイアログの**外部参照**タブ（**オプション→設定→動作環境**）に定義されている基本色、フォルト、キーボード割付の各定義ファイルは、すべて Magic ディレクトリに配置されます。

アプリケーション特性内で、作業ディレクトリの代わりに Magic ディレクトリを参照するようにするには、論理名 %EngineDir% を使用することができます。例えば、DayTimerCalender プロジェクト内で、Magic xpa と一緒にインストールされた**基本色定義ファイル**にアクセスする必要がある場合は以下のようにします。

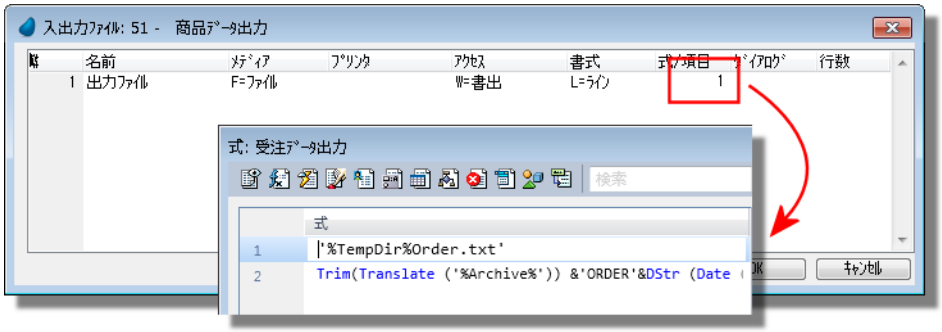


相対パスの前に %EngineDir% を付加することで Magic xpa のインストールディレクトリ内の clr_rnt.jpn ファイルを指定することができます。

システム一時ディレクトリ内のファイルを読み書きするには

Windows™ には、セットアップ時に作成される一時ディレクトリがあります（デフォルトでは、C:¥Documents and Settings の下に作成されます）。これは、多くのアプリケーションが***スクラッチファイル***を書込むために使用されます。

例えば、他の製品にデータをインポートする目的で作成する中間ファイルの作成先などに使用できます。このディレクトリは、論理名 %TempDir% を使用することでプログラムからアクセスすることができます。



この例では、Excel ファイルから読み込み込まれたデータをもとにテキストファイルを作成しています。パス名を指定しない場合、一時ファイルは作業ディレクトリに作成されます。ここで論理名 %TempDir% を追加することで、システム一時ディレクトリに作成するように指定することができます。

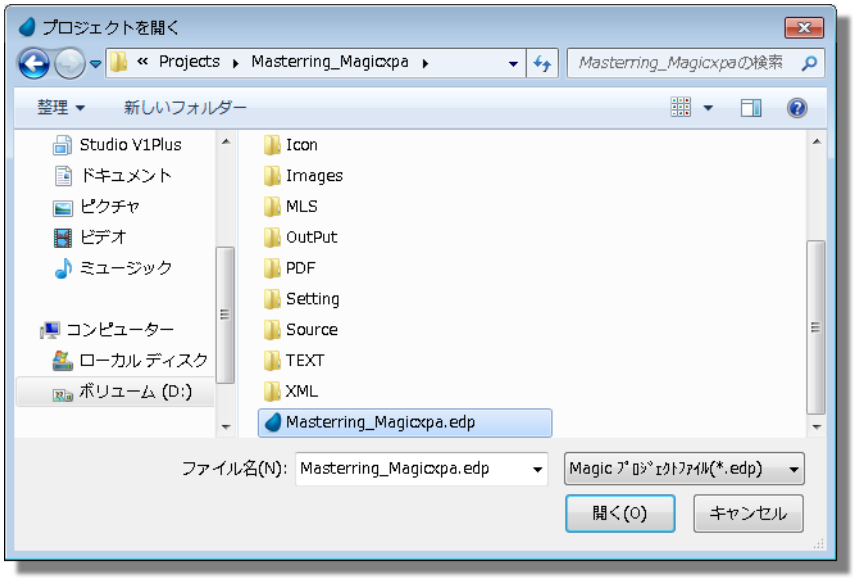
既存のプロジェクトをオープンするには

アプリケーションを 1 つ以上のプロジェクトで構成する場合があります。この場合、いくつかのロジックがコンポーネントを使用してカプセル化されたり、他のサーバ上で動作するように作成されていることもあります。**Magic xpa Studio** 内でプロジェクト間での移動を行ったり、Windows からプロジェクトをオープンすることができます。既存のプロジェクトをオープンする方法として、以下の 3 つの基本的な方法があります。

プロジェクトを開くダイアログを使用する

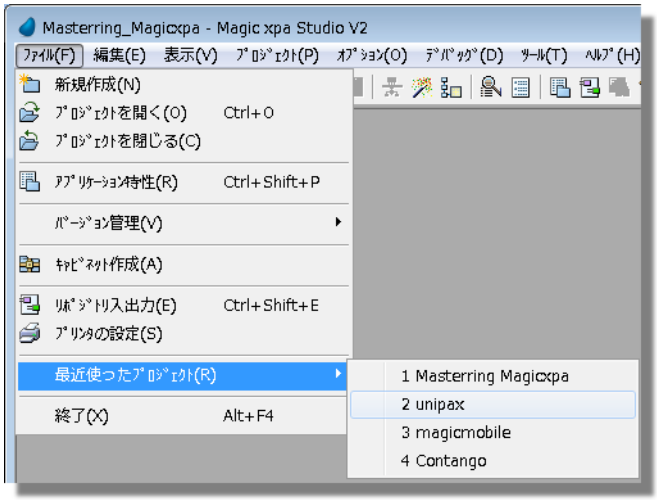
1. プルダウンメニューから**ファイル→プロジェクトを開く**（または、***Ctrl+O*** を押下）を選択します。
2. **プロジェクトを開く**ダイアログが表示されます。プロジェクト（***.edp***）ファイルを選択します。
3. 開くを**クリック**します。

選択されたプロジェクトがオープンされます。



最近使ったプロジェクトを開く

現在のプロジェクトをクローズする必要はありません。プロジェクトのオープン時に現在のプロジェクトは保存、クローズされます。***Ctrl+O***（または、**ファイル→プロジェクトを開く**）を押下してネットワーク上のプロジェクトをオープンすることができます。しかし、以前作業していたプロジェクトを簡単にオープンすることができます。

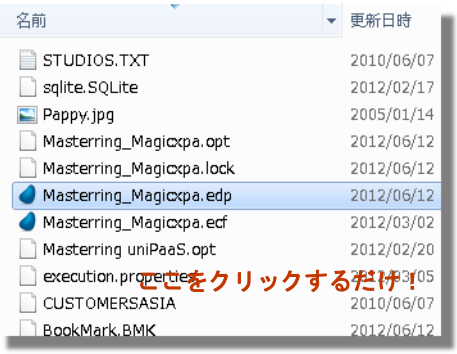


1. プルダウンメニューから**ファイル→最近使ったプロジェクトを開く**を選択します。
2. オープンしたいプロジェクトを探します。表示されているプロジェクト名にカーソルを置くとステータスバーにプロジェクトファイル名がフルパスで表示されるので、この内容も参考にしてください。
3. オープンしたいプロジェクト名を**クリック**します。

現在オープンしているプロジェクトは保存され、選択されたプロジェクトがオープンされます。

参照： 第 1 章：「最近開いたプロジェクトの表示数を変更するには」（10 ページ）

プロジェクトファイル（*.edp*）を直接アクセスする



1. Windows エクスプローラでオープンしたいプロジェクトファイル（.edp）にカーソルを置きます。
2. プロジェクトファイルをクリックします。

プロジェクトがオープンされます。この方法は他と異なり、オープン中のプロジェクトはそのままの状態で、新しく **Magic xpa Studio** のインスタンスが起動されます。プロジェクトファイルを Windows のショートカットとして登録することでこの方法を利用することもできます。

この方法は、Magic xpa のインストール処理によって、拡張子 ***.edp*** が Magic xpa のプロジェクトファイルに関連付けられることで実現しています。

参照： 第 7 章：「アプリケーション用のショートカットを作成するには」（175 ページ）

別のプロジェクトにオブジェクトを転送するには

必要条件： 他のオブジェクトを参照していないオブジェクトであれば、簡単にコピーすることができます。しかし、他のオブジェクトを参照している場合、コピー先のプロジェクトファイルにも同じ参照オブジェクトが定義されている必要があります。

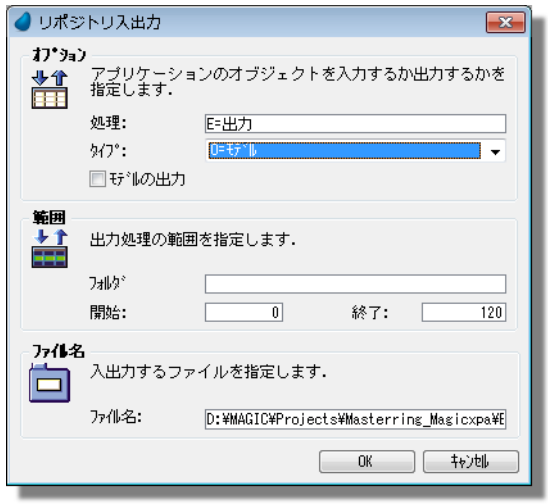
例えば、**プログラム #21** をプロジェクト A からプロジェクト B にコピーする場合を考えます。**プログラム #21** は、**モデル #3、#18、**および **#24** を使用し、また、**プログラム #46** 呼び出しているものとします。

プログラム #21 をプロジェクト B にインポートする場合、**モデル #3、#18、**および **#24** は 2 つのプロジェクトにおいて同じ場所に定義されている必要があります。**プログラム #46** も同じです。

この例の場合、一方のプロジェクトからモデルをエクスポートし、別のプロジェクトにインポートすることから始める必要があります。

オブジェクトを出力する

1. リポジトリ入出力ダイアログ（ファイル→リポジトリ入出力、または ***Ctrl+Shift+E***）を開きます。
2. **処理**で ***E= 出力***を選択します。
3. **タイプ**で出力したいリポジトリ（モデル、データソース、プログラム、ヘルプ、権利、メニュー、コンポーネント、アプリケーション特性、プロジェクト全体）を選択します。
4. 必要であれば、出力対象のフォルダや出力範囲を指定して、出力範囲を絞り込むことができます。出力範囲指定（開始、終了）では、ズームすることで一覧から選択できます。デフォルトでは、指定されたタイプのオブジェクトがすべて出力されます。
5. **ファイル名**に出力したいファイル名を入力します。デフォルトは、作業ディレクトリ内の ***Exports*** サブディレクトリ内に出力されます。
6. OK をクリックします。



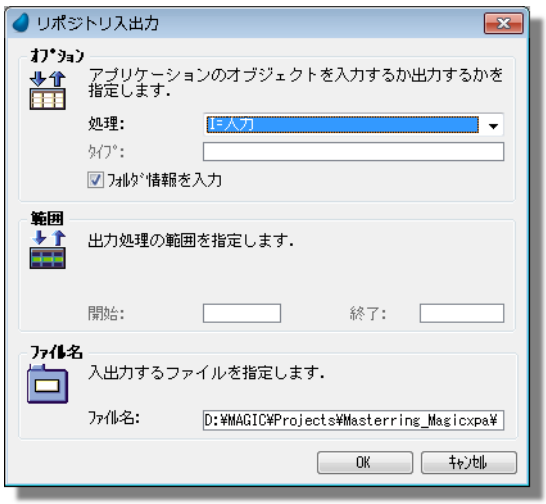
指定したフォルダ内にファイルが作成されます。例えば、モデルを出力した場合 ***Models.xml*** が作成されます。

次に出力したモデルを新規に作成したプロジェクトに入力します。

オブジェクトを入力する

1. リポジトリ入出力ダイアログ（ファイル→リポジトリ入出力、または ***Ctrl+Shift+E***）を開きます。
2. **処理**で ***I= 入力***を選択します。
3. 入力する XML ファイル名を指定します。
4. OK をクリックします。

出力したオブジェクトと同じ内容が、現在のプロジェクトに追加されます。これらは、常に現在のリポジトリ内に定義されているオブジェクトの下に追加されます。



ヒント：リポジトリ入出力機能は、通常、同じプロジェクトの異なるバージョン間でプログラムを移動したり、（モデルなどを）空のプロジェクトにコピーしたり、簡単な汎用プログラムをコピーする場合に使用されます。通常、複数のプロジェクト間でオブジェクトを共有する必要がある場合は、コンポーネントを使用してください。コンポーネントは、容易に共有可能で再利用可能なオブジェクトです。

参照： 第 15 章：「プロジェクト間で Magic xpa のオブジェクトを再利用するには」（357 ページ）

第 3 章 : モデル

再利用可能なインタフェースオブジェクトを定義するには

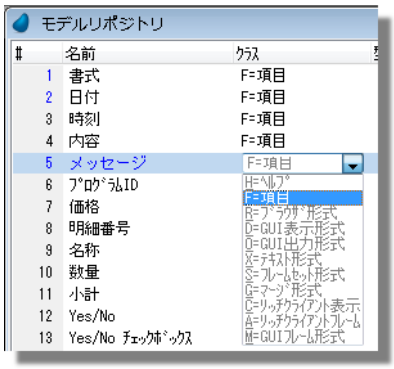
プログラミングにおける難問の 1 つは、複数のウィンドウやブラウザ、あるいは帳票の書式を一貫性のあるものとして維持することです。テキストベースの画面表示は使用されなくなってきているためです。GUI 形式の画面や帳票を設計するには、数百のフォント、サイズ、および色から選択する必要があります。これらの選択作業を各オブジェクト毎に行うと、非常に工数がかかってしまい、更にこれらの一貫性を維持したり、変更したりすることは至難の業です。

Magic xpa はこのような作業を簡単に行うことができます。Magic xpa には、***モデル***という強力な機能が備わっています。モデルは、テキストやラジオボタン、テーブル、テーブルカラム、印刷フォーマット、Web ブラウザ画面、および Windows 互換ウィンドウなどを含んでおり、すべてのインタフェースオブジェクトのルック＆フィールを定義するために使用できます。

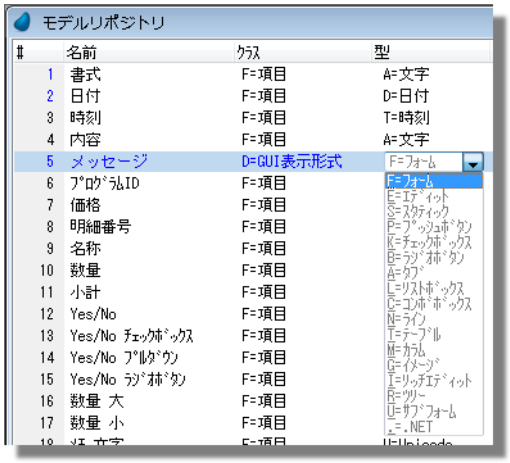
各オブジェクトの特性をモデルとして設定することで、容易に使用することができます。フォームを設計する場合は、オブジェクトに対応した使用したいモデルを選択するだけです。これらの書式を変更する場合は、モデルを変更することで自動的にそのモデルを使用するすべてのオブジェクトの書式が変更されます。

ここでは、メッセージ表示用の簡単な**項目モデル**を設定する例を挙げて説明しています。

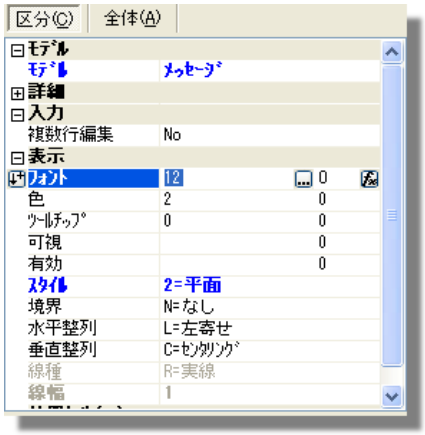
コントロールモデルを作成する



1. モデルリポジトリに移動し（*Shift+F1*）、アクセスしたい行に移動します。
2. *F4*（編集→行作成）を押下します。
3. モデルの**名前**を入力します（ここでは、メッセージ と入力されています）。
4. **クラス**を選択します。
   - ***GUI 表示形式***: 表示画面フォーム用
   - ***GUI 出力形式***: 帳票フォーム用
   - ***テキスト形式***: 他のアプリケーションに渡すためのテキストファイルやテキストプリンタのフォーム用
   - ***リッチクライアント表示形式***: リッチクライアント用



5. **型**を選択します。クラスの選択内容によってここに表示される内容は異なります。ここでは GUI 表示フォームの項目が一覧されています。例えば、**テキスト**コントロールを定義する場合は、***S= スタティック***を選択します。



6. 最後に特性を変更します。特性値の表示内容は、選択したクラスと型に依存します。
7. ここでは、他の表示より太字で表示させたいため、**フォント**特性の指定を変更します。また、**スタイル**特性をデフォルトの *3= 凸立体*から *2= 平面*に変更しています。

モデルの定義を行った後に、作成したコントロールにこのモデルを割り当てることで書式が自動的に設定されます。

参照： （*「プロジェクトのデータ項目とコントロールを標準化するには」（42 ページ）*）

再利用可能なデータオブジェクトを定義するには

大規模なアプリケーションを開発する上で重要な点の 1 つにデータ項目の一貫性を維持することがあります。例えば、住所用のデータ項目を考えてみます。このデータ長が、あるプログラムでは 30 桁に定義されており、別のプログラムでは 40 桁に定義されていることは問題が発生する元になります。また、起動元のレコード ID が 10 桁であるのに対し、起動先のプログラムのパラメータとして定義されたレコード ID が 8 桁になっていては不都合です。

また、項目の有効な値は一貫性があり、開発者が確認しやすいようにすべきです。例えば、ステータスコードが定義されている場合、開発者はそのコードの値が何を意味しているかを知っている必要があります。

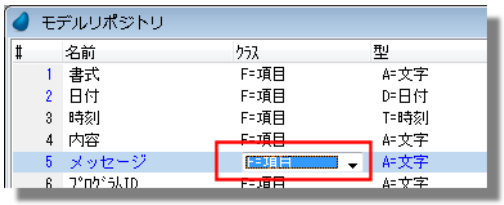
そして最も重要なことは、項目長を変更した場合、その項目を使用するすべてのデータソースやプログラムに反映される必要があることです。

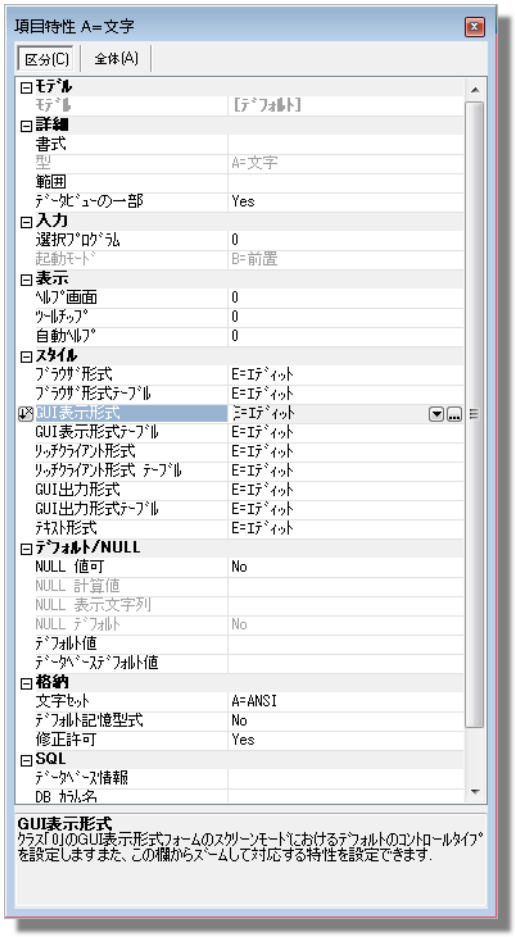
モデルを使用することで、Magic xpa はこれらのことを容易に行うことができます。データにモデルが割り当てられている場合、そのモデルはデータの項目長と有効な値を定義することができます。さらに、モデルは実際の DBMS での記憶形式や、カプセル化された選択プログラム、ヘルプ画面、自動ヘルプ、ツールチップなどを定義することができます。設定内容は、開発者が簡単にアクセスすることができるため、データのタイプを開発時に確認することができます。

モデルの設定は容易に変更することができます。ステータスコードを追加する必要がある場合、ステータスコード用のモデルを変更することで、ステータスコードを使用しているデータソースやプログラムは自動的に変更されます。さらに、変更することでロジックに影響することがないかどうかを確認するために、クロスリファレンス（*Ctrl+F*）ユーティリティを利用することでステータスコードを使用しているすべてのオブジェクトを検索することができます。

データを定義するモデルは、**項目**クラスを指定することで作成することができます。

項目モデルを作成する





1. モデルリポジトリに移動し（*Shift+F1*）、目的の行にカーソルを位置付けます。
2. *F4*（編集→行作成）を押下します。
3. モデルの**名前**を入力します（ここでは、ステータスコードです）。
4. **クラス**カラムで ***F= 項目***を選択します。
5. **型**を選択します。項目モデルでは、以下のデータ型が選択できます。文字、Unicode、数値、日付、時刻、BLOB、.Net など。
6. 次に、モデル特性（*Alt+Enter*）を開き、必要な特性を設定します。ここにはたくさんの選択項目があります。各特性値の内容は『リファレンスヘルプ』を参照してください。

   詳細 : このセクションでは項目の書式や長さを定義することができます。範囲特性に連続値や不連続値を指定することで、有効な値を指定することもできます。ここでは、1 桁の大文字で、4 つの範囲値が指定されています。

   入力 : 値を選択するために *F5* を押下するか、*ダブルクリック*（ズーム）することで起動されるプログラムを指定できます。

   表示 : ここでは、ヘルプ画面や、ツールチップ、ステータス行に表示される自動ヘルプを定義することができます。

   スタイル : このセクションでは、データが表示されるフォームごとにどのように表示されるかを指定することができます。左の例では、ステータスコードが GUI 表示フォーム上ではラジオボタンとして表示され、テーブル形式の GUI 表示フォームではコンボボックスとして表示されるように設定されています。

   デフォルト /NULL: NULL がどのように使用されるか、またデフォルト値を持つかどうかを指定します。

   格納 /SQL: データが DBMS 内でどのように保存されるかを指定します。

項目モデルを使用することで、データの書式がどのように設定され、アプリケーションでどのように使用されているかを制御することができます。**項目モデル**が作成された場合、該当するデータタイプであればどこでも使用することができます。上記の場合、レコード内のカラムとして指定したり、パラメータとして渡されたり、プログラム内の変数として使用することができます。

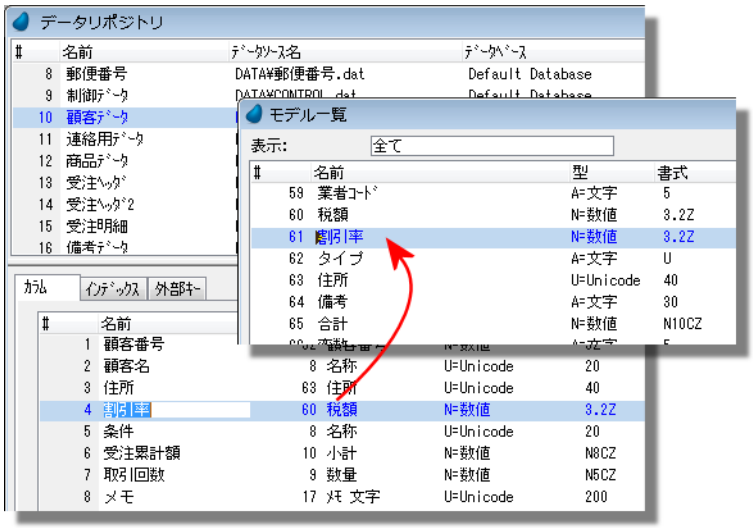
参照 : 「プロジェクトのデータ項目とコントロールを標準化するには」（42 ページ）
「モデルを使用してデータソースのカラムを定義するには」（41 ページ）

モデルを使用してデータソースのカラムを定義するには

データを一貫性を持って定義することは非常によいことです。モデルを使用することで一貫性を実現することが容易になります。一度、データモデルを定義すると、すべてのデータソース内で同じデータ型を持たせることができます。

モデルを使用してデータソースのカラムを定義する

必要条件： 項目モデルは、すでにモデルリポジトリに定義されているものとします。



1. データリポジトリの**カラム**タブに移動します。
2. *F4*（または、**編集→行作成**）を押下して、カラムを追加します。
3. **名前**カラムは空白の状態にしておいてもかまいません。モデルを指定すると、モデル名が継承されて名前カラムに設定されます。
4. **モデル**カラムで**ズーム**すると（*F5*、または**ダブルクリック**）選択可能なモデルが一覧表示されます。ここでは、ステータスコードモデルを選択します。

ここで追加されたカラムの**特性シート**を表示します。ステータスコードモデルから継承された特性値が表示されているはずです。これは、モデルを使用することでステータスコードの関する設定を行う必要がないことを意味しています。もし、ステータスコードモデルの特性値を変更すると、カラムの特性値も変更されます。

必要であれば、この特性値をオーバライドすることもできます。特性値がオーバライドされた場合、モデルとの継承関係が切れ、モデルの特性値が変更されてもこれらの特性値は変わらなくなります。

参照： 「再利用可能なデータオブジェクトを定義するには」（39 ページ）

プロジェクトのデータ項目とコントロールを標準化するには

どのようなプロジェクトでもデータに関する２種類の問題が考えられます。

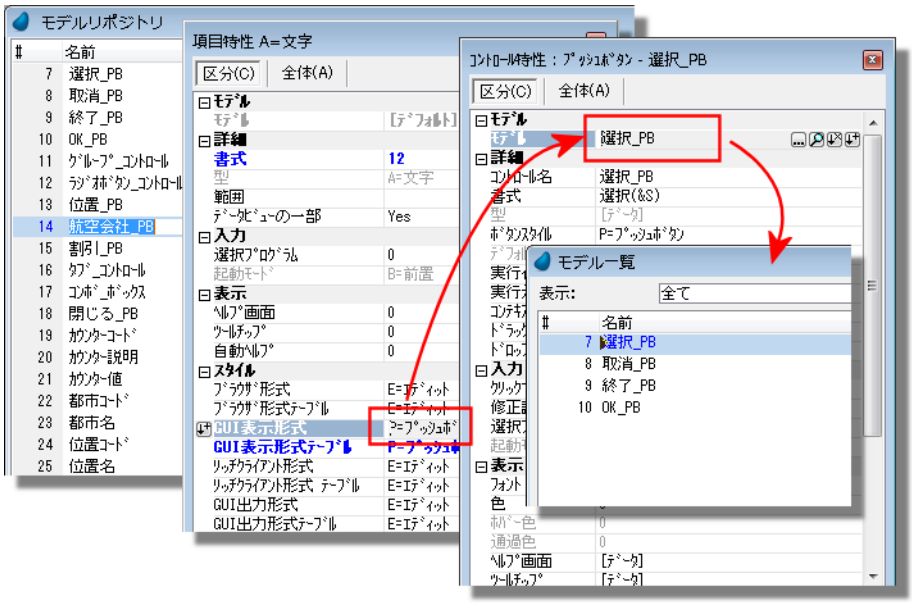
1. データ自身について
   - 各項目の長さがどれくらいか？、
   - どのような形式で保存するか？
   - データタイプは何にするか？
   - どの値を正しいものとするか？

   などです。開発時は、**項目モデル**としてこの情報をカプセル化します。
2. データをどのように表示させるか、またユーザがどのようにアクセスするようにするか？
   コントロールモデルとなる **GUI 表示形式**、**GUI 出力形式**、**リッチクライアント形式**、**ブラウザ形式** そして **テキスト形式** の各モデルは、ルック & フィールをカプセル化します。

フォームにデータ項目を配置したとき、コントロールのすべての特性値を設定するかコントロールのモデルを選択することができます。また、**項目モデル**の一部としてコントロールモデルを指定することもできます。この２つはもっとも基本的なレベルでリンクされています。これによって、プログラミングにかかる時間を短縮し、フォーム上のコントロールとして表示されるデータ形式を標準化することができます。

項目モデルに対するコントロールを定義する

必要条件： **コントロール**モデルはすでに定義されているものとします。



1. **項目**モデルにカーソルを位置付けます。
2. モデルの**特性**シートを開きます（***Alt+Enter***）。
3. **スタイル**セクションの変更したいフォームに移動します。例えば、スクリーンモードでのウィンドウに表示される内容を変更したい場合は、**GUI 表示形式**特性に移動します。
4. ここで**ズーム**すると**コントロール特性**と表示されるペインが新たに表示されます。必要であれば、ここに特性値を設定することができますが、ここではモデルを設定します。
5. **コントロール特性**内の**モデル**特性 から**ズーム**（***F5***、または**ダブルクリック**）して一覧からモデルを選択します。

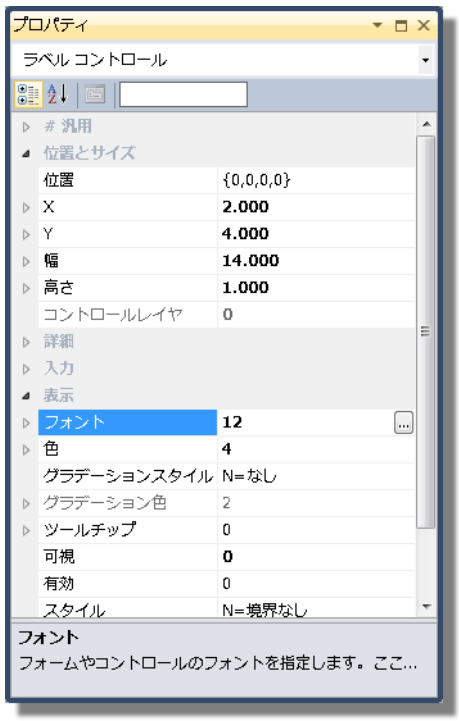
これによって**項目モデル**が**コントロールモデル**とリンクされました。**項目モデル**が、データを定義するために使用されと、**コントロールモデル**で指定された**スタイル**特性を持つことになります。

注： コントロール特性は、フォームエディタ上で表示されるものとは表示形式が異なります。

参照： 「再利用可能なデータオブジェクトを定義するには」（39 ページ）

モデル特性の変更内容をオブジェクトに反映させないようにするには

すでにモデルが定義されている場合、モデルの内容を変更するとそのモデルを使用しているすべてのオブジェクトに反映されます。これはとても効果的です。例えば、すべて***イタリック体***に設定されている項目を標準に戻したい場合、モデルを変更することですべての表示が簡単に変更されます。



ここで、特定のウィンドウの表示だけモデルの変更を反映させたくない場合があるとします。この場合、継承関係を***解除する***必要があります。Magic xpa では、特性値を変更することで表示が**青いボールド表示**となり、継承が***解除された***ことを示すようになっています。

注： 継承状態を表す色とフォントは変更することができます。

- 色を変更するには、**アプリケーション用基本色定義**テーブル（**設定→オプション→基本色→開発**）を開いて、#44 と #45 の色を変更します。
- フォントを変更するには、**アプリケーション用フォント定義**テーブル（**設定→オプション→フォント→開発**）を開いて、#34 と #35 のフォントを変更します。

右の例において、**フォント**特性と**色**特性は、**ボールド体**になっています。これらは、継承関係が解除されていることを表しています。

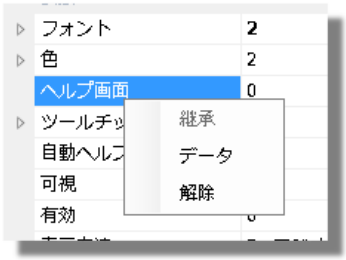
継承関係を解除するには、手動と自動の２つの方法があります。

手動で継承を解除する

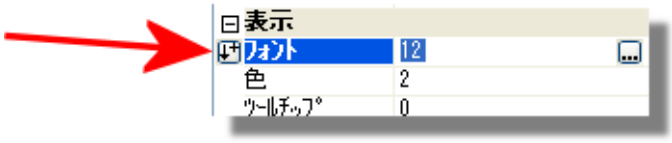
フォームデザイナの特性で継承メカニズムを制御するには、特性値上で右クリックし、***継承***、***データ***また***解除***を選択してください。

変更された特性（継承が解除された特性）は、**ボールド体**で表示されます。データとして設定された特性は、でマークされます。

［データ］オプションは特定の特性でのみ表示され、値がある時だけ［データ］特性に設定します。このオプションが選択されると、特性はデータ特性から継承され、モデルから値を受け取ることはありません。



［フォーム］デザイナでない場合の特性では、特性値の左側に表示されるアイコンを クリック することによって継承を解除する（継承させる）ことができます。



このアイコンは、特性値にカーソルを置いた場合のみ表示されます。アイコンが  と表示されている場合は、クリック すると継承が解除され、特性値が青色に変わります。アイコンが と表示されている場合は、クリック すると継承された状態になり、特性値は黒色に変わります。

自動的に継承を解除する

オブジェクトの特性値を変更するとその特性に対する継承関係は自動的に解除され、特性値が青色に変わります。もしオブジェクトの継承された**色**特性が **2** の場合、この値を **4** に変更すると **色**特性の継承関係は解除されます。

参照： 「継承が解除された特性を継承させるには」（44 ページ）

継承が解除された特性を継承させるには

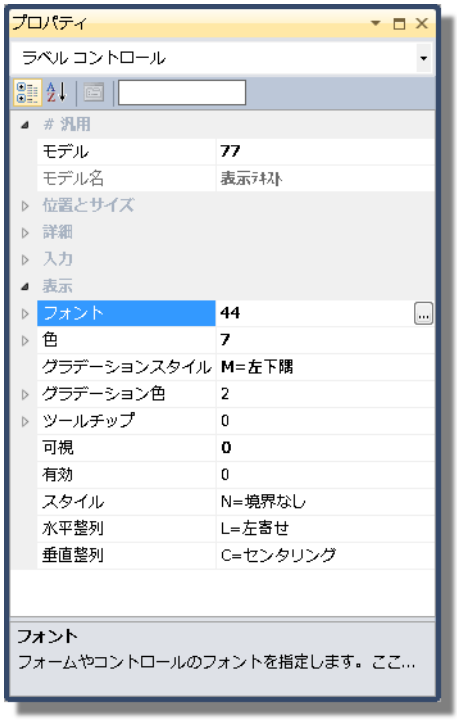
一旦モデルを定義すると、そのモデルに対する変更内容はモデルを使用するすべてのオブジェクトに自動的に反映されます。これは、とても有益です。たとえば、必須項目のすべてがイタリック表示になっている場合、モデルを変更するだけで全て標準表示に変更できます。

しかし、特定のダイアログに対して、モデルの変更の影響を受けないようにしたい場合があると想定してください。このような場合、継承関係を解除する必要があります。Magic xpa では、継承が解除された特性は太字の青色で表示されるため、解除されたかどうかを見分けることができます。

注： 表示するフォントはカスタマイズすることができます。この場合、開発用のフォントと基本色を使用しています。

- 色を変更するには、開発用基本色（オプション → 設定 → 基本色 → 開発）を開き、#44 と #45 の色を変更します。
- フォントを変更するには、開発用フォント（オプション → 設定 → フォント → 開発）を開き、#34 と #35 のフォントを変更します。

右の例では、フォントは、" ボールド体 " になっています。



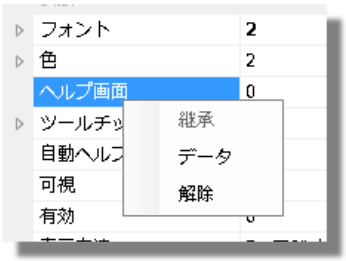
一端解除された継承関係を再び有効にするには、特性値の左側に表示されるアイコンを使用します。

特性値を継承させる

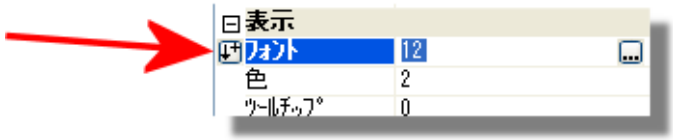
フォームデザイナの特性で継承メカニズムを制御するには、特性値上で右クリックし、***継承***、***データ***また***ブ解除***を選択してください。

変更された特性（継承が解除された特性）は、ボールド体で表示されます。データとして設定された特性は、でマークされます。

［データ］オプションは特定の特性でのみ表示され、値がある時だけ［データ］特性に設定します。このオプションが選択されると、特性はデータ特性から継承され、モデルから値を受け取ることはありません。



［フォーム］デザイナでない場合の特性では、特性値の左側に表示されるアイコンを クリックすることによって継承を解除する（継承させる）ことができます。



このアイコンは、特性値にカーソルを置いた場合のみ表示されます。アイコンが と表示されている場合は、クリックすると継承が解除され、特性値が青色に変わります。アイコンが と表示されている場合は、クリックすると継承された状態になり、特性値は黒色に変わります。

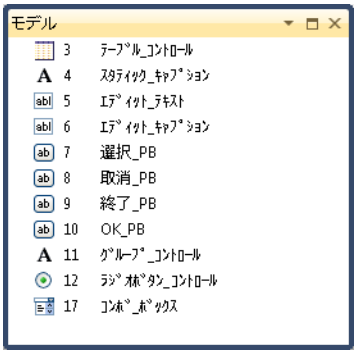
モデルのクラスを変更するには

モデルは一度作成されると、クラスを変更することはできません。モデルを作成し値を設定した後、カーソルを別の行に移動するとクラスの設定は確定されます。何らかの理由で間違ったクラスを指定されたモデルを作成してしまった場合、*F3*（**編集→削除**）を押下してそのモデルを削除し、再度作成し直す必要があります。

そのモデルが他のオブジェクトから参照されているかどうかを確認する場合は、削除する前に *Ctrl+F*（**編集→検索と置換→クロスリファレンス**）を押下して使用しているオブジェクトを検索してください。

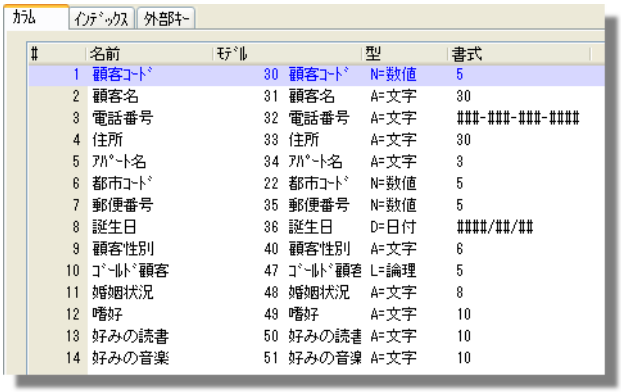
参照： 第 1 章：「クロスリファレンスを使用する」（7 ページ）

コントロールモデルを使用して、フォームにコントロールを自動配置するには



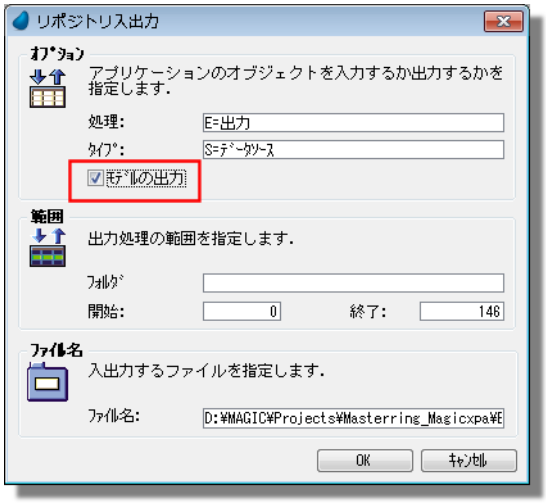
モデルペインからモデルをフォーム上にドラッグするだけです。Magic xpa は、自動的に必要なコントロールを作成して、選択されたモデルをそのコントロールに割り付けます。

モデルを含めてプログラムやデータソースを出力するには



モデルはとても有効的なので、データソースやプログラムなどによく利用されます。しかし、これらをリポジトリ入出力する場合は、多少複雑になります。なぜなら、入力するプロジェクト内に同じモデルが存在している必要があるからです。

このような場合の簡単な対処方法として、リポジトリ入出力ダイアログに表示される**モデルも出力する** オプションを使用することです。このオプションを使用することで、データソースやプログラムが使用しているモデルも一緒に出力されます。



モデルを含めて出力する

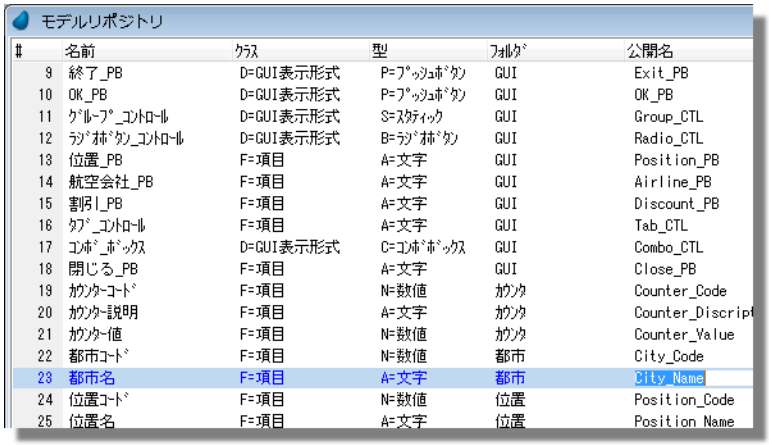
1. リポジトリ出力を行う場合、**モデルも出力する**をチェックします。
2. このリポジトリファイルを入力すると、出力されたモデルが現在のモデルリポジトリの最後に追加されます。これらのモデルへの参照情報が更新され、データソースやプログラムが正しく継承されるようになります。

モデルが定義されたプロジェクトにデータソースを入力すると、モデルの項番は異なりますが、入力したデータソースが使用してたモデルがプロジェクトに追加されます。

参照： 第 2 章：「別のプロジェクトにオブジェクトを転送するには」（36 ページ）

複数のプロジェクトでモデルを共有するには

異なるプロジェクト同じ内容のモデルを使用する必要があります。リポジトリ入出力によってモデルを別のプロジェクトにモデルをコピーすることもできますが、プロジェクト間でモデルを共有することの方が保守も楽になります。また、これによってプログラムを変更することなくモデルの機能を変更することができます。例えば、日付項目に対応したポップアッププログラムとしてカレンダーを追加したい場合、またはアプリケーションが実行している国の通貨に対応したコンポーネントが必要な場合などが考えられます。



コンポーネントとしてモデルを共有する

1. よく使用するモデルを定義します。
2. 共有する際に参照する**公開名**をモデルに指定します。
3. コンポーネントの作成とインストール処理を実行します。

コンポーネントをプロジェクトに定義した場合、コンポーネントモデルはモデル一覧から選択できるようになります。コンポーネントモデルは、プロジェクトのモデルリポジトリに定義されているものと同じように扱うことができます。

参照： 第 15 章：「プロジェクト間で Magic xpa のオブジェクトを再利用するには」（357 ページ）

第 4 章： Magic エンジン

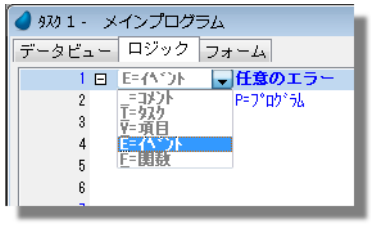
アプリケーションレベルのイベントを定義するには

Magic xpa では、タスクが実行中に発生させることのできるイベントを定義することができます。しかし、どのタスクが実行中でも、タスクが実行していなくても発生させることのできるイベントが必要な場合があります。本章の例は、数分ごとにメッセージをチェックするメッセージ通信システム、または処理されないデータベースエラーに対してメッセージを表示させるグローバルなエラーハンドラ、あるいはホットキーに設定されたポップアッププログラムについて説明しています。これらはどのタスク上でも発生させることができます。

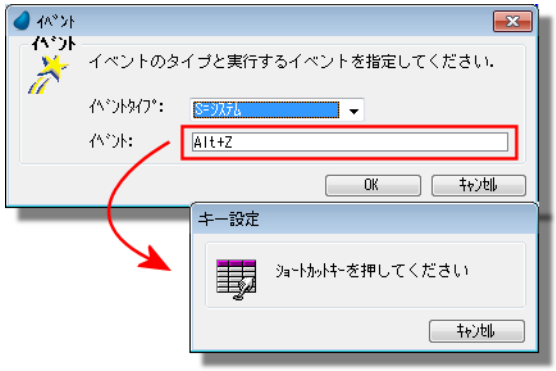
これらのグローバルイベントは、プロジェクトのメインプログラムで定義します。

グローバルイベントを作成する

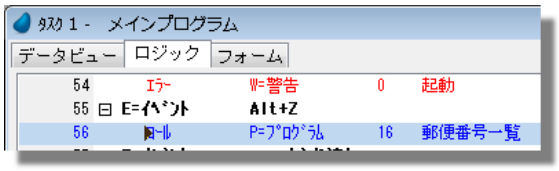
1. メインプログラムを開きます。
2. ロジックタブ（*Ctrl+1*）をクリックします。
3. *F4*（編集→行作成）を押下して行を追加します。2 つの新規行が表示されます。最初の行に移動し、ドロップダウンリストからイベントを選択します。



4. *Tab* で右に移動すると、イベントダイアログが表示されます。
5. 使用するイベントタイプを選択します。この例では、キー操作によるシステムイベントを選択しています。
6. イベントでズーム（*F5*、または編集→ズーム）するとキー定義ダイアログが表示されます。
7. トリガとして定義するキー操作を押下します。この場合は、*Alt+Z* を設定しています。



8. これで、*Alt+Z* を押下することで発生するグローバルイベントが定義できました。次にこのイベントが発生した場合のロジックを定義する必要があります。
ここでは、郵便番号一覧というプログラムを起動するようにしています。このプログラムは、郵便番号コードの一覧を表示するものです。



参照： 第 10 章：「特定のコントロールにパークしている時にのみ実行されるイベントロジックユニットを定義するには」（238 ページ）

Magic エンジンをイベント駆動型で動作させるには

以前のプログラミング言語では、プログラムは主に手続き型でした。すなわち、プログラムは上から下にロジックが実行されるように記述します。プログラムの開始処理以外、ユーザからの入力処理はほとんどありません。

今日、ほとんどのプログラムはイベント駆動型です。すなわちプログラムは、実行されるとイベントの発生を待つ状態になります。イベントが発生するとそれに対応した処理を実行します。

簡単な例として、Web サイトや SOAP サービスがあります。Web サイトは常にそこにあり、キーを押したり、特定の場所でマウスカーソルが通過するでそれに対応した処理が実行されます。SOAP サービスは、決まった書式でデータを送信することでデータが返されます。

Magic xpa では、これら両方の種類のプログラムを作成することができます。その際、バッチプログラムであってもイベント駆動方式を利用することがよくあります。これは古い手続き型の方法より簡単で効率的なためです。

イベントのコンセプト

イベントの基本的な考え方は簡単です。何か（トリガ）が発生し、ロジック（ハンドラ）を実行することでプログラムが応答するというものです。

トリガはユーザによって発生されるものがよくあります。例えば、マウスカーソルを通過させたり、キーを押下したりする操作があります。カーソルを項目内に移動させたり、項目外に移動させたりすることもあります。キー操作が何も行われなければ、何も発生しないこともあります。一定の時間が経過することで発生する場合もあります。

しかし、プログラムはまた、他のプログラムと対話するためにイベントを発生することもあります。ActiveX オブジェクトがよい例です。これには、対応することのできるイベントのグループが含まれています。

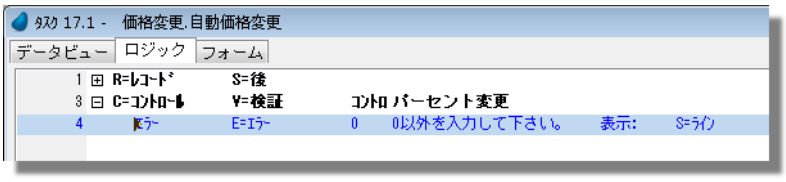
発生させることができる内容を以下に示しました。

発生内容 **イベント**	**Magic xpa でどのように処理されるか** **ロジック**	**サブタイプ**
タスク起動	タスク	前
タスク終了	タスク	後
新規レコード読込	レコード	前
ユーザがこのレコードを離れた	レコード	後
ユーザが項目にカーソルを置いた	コントロール	前
ユーザが別の項目にカーソルを移動した	コントロール	後
X 秒間キー操作を行わなかった	イベント	タイマイベント
ユーザがプッシュボタンを押下した	イベント	ユーザイベント
項目の値が変更された	項目変更	
時間が 11:03 AM になった。	イベント	式
.NET オブジェクトがイベントを発生させた	イベント	.NET
ユーザがキー操作を行った	イベント	システム
DBMS により重複レコードエラーが発生した	イベント	エラー

処理が可能なイベントが数百あっても、現実的には、通常よく利用する処理は Magic xpa によって自動的に実行されます。例えば、明示的に DB テーブルを開いたり、閉じたりする必要はありません。また、初期設定やデフォルト値、データの妥当性をチェックするような処理は**項目モデル**によって処理されます。

また、モデルやフォーム上のコントロールに選択プログラムを定義したり、**チョイス**コントロールをデータソースとリンクさせたりすることで選択プログラムにリンクさせることができます。

処理したいイベント用に、タスクの**ロジックエディタ**に複数の行を追加することになります。ここでは項目への入力を必須化させるための簡単な例を挙げます。



必要条件： フォームにコントロールが配置されており、コントロール名が設定されているものとします。

ロジックユニットを作成する

1. タスクを開き、ロジックタブをクリックします（*Ctrl+I*）。
2. ロジックユニットを入力したい行に移動します。
3. *Ctrl+H*（編集→ヘッダ行作成）を押下してヘッダ行を作成します。
4. ロジックユニットの**タイプ**として *C= コントロール*を選択します。*Tab* で次のカラムに移動します。
5. コントロールロジックユニットの**タイプ**として *V= 検証*を選択します。*Tab* で次のカラムに移動します。
6. ズーム（*F5*、またはダブルクリック）して検証対象となるコントロールを選択します。一覧に選択したいコントロール名が表示されない場合は、フォームにコントロールが配置されていないかコントロール名が定義されていないかのどちらかです。

この設定によって、コントロールを通過するたびにロジックユニットが実行されます。このロジックユニット内でエラー処理コマンドを追加します。これにより、項目にデータが入力されるまでユーザは他の操作が何もできなくなります。

参照： イベント処理のホワイトペーパ
「イベントの階層を利用するには」（52 ページ）

イベントの階層を利用するには

イベントについて考える上で問題になる点の 1 つは、誰がイベントを処理するかということです。すなわち、イベントが実行された場合、処理できるタスクにはいくつかのレベルがあるかもしれないということです。

1 つの例とし DBMS エラーがあります。DBMS がエラーを発生した場合、エラーの内容に応じてエラーメッセージを表示するようにしたいと考えられます。また、データベースにすべての DBMS エラーを記録するようなグローバルなエラー追跡プログラムが必要な場合があります。

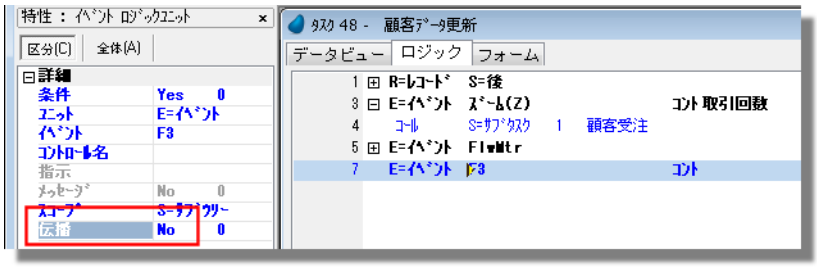
また、例えばキー操作を無効にする必要があるかもしれません。例えば、*F3* はデフォルトでは行を削除するために割り当てられており、*F3* を押下すると自動的に現在のレコードを DB テーブルから削除します。しかし、*F3* に対して削除レコードをマークするように処理を変更する必要があるかもしれません。また、*F3* に対してロジックユニット内で関連するレコードを削除し、現在のレコードを削除させる機能を持たせる必要があるかもしれません。

このようなことは、**伝播**特性を使用することで実現できます。**伝播**特性が *Yes* に設定されている場合、ロジックユニットが実行され、さらにイベントが上位のタスクレベルに伝播されます。*No* に設定された場合、現在のロジックユニットが実行するだけで終わります。

Magic xpa は、ロジック階層を決定するための決められた検索順序を使用します。それは、最初に特定のコントロールに対応したイベントが処理され、その後、対応していないコントロールに移ります。この検索処理はタスクツリーを上位方向へと実行され、伝播指定が有効であればメインプログラムまで行われます。

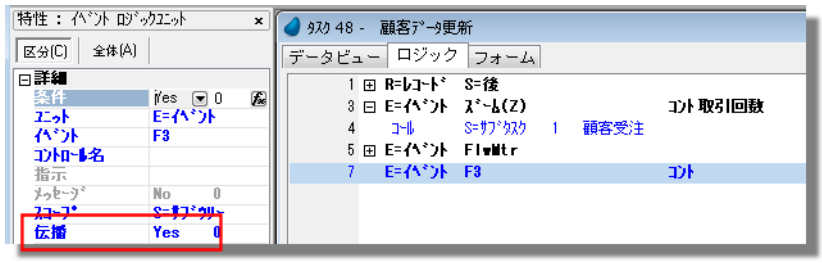
では具体的な例で説明します。

システムイベントをブロックする



ここでは、*F3* の動作をブロックするイベントがあります。ユーザが *F3* を押下するとこのイベントが起動されます。しかし、このイベントは伝播されず、Magic xpa はこれを処理できません。このためレコードの削除は行われません。

システムイベントに機能を追加する



このロジックユニットは、関連するレコードを削除するサブタスクを起動するように定義されています。*F3* のシステムイベントを Magic xpa に渡すと現在のレコードが削除されます。

ヒント : *この例では、キー操作をブロックする方法について説明していますが、通常は、行削除の内部イベントを使用するようにしてください。これは、ユーザがプルダウンメニューの行削除を使用したり、キー割付が変更されたり、プッシュボタンにイベントを割り当てることでプログラム的に削除される場合があるためです。行削除のイベントの使用方法については、「内部イベントをブロックする」（53 ページ）を参照してください。*

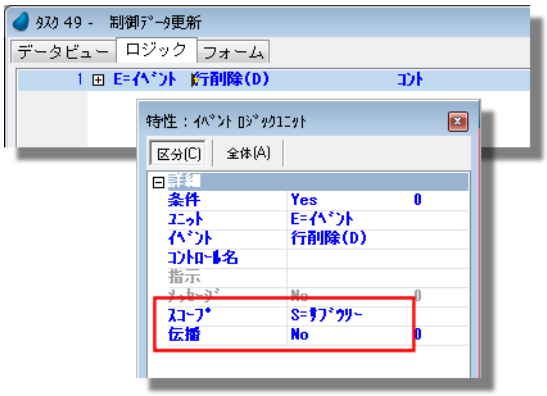
行削除のような内部イベントの発生を防止するには

Magic xpa には、たくさんの組み込み機能を持っています。これによって開発工数を軽減することができます。行作成や削除、終了 のような共通のイベントはエンジンに組み込まれており、プログラムに組み込む必要はありません。

デフォルトのキーボード割付やメニューの設定を変更することでシステムレベルのイベントをブロックすることができます。また、タスクオプション（タスク特性→オプションタブ）内のパラメータを無効にすることで、イベントが発生しても実行させないようにすることもできます。また、内部イベントを受け取るロジックユニットに処理を組み込むことでタスクレベルでこれらを制御したりオプション化することもできます。

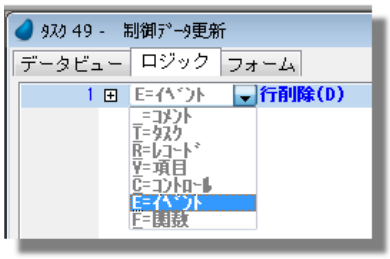
この例では、受注状態が N（新規）でない場合、行削除イベントをブロックします。

内部イベントをブロックする

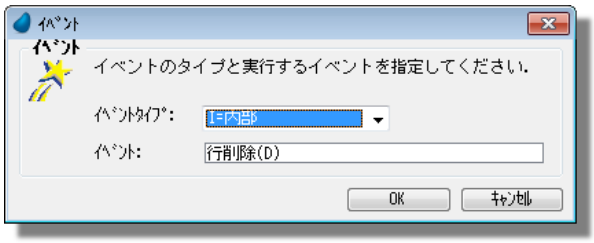


ここでは、受注状態 <>'N' の場合に行削除イベントをブロックするロジックユニットを例に説明します。このロジックユニットではユーザに対してメッセージを表示するようにする必要があるかもしれませんが、ブロック処理には必要ありません。以下にこのイベントをどのように定義するかを説明します。

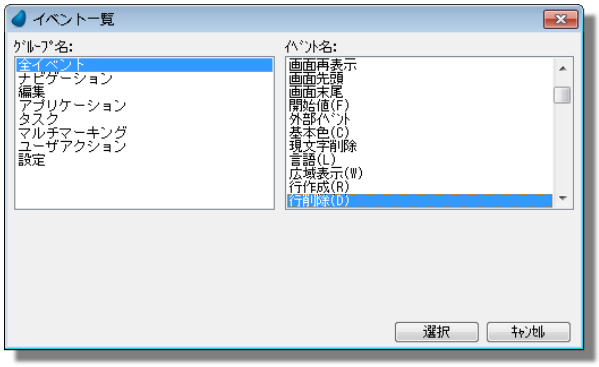
1. タスクを開き、**ロジック**タブ（*Ctrl+1*）を**クリック**します。
2. **ロジックユニット**を入力したい行に移動します。
3. *Ctrl+H* を押下し、ヘッダ行を作成します。
4. **ヘッダタイプ**で ***E= イベント***を選択します。



5. ダイアログボックスが表示されます。**イベントタイプ**で ***I= 内部***を選択します。*Tab* で次の**イベント**に移動します。

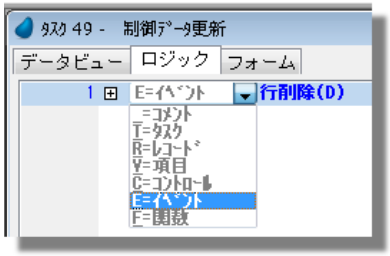


6. *Tab* を押下すると**イベント一覧**ダイアログが表示されます。
   選択したいイベント名の文字を入力することで該当するイベントに位置づけられ選択することができます。この場合は、行と入力することで***行***という名前を持つイベント名に位置づけることができ、カーソルキーで**行削除**のイベント名の位置付けることができます。
   使用したいしたいイベント名が見つかったら、選択ボタンを押下するか *Enter* を押下します。

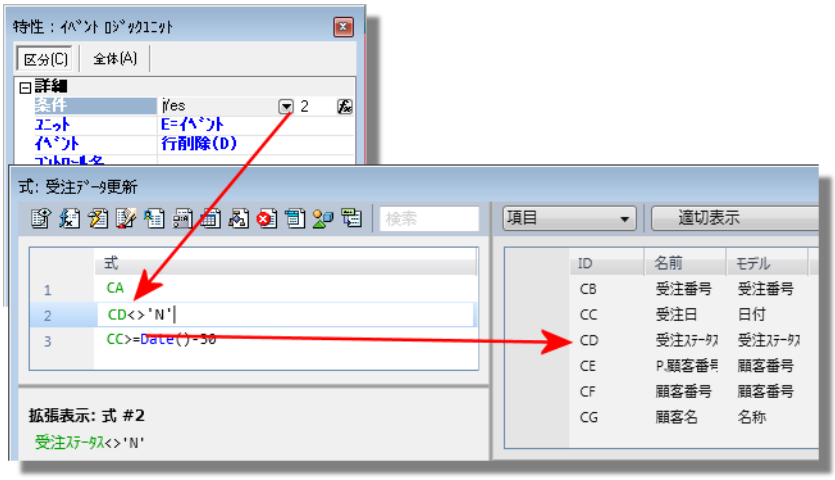


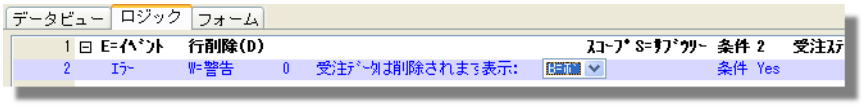
7. もう一回 *Enter* を押下し、**イベント**ダイアログを閉じます。
8. これで**行削除**イベントが発生した場合に実行される **イベント**ロジックユニットが定義できました。
   イベントが伝播されることを防ぐため、**イベント特性**シート（*Alt+Enter*）を開き、**伝播**特性を ***No*** に設定します。
9. **行削除**イベントトリガが下位のタスクに渡らないようにため 、**スコープ**特性を ***T= タスク***に設定します。

1. タスクを開き、**ロジック**タブ（*Ctrl+1*）を**クリック**します。
2. **ロジックユニット**を入力したい行に移動します。
3. *Ctrl+H* を押下し、ヘッダ行を作成します。
4. **ヘッダタイプ**で ***E= イベント***を選択します。



10. これで**行作成** イベントが発生すると必ずこの**ロジックユニット**が実行されるようになります。
しかし、ステータスコードが N でない場合のみ実行するようにしなければなりません。
この処理を実現するには**条件**特性で**ズーム**（*F5*、または**ダブルクリック**）し、**式エディタ**を開きます。ここで条件式を定義します。参照したい項目を見つける場合は、**ズーム**を行い右側の項目一覧に移動することで参照することができます。（第 20 章：「式エディタ内で式の体裁を整えるには」（461 ページ）を参照してください。）





これで作成できました。この**ロジックユニット**にメッセージを表示するなどの処理を追加したりすることができます。

参照： 「Magic エンジンをイベント駆動型で動作させるには」（50 ページ）

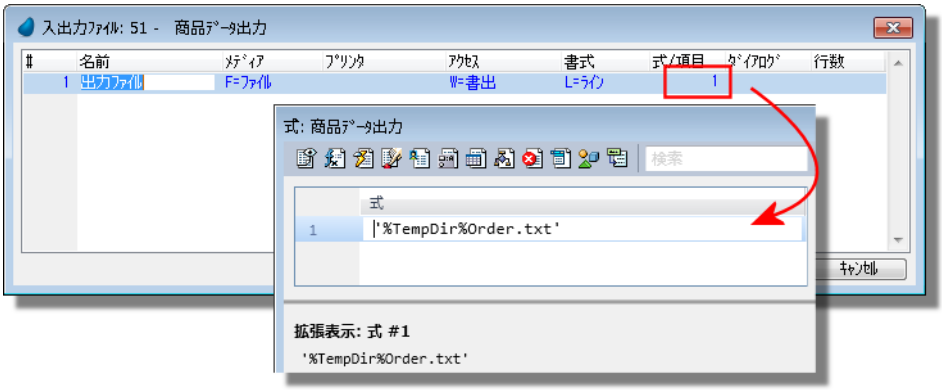
入出力ファイル名を動的に設定するには

Magic xpa で入出力ファイルを定義 / 利用する場合、以下のような方法でファイル名を定義することができます。

- 固定のファイル名を指定できます。（例：'C:¥Temp¥Report.txt'）
- データ項目で指定できます（例：**Trim** (*AF*)）。指定されるデータ項目は、データソースのカラムであったり、ユーザによって入力されたデータの場合もあります。ファイル名をランダムに指定したり、項目内で保存することもできます。一時ファイルを使用する場合に有効です。
- ファイル名の一部または全部を***論理名***を使用して設定することができます。パス名を実行時に動的に指定するうえで有効な方法です。よく利用する論理名に、**%TempDir%** があります。これは、Magic エンジンが自動的に設定するものです。

上記のすべての場合において、ファイル名は式エディタで指定されます。ここでは、テキストファイルを出力する方法について説明します。

入出力ファイル名を設定する



1. 最初に、テキストファイルを出力するタスクを開きます。ナビゲータペイン（*Alt+F1*）からタスクをクリックすることで開くことができます。
2. *Ctrl+I*（タスク→入出力ファイル）を押下し、入出力ファイルテーブルを開きます。まだ出力ファイルが定義されていない場合、以下のようにして作成します。
   - *F4*（編集→行作成）を押下して 1 行追加します。
   - **名前**カラムに出力ファイルの名前を入力します。
   - **メディア**カラムでは、***F= ファイル***を選択します。
   - **アクセス**カラムでは、***W= 書込***を選択します。
   - **書式**カラムでは、***L= ライン***を選択します。
3. 次に式カラムに移動し、ズーム（*F5*、またはダブルクリック）して式エディタを開きます。
4. ファイル名が返る式を入力します。ここでは、**'%TempDir%Orders.txt'** が入力されています。指定された論理名が正しく評価される限り、この論理名とファイル名の組み合わせは常に使用することができます。

このプログラムを実行させると、テキストファイルが Windows の一時ディレクトリに作成されます。

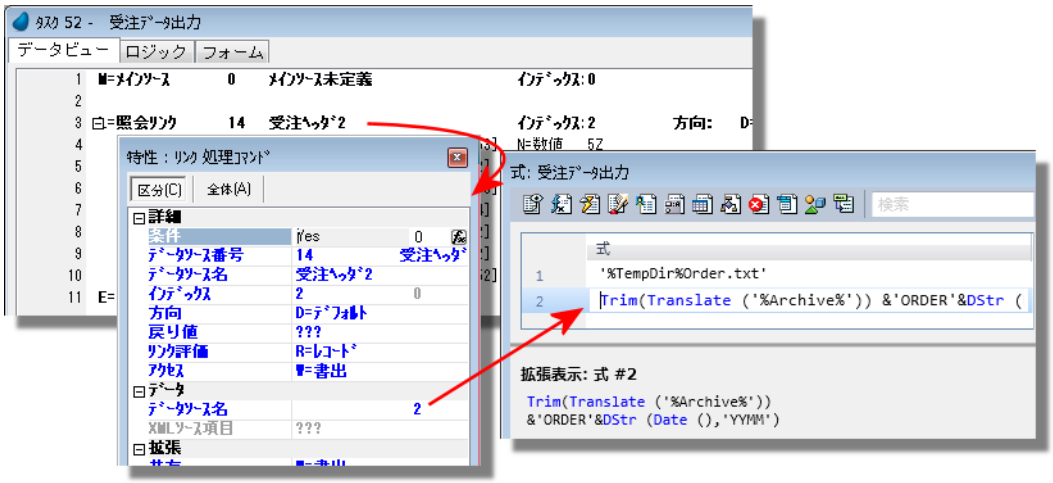
参照： 第 20 章：「式エディタ内で式の体裁を整えるには」（461 ページ）
第 2 章：「プロジェクトのディレクトリのファイルを読み書きするには」（31 ページ）

データソースの名前を動的に設定するには

通常、データソース名はデータリポジトリで設定されます。これによって保守が容易になります。一カ所で管理されていることで、データソース名を探したり、変更することが容易になります。

しかし、タスクレベルでデータソース名を変更したい場合があります。例えば、ユーザ毎に個別のテーブルを定義したり、同じデータ定義で日付によって異なるデータソース名にするようなことが考えられます。

タスクレベルでデータソース名を指定する

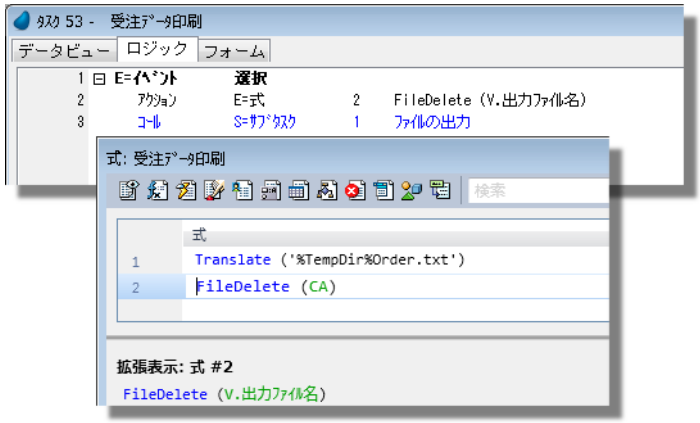


この例では、受注用のヘッダーレコードを複写するサブタスクがすでに定義されているものとします。ここではデータリポジトリ内の同じオブジェクトを開いてはいますが、**データソース名**特性を指定することで、デフォルトのデータソース名を上書きしています。データソース名は、実行時の年と月にもとづいて定義され（例えば、2007 年 12 月の場合は、'0712'）、論理名 **%Archive%** で指定されたディレクトリ内に作成 / 修正されるようになっています。

注： 同じタスク内では、同じデータソースに対して異なる名前を定義することはできません。この例では、親タスクがデフォルト名で同じデータソースを開くようになっているため、サブタスク内では保管用のレコードを作成しています。

参照： 第 17 章：「Magic xpa でデータベーステーブルを作成するには」（395 ページ）

同じファイルやデータソースを扱っているタスク内でファイルやデータソースを削除するには



Magic xpa のタスクが実行されると、プログラムの制御を行うことができる前に、そのタスクが使用するすべてのファイルとデータソースはオープンされます。このため、そのタスク内でファイルやデータソースを削除することはできません。

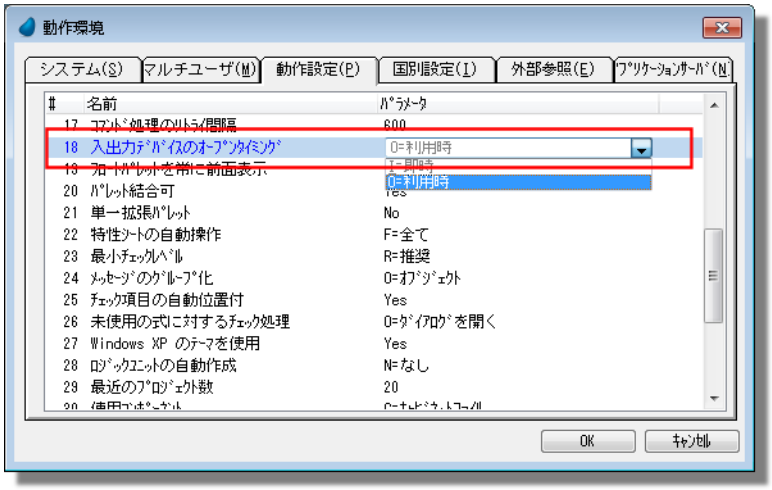
しかし、一般的には、タスクが実行される前にファイルを削除する必要があります。これを実現するには、親タスクでファイルを削除し、その後でタスクを起動する用にします。

この例では、古いファイルを削除し、ファイルを作成するタスクを呼び出すロジックユニットが定義されています。ここでは、**FileDelete ()** 関数と、ファイル名を格納する変数「CA」を使用しています。これらの処理は親タスク内で実行されるため、ファイルを開くサブタスクと競合することはありません。

出力する内容がない場合にファイルを作成したり空白ページを出力させないようにするには

デフォルトでは、タスクが起動される前に出力ファイルやプリンタがオープンされます。これは旧バージョン（Ver8）以前での Magic の仕様でした。しかし、タスクが何も出力しないことがわかっていれば、出力処理を実行しないようにするはずです。

開発者は、処理毎に出力内容が存在するかどうかは判別できません。従って、空データを出力することを防ぐには、実際にデータの存在が確認できるまで出力ファイルのオープンを遅らせる必要があります。



入出力ファイルをオープンするタイミングを指定する

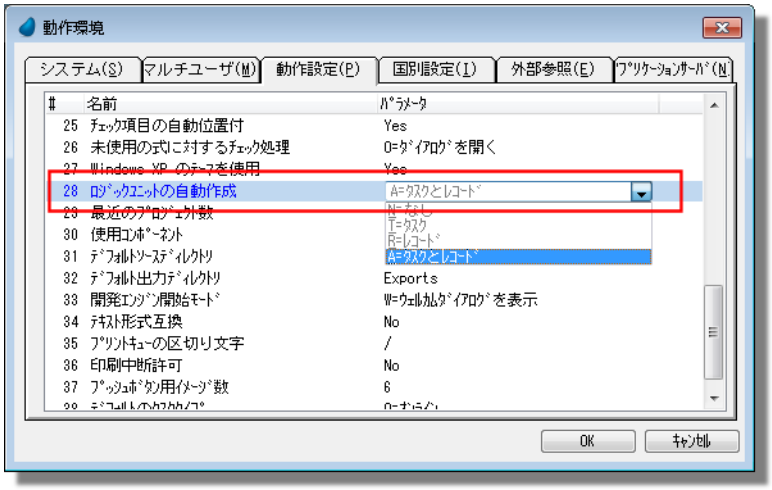
1. **動作環境**ダイアログ（**設定→環境設定**）を開き、**動作設定**タブを**クリック**します。
2. 行 #18 **入出力デーバイスのオープンタイミング**に移動します。
3. ***O= 利用時***を選択します。

これにより、Magic エンジンが**ファイルの書込**処理を実行するまで出力デバイスはオープンされなくなります。

タスク作成時にタスクレベルのロジックユニットを自動的に作成するようにするには

旧バージョンの Magic を使用している場合は、プログラム内にタスクとレコードのロジックユニット（ハンドラ）が常に表示されていることに慣れていると思います。これらは、実際に使用されていなくても常に表示されていました。

Magic xpa では、デフォルトではロジックユニットは作成されません。必要に応じて、*Ctrl+H* を押すことで作成することになります。しかし、これらを自動的に作成したいようであれば、以下のように環境設定を行ってください。



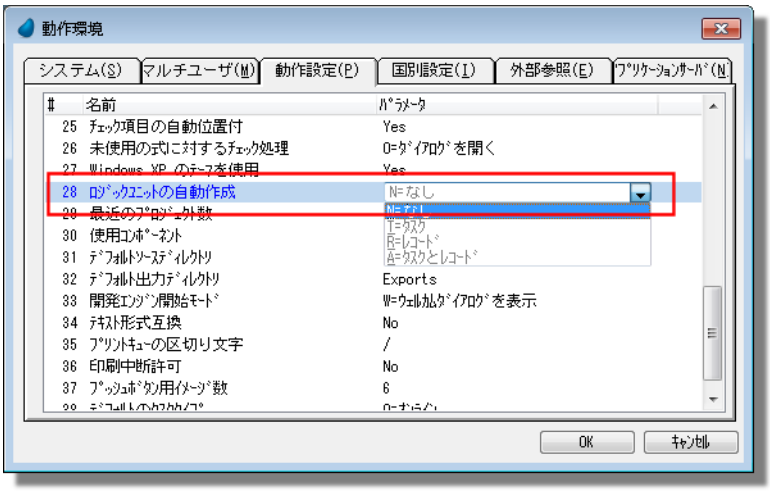
タスクとレコードレベルのロジックユニットを自動的に作成する

1. **動作環境**ダイアログ（**設定→環境設定**）を開き、**動作設定**タブを**クリック**します。
2. 行 #28 **ロジックユニットの自動作成**に移動します。
3. ***A= タスクとレコード***を選択します。

これで新規タスクを開くと、タスクとレコードレベルのロジックユニットが自動的に作成されます。

タスク作成時にロジックユニットが作成されないようにするには

Magic xpa が自動的にロジックユニットを作成するかどうかは、環境設定で決まります。新しいタスクを作成するたびに、ロジックユニットが作成されているようであれば、以下の操作でこの機能を止めることができます。



ロジックユニットの自動作成を止める

1. **動作環境**ダイアログ（**設定→環境設定**）を開き、**動作設定**タブを**クリック**します。
2. 行 #28 **ロジックユニットの自動作成**に移動します。
3. ***N= なし***を選択します。

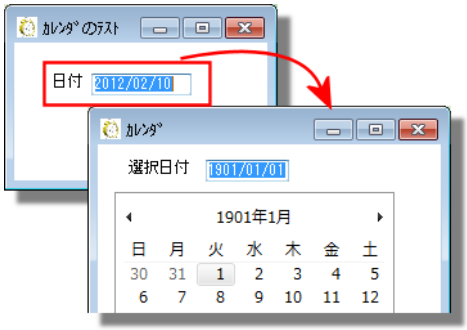
これで新規タスクを開いても、**ロジックユニット**は作成されなくなります。

イベント処理中にエディットコントロールの値を取得するには

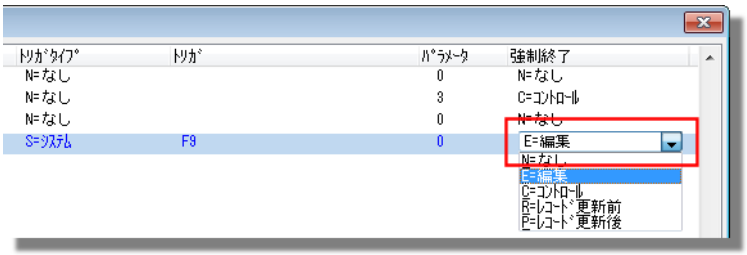
デフォルトでは、カーソルがエディットコントロール上に位置している間に、ハンドラが実行されても、そのコントロールの値は読み込まれません。これは、ユーザが新しい値を入力しても、そのコントロールから移動するまで、入力されたそはデータ項目内に存在していないためです。

これは、コントロール操作で起動されるハンドラによって発生する問題です。例えば、ユーザが *F9* を押下すると、表示されるカレンダが定義されているものとします。

デフォルトの日付は「1901/01/01」です。「2012/02/20」と入力し *F9* を押下します。しかし、起動されたカレンダは修正前の「1901/01/01」が表示されています。これでは、期待する動作にはなりません。



この動作を変更するには、ユーザイベントテーブルの**強制終了**カラムを変更する必要があります。



強制終了を ***E= 編集***に設定することで、ハンドラが実行される前に編集モードの状態のまま、入力された値によって項目が更新され、またこれによって再計算が実行されます。

この設定により日付のデータは期待された値で渡されます。

ヒント :強制終了 *特性は、ユーザイベントでのみ有効です。キー操作にようなシステムイベントで指定したい場合は、上記の例で行ったようにユーザイベントのトリガとしてシステムイベントを割り当てることで可能になります。*

注 : カレンダプログラムは、モデルの**選択プログラム**特性を使用して日付モデルに割り当てることができます。これによって日付項目は自動的に「ユーザが作成した」フォームに値が渡されるようにすることができます。しかし、ロジックユニットを使用することでより多くのオプションを与えることができます。例えば、*F9* を押下すると通常のカレンダが表示され、*F10* を押下するとスケジュールが表示されるようにできます。

参照 : 「Magic エンジンをイベント駆動型で動作させるには」（50 ページ）
『リファレンスヘルプ』: 強制終了

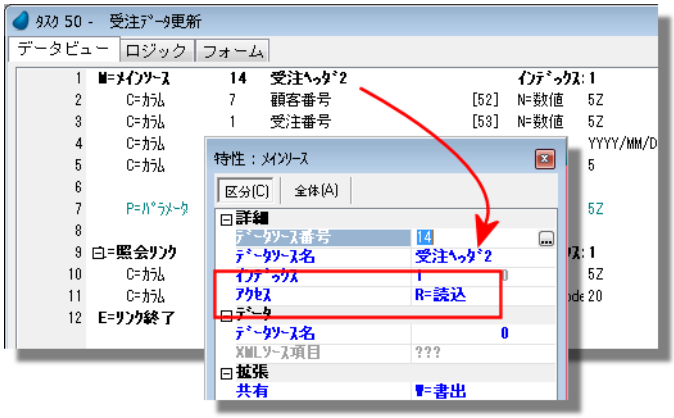
エンドユーザがタスク内でレコードを修正することを防ぐには

デフォルトでは、作成されたオンラインタスクを実行すると、ユーザはレコードに対してあらゆる操作（追加や削除、修正）が可能になります。データソースは書込モードでオープンされ、修正された場合レコードはロックされます。

これは、データソースを保守する際の簡単なプログラムを作成する上では有効なのですが、一般的なアプリケーションではユーザは、特性のレコードに対する修正処理に制限がかかっています。また、多くの画面は表示のみで、レコードの修正やロック処理に関する処理を必要としていません。

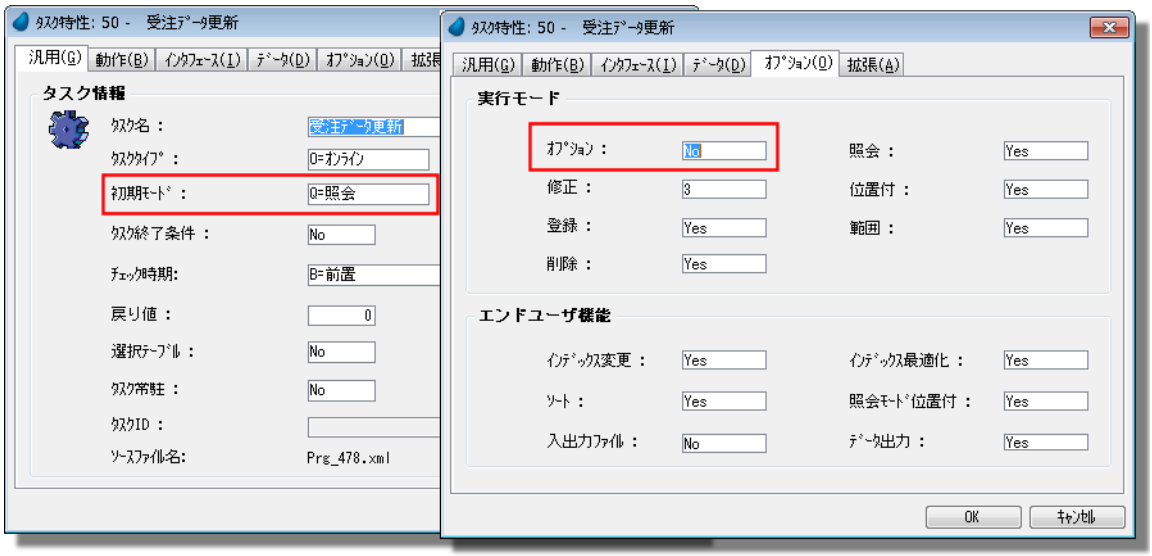
Magic xpa では、データ修正に関にして複数の制御レベルを持っています。以下この件について説明します。

データソースのアクセスモードを設定する



データソースの**アクセス**特性を設定することで、確実にレコードの修正を防ぐことができます。**読込**モードでオープンされたデータソースを修正しようとした場合、DB エラーが発生するだけでレコードは更新されません。またこの場合は、データソースをより早くオープンすることができます。このため、データの更新が必要でないタスクでは**読込**モードでオープンするようにすることを推奨します。

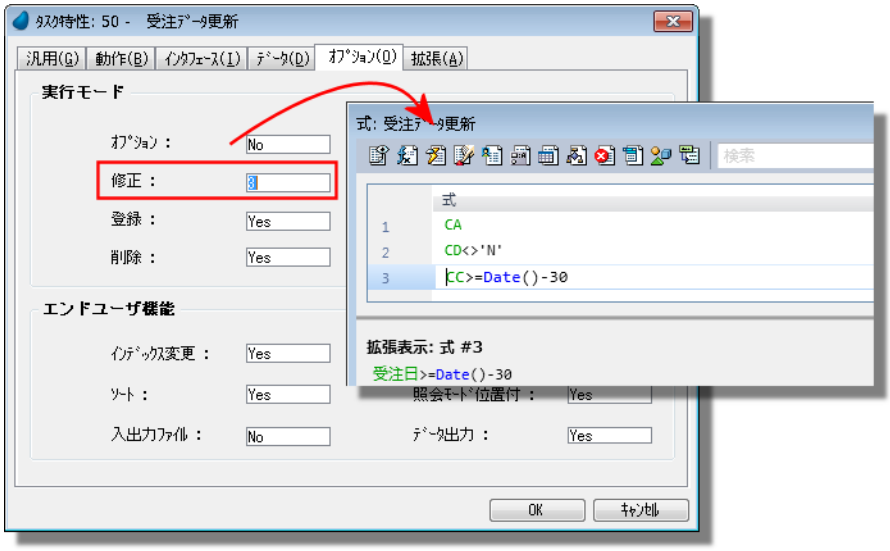
タスク特性の初期モードを使用する



タスク特性の**初期モード**特性を ***Q=照会***に設定することで修正できないモードでタスクを実行させることができます。これは、「検索」または「表示」のみのタスク用のモードです。

ただし、右上のフレームに表示されているように**オプション**特性（**タスク特性→オプション**タブ）を選択し、***No*** を設定することがない限り**オプション→修正**（***Ctrl+M***）を選択することで***照会***から***修正***にモードを切り替えることができます。

タスク特性のオプションを使用する



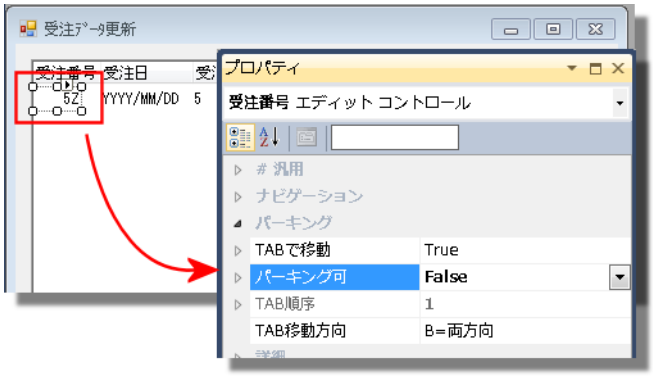
タスク特性（*Ctrl+P*）のオプションは、このタスクで有効な制御内容を指定することができます。各オプションは *Yes* または *No* あるいは、*式*で有効にするかどうかを指定できます。式で指定することでユーザ毎やレコード毎に動的に切り替えることができます。

上記の例では、レコードの削除は許可されていませんが、受注日付が 30 日未満の場合は修正が可能になっています。

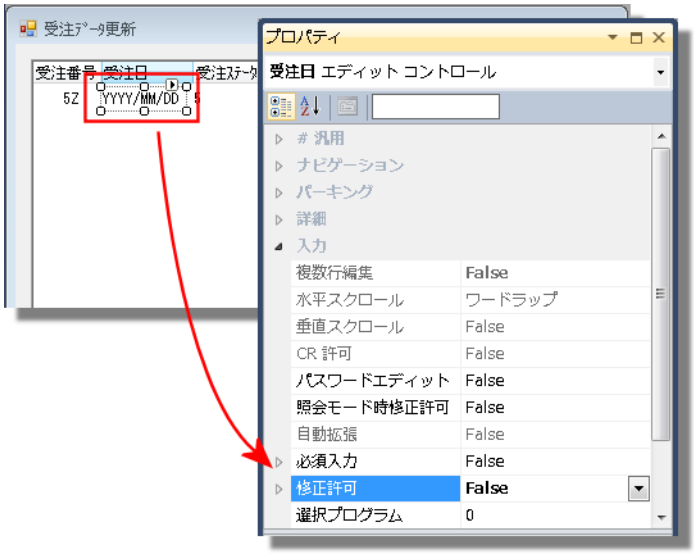
項目レベルで修正する

データソースレベルの修正に加え、項目レベルの制御も可能です。

- コントロール特性の**パーク可**特性を *False* に設定することで、ユーザはこの項目にパークすることができなくなります。



- コントロールの**修正許可**特性を使用することで項目にパークはできるが修正できないようにすることができます。ここには、*True* または *False* あるいは論理値が返る式で指定できます。



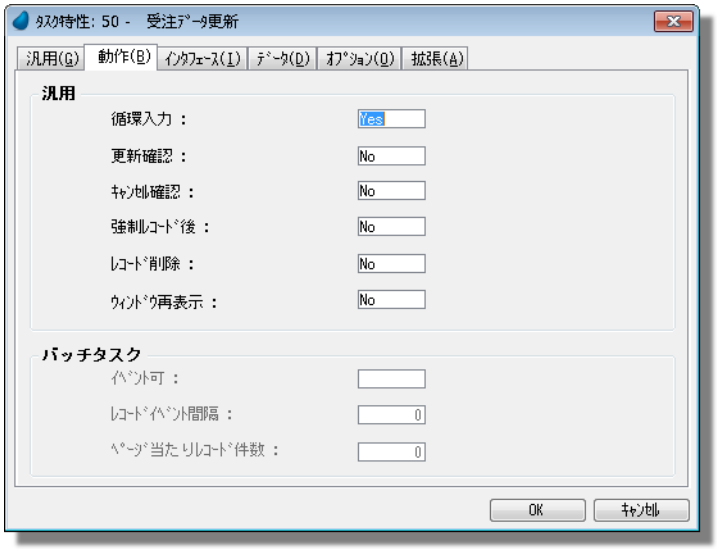
レコードが更新されなくてもレコード後を実行させるには

通常オンラインタスクでは、レコードがユーザによって更新された場合のみレコード後が実行されます。実際、レコードが更新されない限りレコードを書き込む意味はありません。

しかし、変更がなくてもレコード後を実行させたい場合があります。例えば、何らかのメッセージを表示させたり、計算処理やログを採取するような場合などが考えられます。

強制的にレコード後を実行させる

1. タスク特性（*Ctrl+P*、またはタスク→タスク特性）を開きます。
2. 動作タブを選択します。
3. 強制レコード後を *Yes* に設定します。
4. 式で動的に指定したい場合は、ズーム（*F5*、ダブルクリック、または編集→ズーム）して式エディタを開き、論理式を定義することができます。



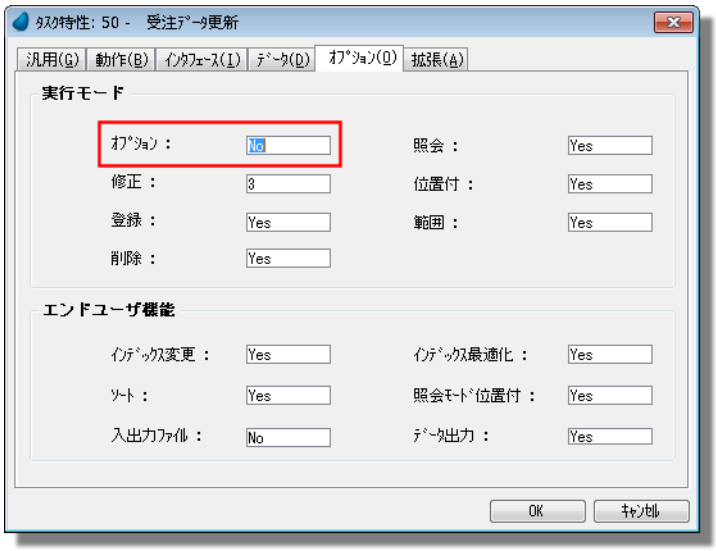
注： 今までレコード後で実行させていたことは、コントロールイベントを使用することで実現できます。コントロールイベントはコントロールに関して最も良いレベルになります。

参照： 第 20 章：「式エディタ内で式の体裁を整えるには」（461 ページ）

エンドユーザがタスクモードを変更することを防止するには

デフォルトでは、Magic xpa はエンドユーザがレコードの作成や削除、修正、照会ができるようにしています。これによって、データビューを簡単に表示させるタスクを作成することができます。しかし、ほとんどのアプリケーションでは、ユーザによってまたは実行するプログラムによって機能を制限するようにしています。例えば、一般のユーザはレコードの参照のみ許可され、管理者だけが修正することができるタスクなどを作成することができます。

ユーザがタスクモードを変更することを防止する

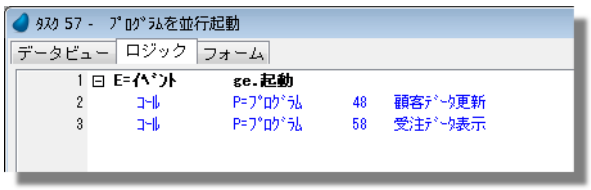


1. **タスク特性**（*Ctrl+P*、または**タスク→タスク特性**）を開きます。
2. **オプション**タブを選択します。
3. **オプション**特性を *No* に設定します。この設定によってユーザがタスクのモードを変更することができなくなります。
   式で動的に指定したい場合は、**ズーム**（*F5*、**ダブルクリック**、または**編集→ズーム**）して**式エディタ**を開き、論理式を定義することができます。

注： このタブ内にある他のオプションを使用して特定の処理の利用を制限することもできます。例えば、**削除**特性を *No* に設定することでユーザはレコードの削除ができなくなります。

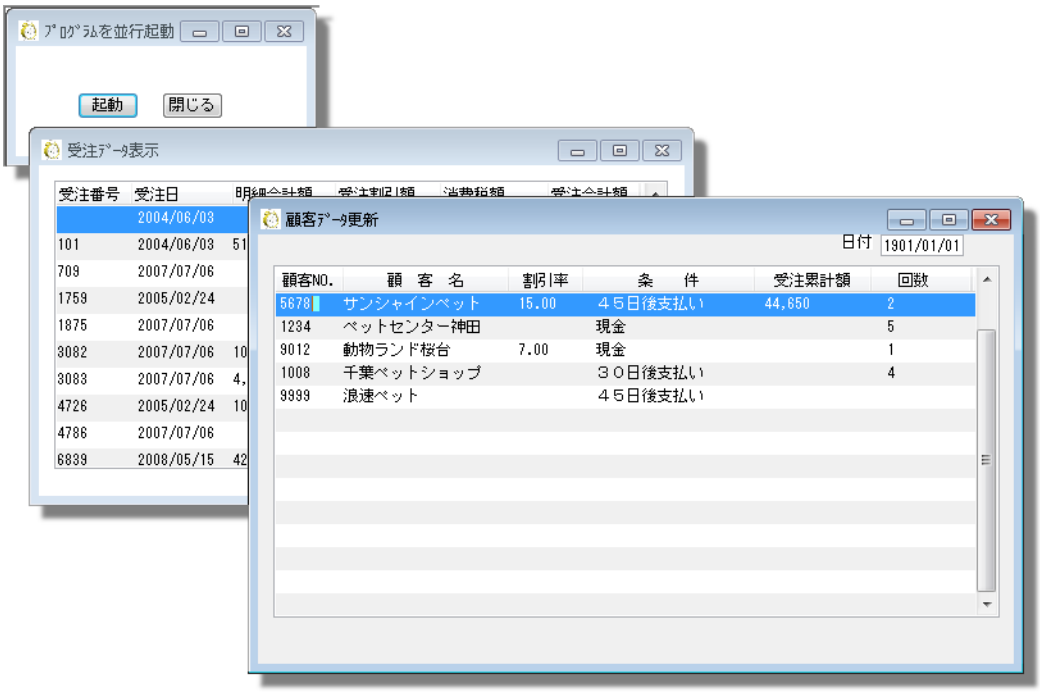
参照： 第 20 章：「式エディタ内で式の体裁を整えるには」（461 ページ）

２つのタスクを並行に実行させるには



通常、ここに表示されているように２つのプログラムを起動すると、Magic xpa は最初のタスクを実行し、ユーザがそのタスクを終了すると２番目のタスクが起動されます。

しかし、この２つのプログラムを同時に実行させることもできます。これは２つのウィンドウが同時にオープンされているように少しづつ動作することで実現しています。これらは両方ともオープンされていますが、各々独立して動いています。どのように独立しているかは Magic xpa の設定に依存します。

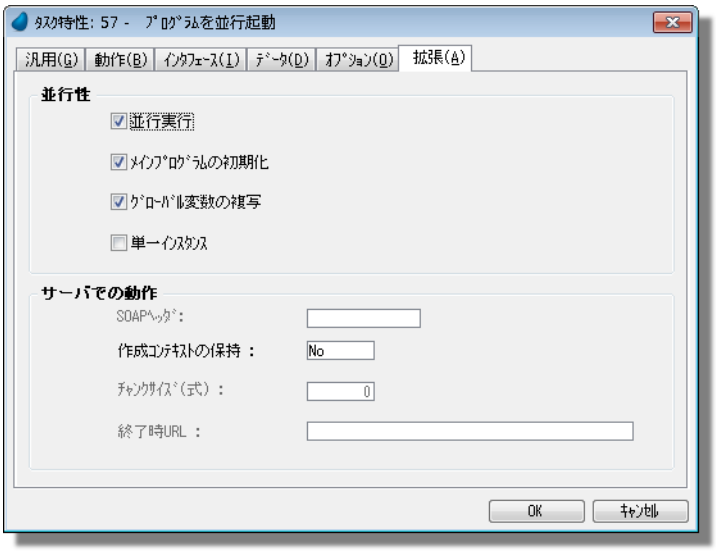


この例では、並行に実行することのできる２つのタスクを設定しています。そして最初のタスクが実行され、次に２番目のタスクが実行されています。これらは両方とも実行されたままの状態になっています。２つのタスクの間でフォーカスを変更し、個別に更新処理を行うことができます。

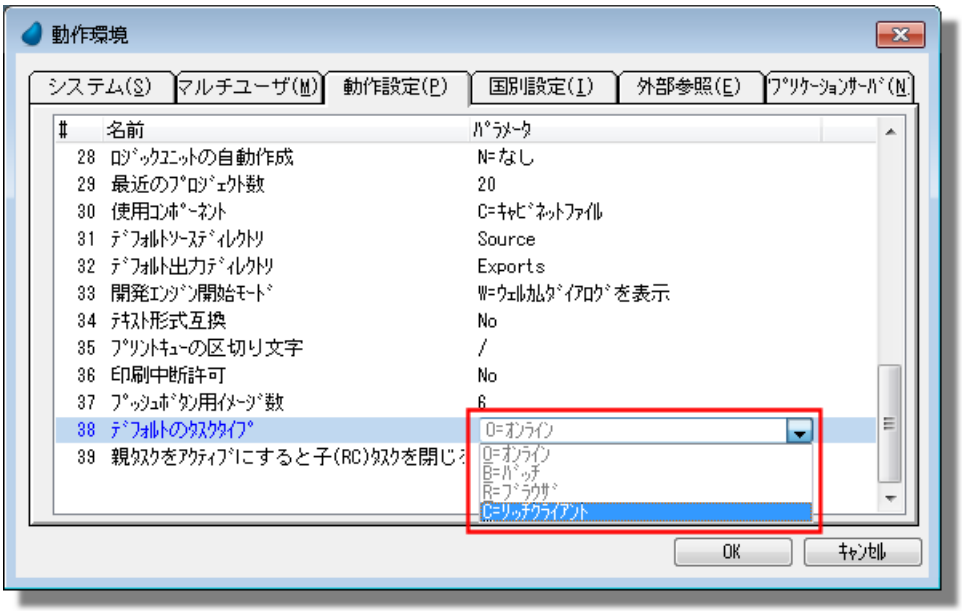
ここではロックを防止するような対応はされていません。各タスクに対して、２人のユーザが同じデータをアクセスするような操作を行えば、同じような結果になります。この例では、同じレコードを２つの異なるウィンドウで表示していますが、表示のみの処理であれば問題は発生しません。

並行実行するようにプログラムを設定する

1. **タスク特性**（*Ctrl+P*、または**タスク→タスク特性**）を開きます。
2. **拡張**タブを選択します。
3. **並行実行**特性のチェックボックスを**チェック**します。
4. 必要に応じて他のチェックボックスを**チェック**します。



タスクタイプのデフォルト値を変更するには



プロジェクトを開発する場合、1 つの特定のタスクタイプを沢山作成する場合があります。たとえば、リッチインターネットアプリケーション（RIA) のプロジェクトを開発している場合、作成する大部分のタスクはリッチクライアント・タスクです。また、インターネット・アプリケーションをサポートするプロジェクトが主にバッチタスクを必要とします。このような場合、新規に作成するタスクのデフォルトタイプが自動的によく使用するタイプに設定されていれば便利です。

タスクのデフォルトタイプを変更する：

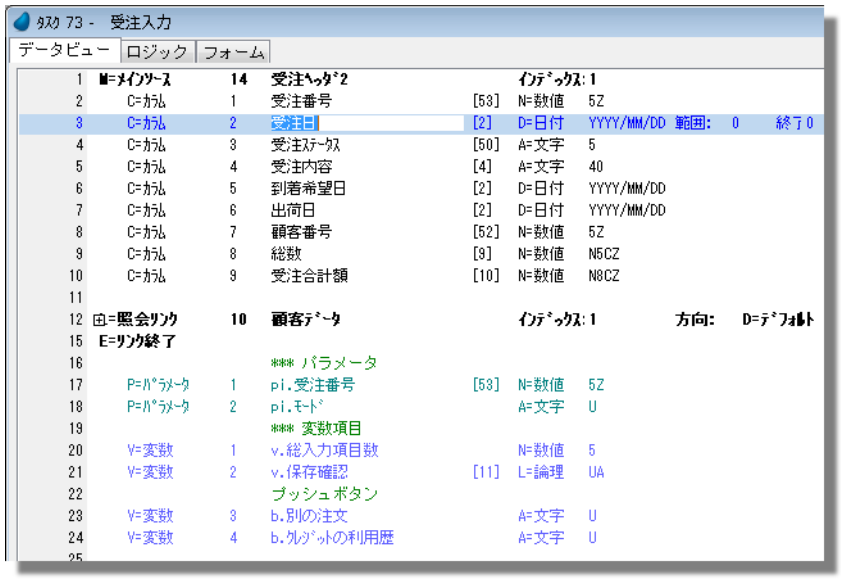
1. **オプション→設定→動作環境→動作設定**を開きます。
2. デフォルトのタスクタイプの行に移動します。
3. プルダウンリストからタスクタイプを選択します。

これで、新しいタスクが作成された場合、常に指定されたタスクタイプがデフォルトになります。

第 5 章： タスク

注： 表示フォーム用のフォームデザイナと出力フォーム用のフォームエディタでは同じような機能が異なる名称で表示されている場合があります。

プログラムのデータビューを定義するには



Magic xpa のタスクで使用されるデータは**データビューエディタ**で定義します。データソースを使用している場合、そのデータソースのすべてのカラムを定義する必要はありません。使用したいカラムのみを定義するだけで十分です。また、必要に応じて変数やパラメータを定義することもできます。

データビューエディタでは、５種類の基本的なタイプのデータを定義することができます。最初の３つのタイプはデータソースをもとに定義します。

- **メインソース / ダイレクト SQL：**メインソースが定義されている場合、タスクはデフォルトでデータソース全体を読み込みます。つまりオンラインタスクであれば、ユーザはテーブルのすべてのカラムを参照することができ、バッチタスクであればテーブルのすべてのレコードをアクセスするようになります。しかし、範囲を指定することで、読み込むレコードを制限したり、1 レコードのみ読み込むようにすることもできます。
- **リンクテーブル：**他のデータソースと 1 対 1 にリンクされるテーブル（ソース）です。この例では、顧客の姓名を表示するために、注文データと照合された顧客レコードを取得しています。データビューを取得するには、テーブルのすべてのカラムが必要なわけではありません。必要なカラムだけ定義すれば十分です。
- **宣言：**現在のタスクでは使用しないがデータソースをオープンしておくことができます。これは下位のタスクがそのデータソースを使用する場合、オープン / クロースの処理をしないようにすることで効率化させるための方法です。

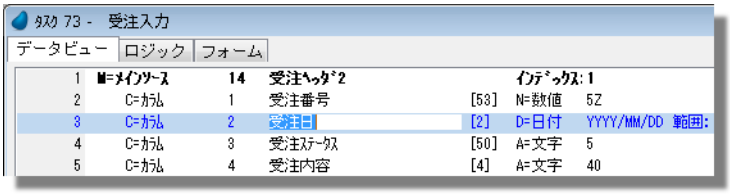
最後の２つのタイプは、データソースに含まれない項目です。

- **パラメータ：**他のプログラムに値を渡すために使用します。
- **変数：**変数項目はタスク実行時にのみ有効な一時的なデータ項目です。

メインソースはデータビューの先頭の定義される必要がありますが、これ以外のデータはどの順番でも定義することができます。しかし、従属関係があることを考慮してください。例えば、リンクテーブルはリンクされるデータソースの下に定義しなければなりません。これは、データを整理する上で最良の方法です。また、コメントや余白を追加することで、ロジックが理解しやすくなります。

以下は、これらのことをどのように行うかを示した概要です。ここには、これ以外にも学ぶ内容がたくさんがあります。また、Magic xpa のマニュアルを参照することで、これらの理解を助けることになるはずです。

メインソースを定義する



1. メインソースのヘッダ行は常に存在しています。これを作成する必要はありません。*Tab* を押下してメインソースが表示されている次のカラムに移動し、アクセスしたいデータソースの番号を入力するか、ズーム（*F5*、またはダブルクリック）を実行して一覧からデータソースを選択します。再度 *Tab* を押下します。
2. カーソルは、**データソース名**のカラム上に位置付けられているはずです。この名前は変更することができます。データソースの名前を変更することは、アクセスするデータリポジトリにも、エンジンがデータにアクセスする際にも影響しません。これは同じデータソースを１つのタスクツリー内で何回も定義するような場合に、ロジックを理解しやすくするする利点があります。再度 *Tab* を押下します。
3. 次にカーソルは、**インデックス**のカラムに移動します。インデックス番号を入力するか、ズームして一覧から選択します。適切なインデックスを選択することは非常に重要です。これによって、レコードを一覧表示させる場合の順番が決定され、範囲指定されている場合の動作の処理速度に影響します。
4. 最後に、マウスの右クリックを行い、コンテキストメニューから特性を選択します。ここでは、データソースをどのようにオープンするかを定義することができます。

メインソースを定義する、*F4* の押下でヘッダ行の下に詳細行が追加され、データソースに定義されているカラムを選択することができます。これらのカラム行では、データビューを限定するための**範囲**条件をメインソースに定義することができます。例えば、１つの注文情報や１人の顧客を表示させたい場合に指定します。

ヒント：メインソースが定義されたタスク内では、リンク/リンク終了内の行を除いて、カラムと指定されたデータビューはすべてメインソースを参照します。これは、データビュー内に複数のリンクテーブルや変数が定義されていても同じです。しかし、メインソースのヘッダ行内にカラムを集めておいた方が、ロジックは見やすくなります。

リンクテーブルを定義する

リンクテーブルの定義は、メインソース同じように行うことができます。

1. 最初にリンクテーブルを作成したい行で *Ctrl+H* を押下してヘッダ行を作成します。
2. 左端のコンボボックスから**リンクタイプ**を選択します。各タイプは異なる動作をします。通常は照会リンクを使用します。*Tab* を押下して次のカラムに移動します。リンク終了はヘッダ行の下に自動的に追加されます。
3. リンクしたいデータソースの番号を入力するか、ズーム（*F5*、またはダブルクリック）を実行して一覧からデータソースを選択します。再度 *Tab* を押下します。
4. **インデックス**カラムでは、リンクレコードの検索に使用したいインデックスを選択します。選択されたインデックスのセグメントとして定義されているカラムが自動的に追加されます。これらを使用してリンクレコードの位置付け定義を行うことができます。
5. マウスの右クリックを行い、コンテキストメニューから特性を選択することでデータソースをどのように処理するかを定義することができます。

リンクと終了リンクの行間で *F4* を押下するとリンクで使用するカラムを選択することができます。**位置付**特性は、どのレコードをリンクするかを制御するために使用されます。この例では、顧客レコード内の顧客 ID と受注ヘッダ内の顧客 ID がリンクされています。

変数とパラメータを定義する

データソースのカラムではないデータ項目を定義することができます。これらの項目には２つのタイプがあります。***変数***と***パラメータ***です。これらの唯一の違いは、パラメータは別のプログラムからデータを受け取るためのものであり、パラメータの数や書式は、起動元のプログラムの定義内容とチェックされるところです。

1. 変数またはパラメータを作成したい行に移動し、***F4***（**編集→行作成**）を押下します。
2. ドロップダウンリストから ***V= 変数***か ***P= パラメータ***を選択します。***Tab*** を押下して次のカラムに移動します。
3. 現在カーソルは、**名前**のカラムにあります。ここには、任意の名前を入力することができます。何らかの命名規約（例えば、変数ならば接頭辞として「v.」を使用する等）を使用することもあるかもしれません。名前を入力後、***Tab*** を押下して次のカラムに移動します。
4. 次に、**モデル**のカラムに移動します。ここから**ズーム**して一覧からモデルを選択できます。モデルを使用することで開発工数が減ったり、エラーの防止やタスクの規格化に役立ちます。モデルを使用する場合は、ここでモデルを選択するだけで十分です。モデルを選択しない場合は、***Tab*** を押下して次のカラムに移動します。
5. 次は、**型**のカラムに移動します。モデルを使用しない場合、型の先頭文字を入力してデータ型（文字、論理、日付、時間など）を選択する、プルダウンリストから選択します。再び ***Tab*** を押下します。
6. ここには、項目の書式を入力するか、**ズーム**して**書式**ダイアログを表示させます。**書式**ダイアログでは、***F1*** キーを押すことで書式に関するヘルプを表示させることができます。
7. ***Alt+Enter*** を押下するか、**特性**シートを**クリック**して**項目**特性に移動します。**項目**特性は、データ項目のサイズや有効値を指定するだけでなく、フォーム上でどのように表示させるかを定義することもできます。モデルを使用した場合、これらの特性値はそのモデルの値が設定されますが必要に応じて、継承された値を上書きすることもできます。

ヒント :*Magic xpa で最も一般的なエラー原因の 1 つに書式を指定しないで項目を作成することがあります。型を選択したら、すぐに特性シートを開き項目の書式を設定してください。データ長が 0 の場合、このプログラムは実行できないかもしれません。*

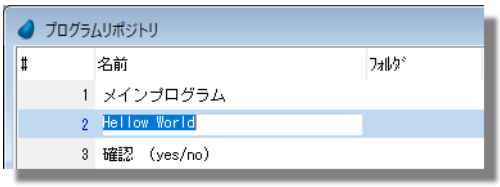
参照 : 第 3 章 :「再利用可能なデータオブジェクトを定義するには」（39 ページ）

簡単なプログラムを作成するには

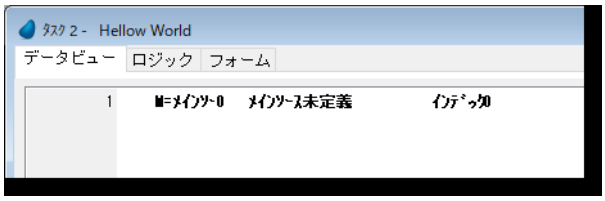
一般的なプログラミングの書籍では、Hello World という簡単なプログラム例もとに説明しています。ここでは、Magic xpa による Hellow World の説明を行います。

ヒント：まだ Magic xpa を始めたばかりの場合は、F1 を押下することでヘルプを表示させることができます。ここには、現在表示されているパラメータに関連したトピックが表示されます。

Magic xpa で「Hello World」を作成する

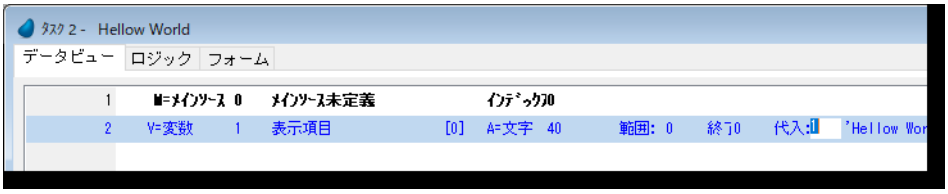


1. プログラムリポジトリに移動し（*Shift+F3*）、プログラムを作成したい行にカーソルを置きます。
2. *F4*（編集→行作成）を押下し一行追加します。
3. プログラムの名前を入力します。この場合は、Hello World です。Magic エンジンはこの名前を実行時は使用しないため、どのような名前でも入力できます。
4. ズーム（*F5*、またはダブルクリック）を押してプログラムを開きます。
5. 新規プログラムのため、タスク特性ダイアログが表示されます。*Esc* か OK ボタンを押下してダイアログを閉じます。簡単なプログラム用にデフォルト値が設定されています。

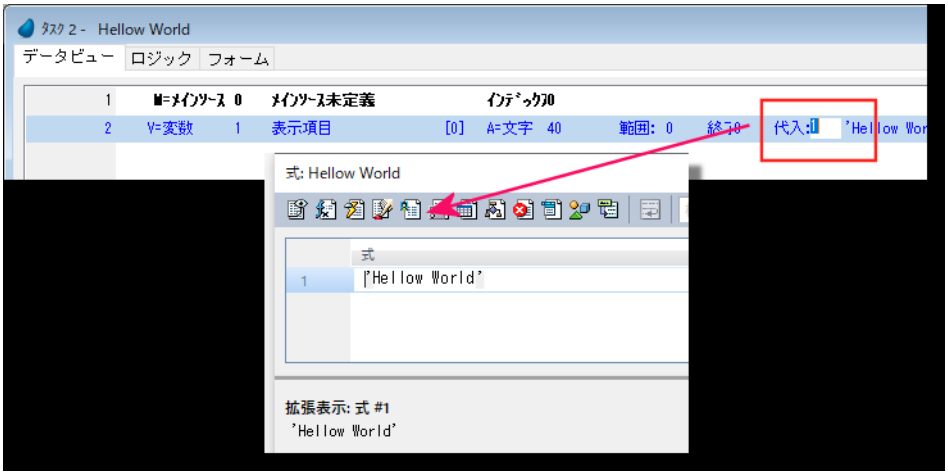


6. 次に作成されたプログラムを見てください。３つのタブが表示されています。データビュー、ロジックそして フォームです。これらは、Magic xpa でプログラムを作成する場所になります。

データビューを作成する



1. 最初に、データビューを定義します。データビュータブをクリックします。メインソースのヘッダ行がすでに定義されていることを確認してください。この例では、メインソースを使用しないためこの設定を無視してください。
2. *F4* を押下し一行作成します。
3. プルダウンリストから ***V= 変数***を選択します。これは変数項目を作成するという意味です。*Tab* で次のカラムに移動します。
4. 変数の名前を指定します。ここでは表示項目と入力します。
5. *Tab* を２回押下して **A= 文字**と表示されているカラムに移動します。ここはデータの型を指定します。デフォルトは文字型です。プルダウンをクリックすると他の選択肢をみることができます。もう一度 *Tab* を押下します。**書式**を指定するカラムに移動します。ここに *40* と入力します。これは 40 桁の文字項目を意味しています。**代入**と表示されたカラムまでカーソルを移動します。



6. **代入**特性は、タスク実行時に初期設定する値や実行時に再計算される値を指定します。この場合は、*Hello World* という文字列で初期設定します。*F5* を押下するか**ズーム**して**式エディタ**を開きます。
7. **式エディタ**では、*F4* を押下して一行追加し、*'Hello World'* という文字列を入力します（シングルクォーテーションも含めます。）。**OK** を**クリック**します。
8. **式エディタ**を閉じると**代入**特性に *1* という数字が表示されます。これは式番号 **#1** が定義されていることを意味しています。また、右側に式の内容の先頭部分が表示されます。

ヒント：このような簡単な式の場合は、代入カラムに = を入力することでも設定できます。この場合、簡単な入力ボックスが表示され、ここに式の内容を入力します。

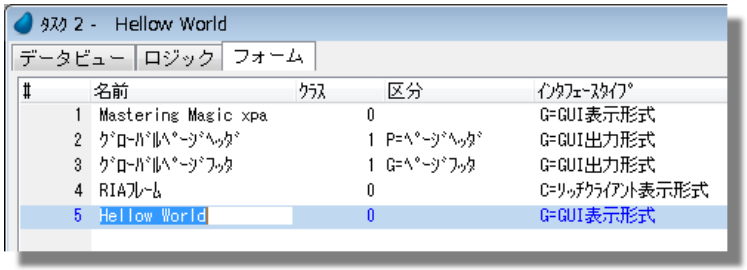
これでこのプログラムに対して**データビューエディタ**で行う作業は終了しました。

ロジックを作成する

このプログラムでは、**ロジックエディタ**で何かを行う必要はありません。Magic xpa は、通常のプログラムで行うようなほとんどの処理をエンジン内で行うため、そのための明示的なロジックを作成する必要はありません。

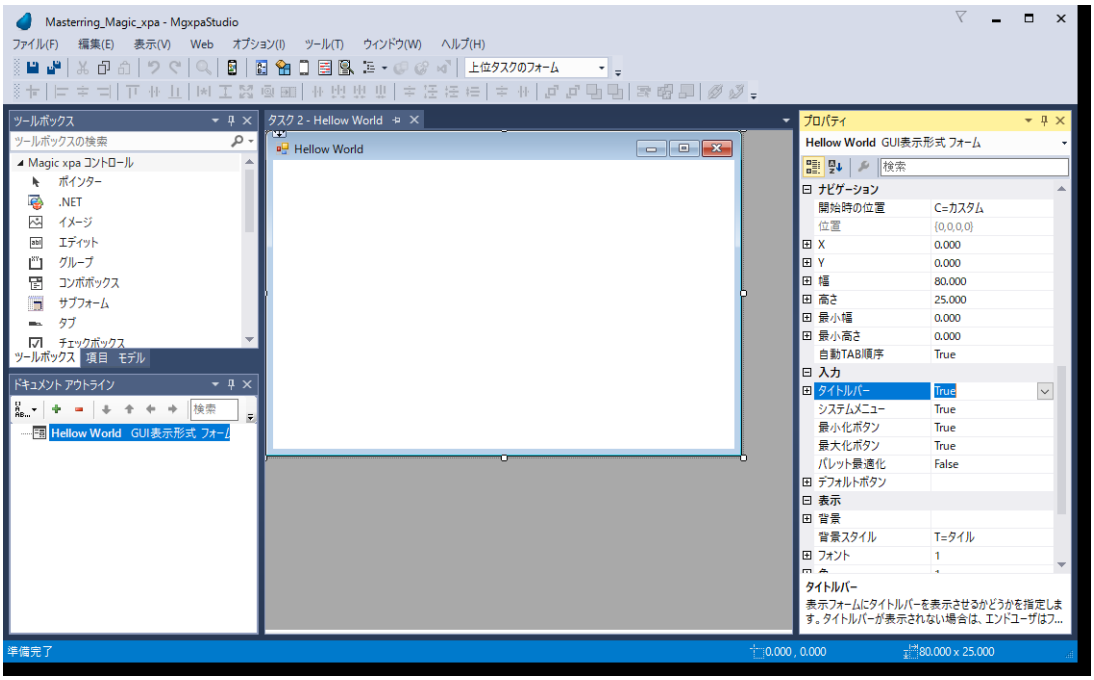
ここでは、データ項目の検証や他のプログラムを呼び出すなどの処理を定義することができます。Magic xpa でロジックとイベントを使用する方法については、第 4 章：「Magic エンジンをイベント駆動型で動作させるには」（50 ページ）を参照してください。

表示画面を作成する



1. **フォーム**タブを**クリック**します。すでに **Hello World!** という名前のフォームが作成されているはずです。デフォルトではタスクの名前と同じ名前がフォームに設定されますが、必要に応じて変更することができます。またデフォルトでは、この名前がウィンドウのタイトルバーに表示されます。
2. **名前**カラムで**ズーム**（*F5*、または**ダブルクリック**）すると、フォームの内容が表示されます。この時点では基本的な空のウィンドウが表示されます。このウィンドウは位置を変更したり**ドラッグ**することでサイズを変更することができます。**特性**シート（*Alt+Enter*）には、このウィンドウの表示内容を変更するための色々なパラメータがあります。しかしここではデフォルトのままにしておきます。

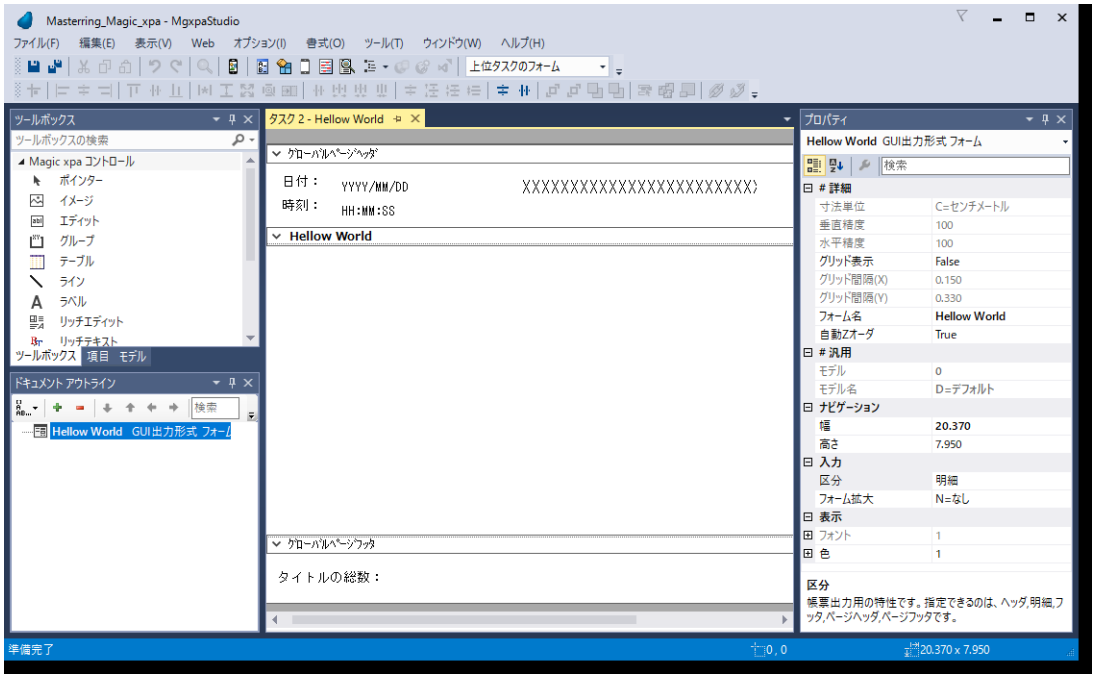
表示フォーム



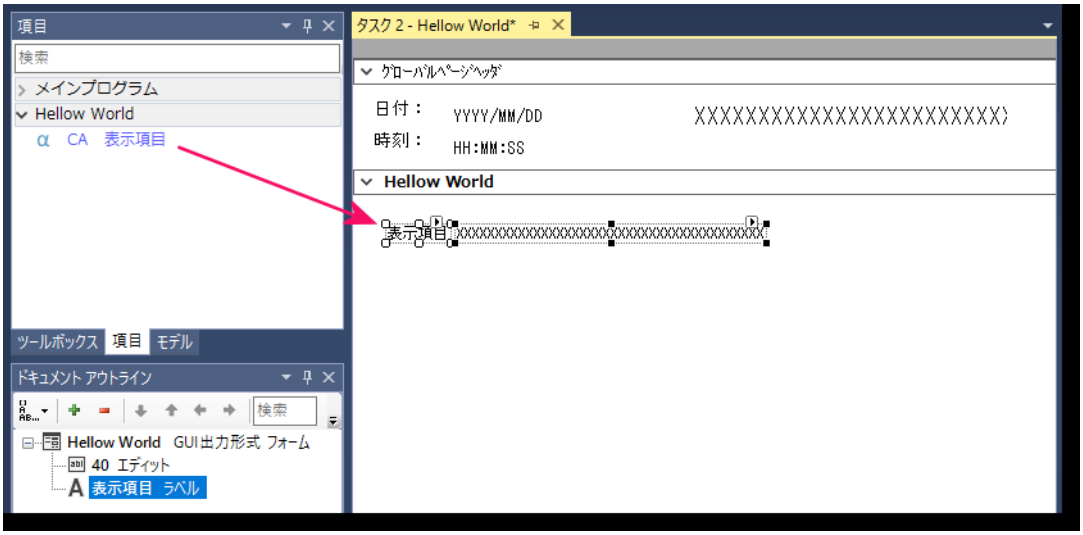
3. フォームを表示する場合は、以下が含まれている**フォームデザイナ**を使用します。
   - **レイアウトツールバー** …… コントロールの位置やサイズなどを揃えるためのコマンドを使用することができます。
   - コントロールの**ツールボックス**ペイン …… ここには、フォームに追加することができるコントロールが表示されます。
   - **ドキュメントアウトライン**ペイン …… ここには、フォームに存在するコントロールが表示されます。コンテナから別のコンテナにコントロールを移動する場合に使用することができます。
   - **項目**と**モデル**のペイン …… ここには、データビュー内の項目とモデルが全て表示されます。

   Magic xpa でフォームを編集している間は、これらのペインを使用します。同じオプションが、メニューやショートカットキーで利用することもできます。オプションやショートカットキーに慣れると便利です。
4. 何かの理由でペインがワークスペースに表示されていないのであれば、[表示] メニューで該当するペインを選択してください。
5. 項目をフォームにドラッグ＆ドロップしてください。フィールドがそのフィールドのプロンプトと一緒に配置されます。デフォルトでは項目名になります。変数名を慎重に定義すれば、フォームの作成のための工数を減らすことができます。

出力フォーム



6. 出力フォームの場合も、[フォーム] デザイナを使用します。[項目] ペインのタブをクリックします。ここには、データビューとしてタスクがアクセスできるすべての項目が表示されています。ここでは 1 つだけ表示されています
7. コントロールパレットの項目タブをクリックします。ここには、データビューとしてタスクがアクセスできるすべての項目が表示されています。ここでは 1 つだけ表示されています。



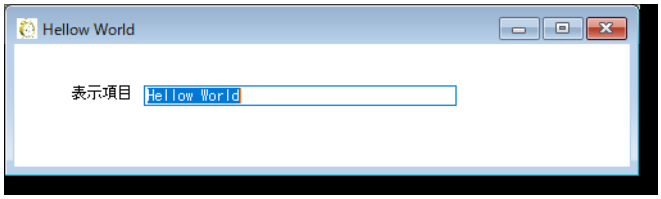
8. 項目をクリックします。カーソルがボックス表示に変わり、選択されていることを示すようになります。フォームをクリックして項目をドロップします。項目が貼り付けられます。項目名も [ラベル] コントロールとして貼り付けられます。このことを考慮して項目の名前を定義していれば、フォーム作成の工数を減らすことができます。

フォームエディタでの作業はこれで終わりです。オプション→オブジェクトを保存 / 閉じるを選択して、プログラムリポジトリに戻ります。

プログラムを実行する

1. 作成したプログラムを実行する前に、*F8*（オプション→構文チェック）を押下します。プログラムに問題がなければ、ステータス行に「プログラムは正常です。」というメッセージが表示されます。何らかの問題点があれば、チェック結果ペインにエラーメッセージが表示されます。どのようなプログラミング言語であれ、プログラムに問題があれば、実行させても予期しない結果になります。プログラムを実行する前にはエラーをチェックして問題がないことを確認する必要があります。

2. プログラムを実行するには ***F7***（**デバッグ→実行**）を押下します。別のウィンドウが表示され、実行エンジンによってプログラムが起動されます。プログラムが実行中は、開発環境はロックされ読み込み専用の状態になります。

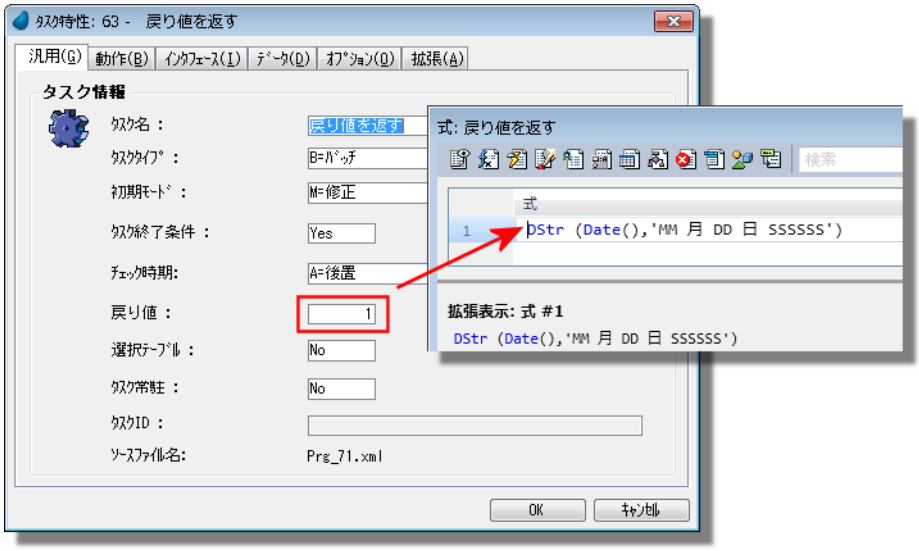


3. これで Hello world! プログラムの作成が終わりました。プログラムを終了すると開発エンジンの環境に戻り、プログラム作成を継続したり、実行し直したりすることができます。このようにして Magic xpa を使用することでプログラムのインクリメンタル開発を行うことができます。
   またデバッグ機能も利用できるため、プログラムが実行中にエンジン内でどのような処理が行われているかを確認することができます。

起動元のプログラムに戻り値を返すように設定するには

起動したプログラムに戻り値を返すには２つの方法があります。１つはパラメータを使用する方法です。この方法については、「起動元と起動先のパラメータを同期させるには」（89 ページ）を参照してください。もう１つは戻り値を使用する方法です。

式を使用して Mahc プログラムを起動したり、Magic xpa 外のプログラムを呼び出す場合に戻り値が必要になります。ここに、どのように設定するかを示しています。



戻り値を定義する

1. 起動されるプログラムを選択し**ズーム**で開きます。**タスク特性**の**戻り値**特性（*Ctrl+P*）を選択します。戻り値としたい値を返す式を定義します。

この場合は、１回だけループを繰り返すバッチタスクで、今日の日付を指定された文字型書式で返すようにしています。もし書式を変更する必要があれば、このプログラムを変更するだけで済むようになります。

戻り値を使用する

1. 式を定義することのできるプログラムであれば、どれでも使用することができます。この例では、**代入**カラムに使用しています。**CallProg()** 関数は、プログラムを呼び出し、定義された文字項目に値が自動的に返ります。

   CallProg() 関数はプログラム番号を使用して呼び出します。もし、**#36** のプログラムを別の場所に移動したらどうなるでしょうか？ **PROG** リテラルを使用することでこのような心配は不要になります。プログラム番号が変更された場合、式内の番号が自動的に変更されるようになります。

   プログラムの**公開名**を使用して **CallProg()** 関数で呼び出すこともできます。この場合の構文は以下のようになります。

   ```
   ’今日の日付は、’ & CallProg(ProgIdx(’DateString’,’TRUE’LOG))
   ```

   CallProg() や **ProgIdx**() 関数についての詳細は、*F1* を押下して表示されるヘルプを参照してください。

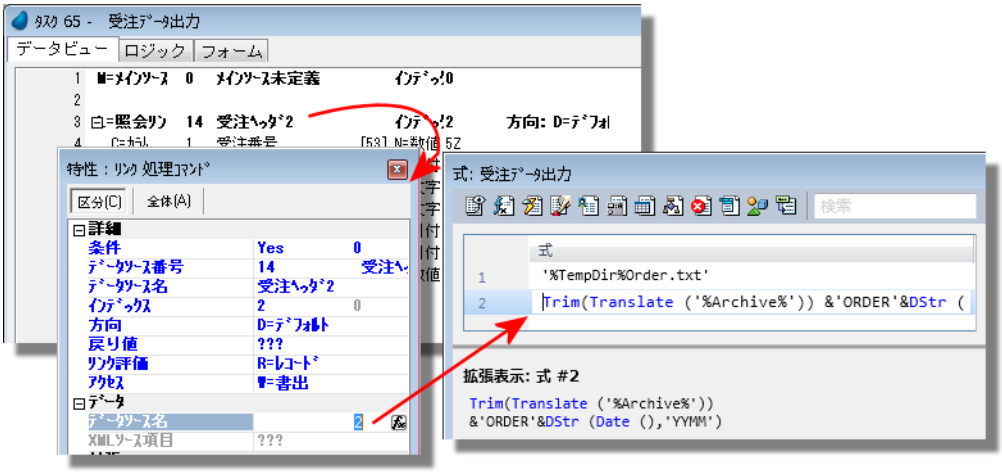
注： Magic xpa にはプログラムを標準化するための様々なオプションがあります。この例の場合は、日付書式の**項目モデル**や**メインプログラム**に処理を定義するようなことが考えられます。

データリポジトリに定義されている名前と異なるデータソース名を使用するには

通常、データソース名はデータリポジトリで設定されます。これによって保守が容易になります。一カ所で管理されていることで、データソース名を探したり、変更することが容易になります。

しかし、タスクレベルでデータソース名を変更したい場合があります。例えば、ユーザ毎に個別のテーブルを定義したり、同じデータ定義で日付によって異なるデータソース名にするようなことが考えられます。

タスクレベルでデータソース名を指定する

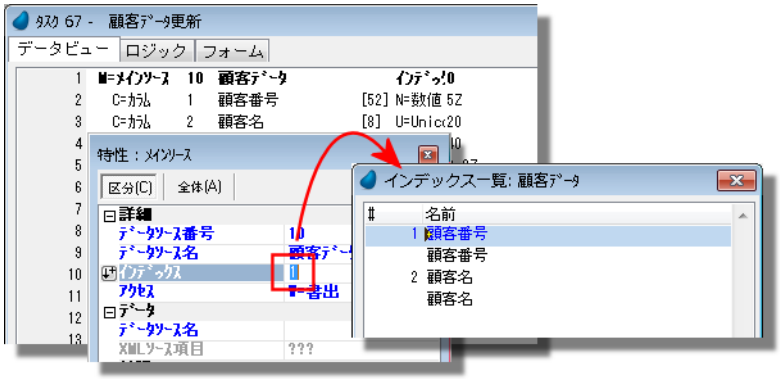


この例では、受注用のヘッダレコードを複写するサブタスクがすでに定義されているものとします。ここではデータリポジトリ内の同じオブジェクトを開いてはいますが、**データソース名**特性を指定することで、デフォルトのデータソース名を上書きしています。データソース名は、実行時の年と月にもとづいて定義され（例えば、2007 年 12 月の場合は、'0712'）、論理名 %Archive% で指定されたディレクトリ内に作成 / 修正されるようになっています。

注： 同じタスク内では、同じデータソースに対して異なる名前を定義することはできません。この例では、親タスクがデフォルト名で同じデータソースを開くようになっているため、サブタスク内では保管用のレコードを作成しています。

参照： 第 17 章：「既存のデータベーステーブルにアクセスするには」（400 ページ）

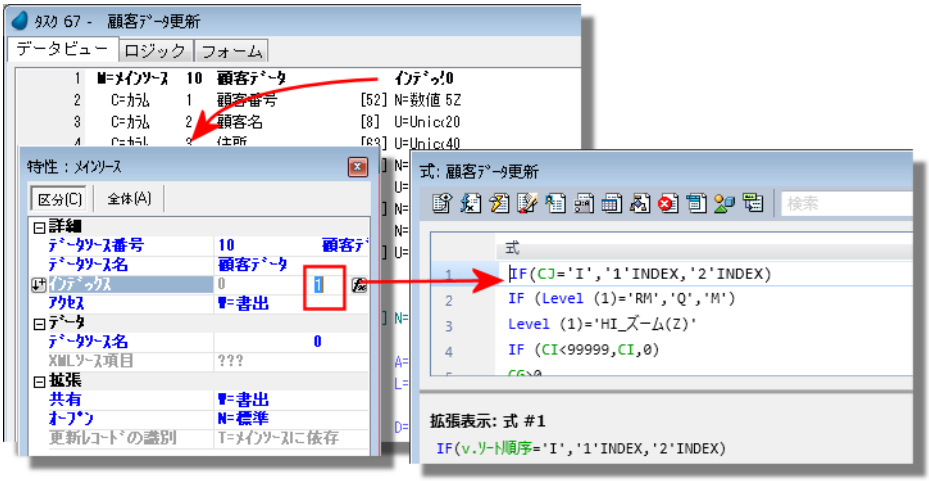
レコードの表示順を動的に変更するには



レコードを表示する順番は２つの方法で定義できます。１つはソートテーブルで実行時のソートの順番を定義する方法であり、もう１つは最もよく使用される方法ですがメインソースの**インデックス**特性を変更することです。この例では、インデックスは *1* が指定されています。この場合は、顧客 ID を元にソートされます。

式を使用することで動的にインデックスを設定することもできます。ユーザがテーブルの表示順を変更できるようにする場合や、帳票出力時に選択したフィルタを元に処理を最適化する場合に使用します。

インデックス用の式を使用する



1. **データビューエディタ**に移動し、**メインソース**に位置付けます。*Alt+Enter* を押下します。**メインソース特性**にカーソルが移動します。
2. **インデックス**特性に *0* を入力します。これで、右側に *Tab* 移動することができます。ここには、インデックスを式で定義することができます。
3. インデックスの**式**特性で**ズーム**するか *fx* ボタンを**クリック**します。
4. インデックス式を入力します。式では、ソート順が I の場合インデックス番号を *1* に設定し、それ以外の場合は、2 に設定するようにします。インデックスの位置がデータソース内で移動する可能性を考慮して **INDEX** リテラル（例：'I'INDEX）を使用することで、設定内容が維持されます。
5. OK を**クリック**すると**特性**シートに戻り式番号が設定されます。

注： ユーザが表示されたレコードのソート順を変更するには、この他に**テーブル**コントロールのカラムヘッダを**クリック**したり、実行時のオプションとして *Ctrl+K*（**オプション→インデックス**）を押下するしたり、*Ctrl+T*（**オプション→ソート**）を押下して独自のソート順を定義する方法があります。

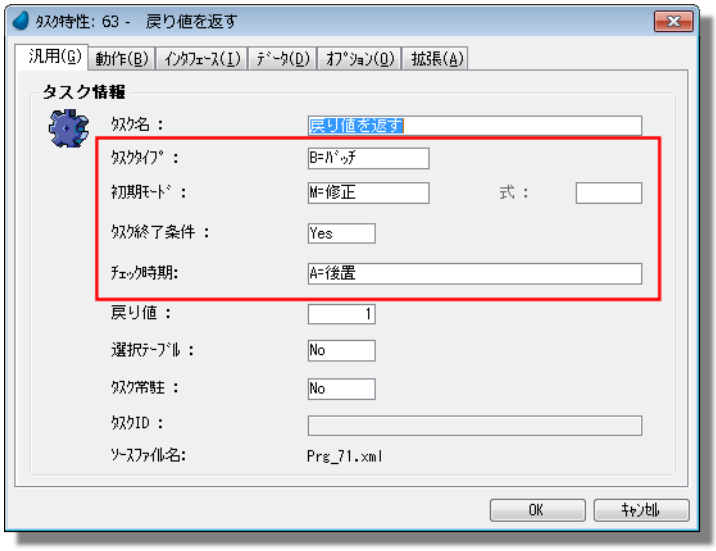
参照： 第 19 章：「タスクで処理されるレコードの順番を動的に設定するには」（431 ページ）

簡単なバッチプログラムを作成するには

ここには、２種類の簡単なバッチプログラムがあります。

- データソース内のすべてのレコードを読み込むプログラム。一般的に、ファイルに出力するような目的で使用されます。このようなプログラムは、**APG**（*Ctrl+G*）を使用して作成することができます。第 5 章：「データソースの内容をテキストファイルに出力したり、テキストファイルをデータソースに入力するには」（85 ページ）も参照してください。
- 一回のみループするだけのプログラム。通常は、戻り値を返すようになっています。ここではこのようなプログラムの作成方法について説明します。

簡単なバッチタスクを作成する



1. **プログラム**リポジトリで、*F4* を押下し一行追加します。
2. プログラムに名前を入力します。この名前はアプリケーション実行時、Magic xpa は使用しないため任意の名前を使用することができます。
3. プログラム名で**ズーム**（*F5*）します。新規プログラムとなるため、**タスク特性**ダイアログが表示されます。
4. デフォルトの**初期モード**特性は ***M= 修正***になっています。これは一般的によく使用されるモードです。照会モードに設定した場合、レコードは更新できなくなります。
5. **タスクタイプ**特性では ***B= バッチ***を選択します。
6. **タスク終了条件**特性を *Yes* に設定します。これは、タスクが永久ループすることを防止するためです。この設定により、1 回だけループするようになります。ループする回数を指定したい場合は、ここでズームしてタスクが終了するための条件式を定義します。
7. **チェック時期**特性を ***A= 後置***に設定します。

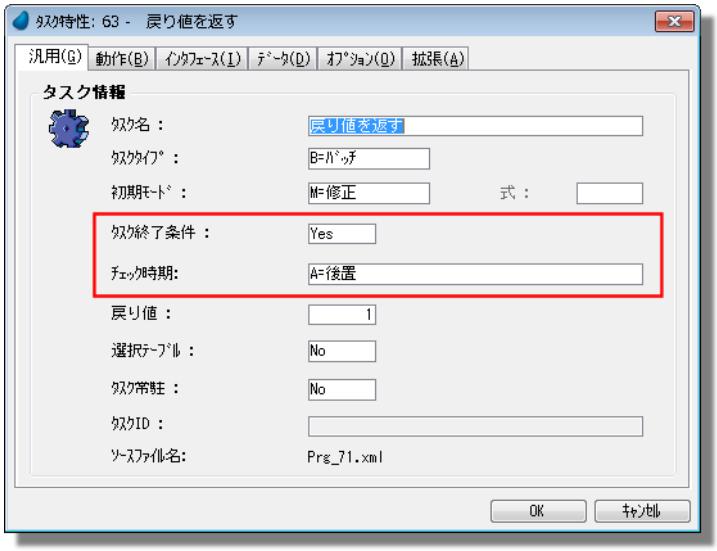
これはバッチプログラムなので、デフォルトでは、ウィンドウは表示されません。処理に時間がかかるようであれば、処理中であることを表示するウィンドウを表示する必要があるかもしれません。

この後、このプログラムに必要なロジックを定義することができます。レコードを更新したり、パラメータとしてデータを返したり、ウィンドウに表示させたり、戻り値としてデータを返したりすることができます。

参照： 「起動元のプログラムに戻り値を返すように設定するには」（79 ページ）
第 21 章：「データをテキストファイルとして出力するには」（487 ページ）

永久ループするバッチタスクを止めるには

デフォルトでは、バッチタスクは永久ループするようになっています。



永久ループすることを防止するには、以下のどれかの設定が必要です。

- **データビューエディタ**でメインソースを設定します。メインソースが選択された場合、タスクはメインソースのレコードのアクセスが終了するとタスクは終了します。
- 右図で示されているように、**タスク特性**で**タスク終了条件**特性を設定します。タスクを 1 回だけループさせるには以下のように設定します。

  タスク終了条件：*Yes*

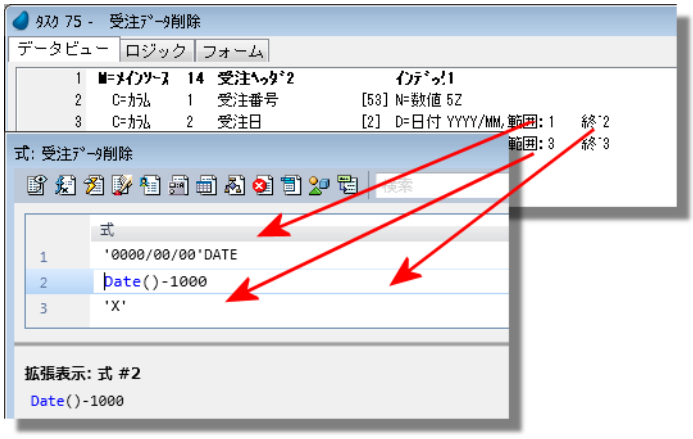
  チェック時期：*A= 後置*
- これ以外の条件でタスクを終わらせたい場合は、**タスク終了条件**特性から**ズーム**して式を定義します。式が True と評価された場合、タスクは終了します。

データソースからレコードをまとめて削除するには

データをまとめて削除するプログラムが必要になる場合があります。データベースを削除したり、レコードを保存用テーブルに移動するような場合に使用されます。

削除用のバッチタスクを作成する

1. 必要な場所（サブタスクか個別のプログラムかに関わらず）でタスクを作成します。
2. **タスク特性**ダイアログを開きます。（*Ctrl+P*）
3. 以下の特性を設定します。
   タスクタイプ：***B= バッチ***
   初期モード：***D= 削除***
4. OK をクリックします。
5. 次に**データビューエディタ**を開きます。レコードを削除したいデータソースをメインソースとして選択します。この場合に、受注ヘッダを選択します。



6. 削除するレコードを選択するために必要なカラムを定義します。この場合に、必要なカラムは注文状況のみですが、デバッグのために他のカラムも定義します。
7. **範囲（最大 / 最小）**特性で**ズーム**し、削除するレコードの範囲を定義します。この場合は、0 から 1000 日以前の日付を持つすべてのレコードが削除されます（削除対象のレコードには、受注ステータスに「X」が設定されています）。値が設定されていない場合、デフォルト値として '0000/00/00'DATE が設定されていることと同じ意味を持つため、最小値の設定は必要ありません。

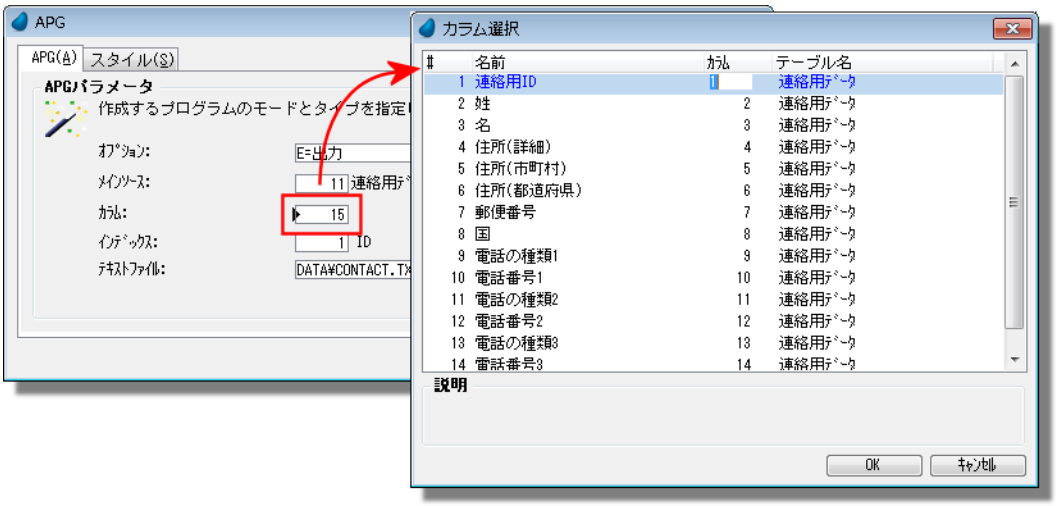
このタスクを実行すると、範囲内のレコードはデータソースから削除されます。

データソースの内容をテキストファイルに出力したり、テキストファイルをデータソースに入力するには

非常によく使用されるタスクの１つに、データソースの内容をテキストファイルに出力したり、テキストファイルの内容をデータソースに読み込むものがあります。Magic xpa はこれらの処理を簡単に行うための機能があります。

データソースの内容をテキストファイルに出力する

1. *F4*（編集→行作成）を押下して、プログラムリポジトリに一行追加します。
2. *Ctrl+G*（オプション→ APG）を押下します。



3. APG ダイアログが表示されます。以下のオプションを設定します。

   オプション：*E= 出力*

   メインソース：出力したいデータソース。ズームすることで一覧から選択できます。

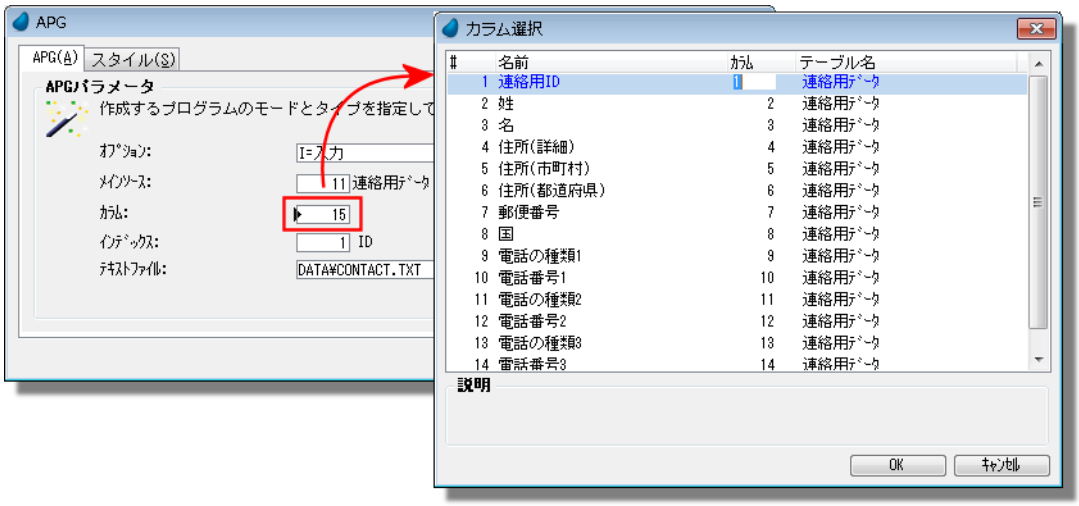
   カラム：ズームして出力するカラムを選択することができます。デフォルトでは、すべてのカラムを出力するようになっています。この例では、ほとんどのカラムの値は 0 のままなので、姓、名、電話タイプ 1、および電話番号 1 だけを選択します。

   テキストファイル：出力するテキストファイルの名前を指定することができます。
4. OK をクリックします。

作成されたプログラムを実行すると、メインソースからすべてのレコードが出力されます。このプログラムを修正することで、出力する書式を変更したり、範囲を指定して出力されるレコードを制限したりすることができます。

テキストファイルの内容を読み込む

1. *F4*（編集→行作成）を押下して、プログラムリポジトリに一行追加します。
2. *Ctrl+G*（オプション→ APG）を押下します。



3. APG ダイアログが表示されます。以下のオプションを設定します。

オプション：*I= 入力*

メインソース：入力先のデータソース。ズームすることで一覧から選択できます。

カラム：ズームして入力するカラムを選択することができます。デフォルトでは、すべてのカラムを出力するようになっています。

インデックス：ズームして入力時に使用するインデックスを選択することができます。入力時の順番は、入力ファイルの並びに依存するため変更する必要はありません。

テキストファイル：入力するテキストファイルの名前を指定することができます。作成されたプログラムでは、データ項目や論理名を使用して設定することができます。

4. OK を**クリック**します。

作成されたプログラムを実行すると、テキストファイルの内容がメインソースのレコードとしてが入力されます。入力するテキストファイルの書式に合うように入力フォームの内容を修正する必要があります。

APG によって作成されたプログラムはレコードの重複チェックを行いません。このため、空のデータソースで実行されていたり、データの重複入力ができなかったり、DBMS からのエラーメッセージを受け取った場合の扱いは開発者に依存しています。

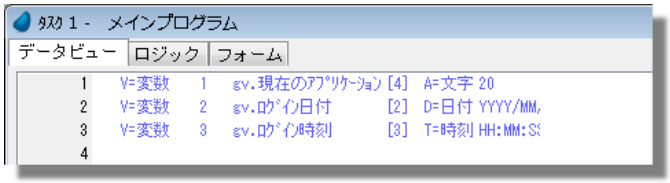
ヒント：もしカラムのレイアウトをデフォルトのままで出力プログラムを作成した場合、入力プログラムもデフォルトのままで作成できます。このため、ある場所から別の場所にデータをコピーするだけの簡単な方法として、APG を使用することができます。

プロジェクト内のどこでもアクセス可能なグローバル変数を定義するには

Magic xpa のタスク内で定義された変数やデータソースは、このタスクおよびそのサブタスクでのみアクセスできます。しかし、プロジェクト内のすべてのタスクで利用可能な変数が必要な場合があります。

これらのグローバル変数は、メインプログラムで定義されることを除いて他の変数と同じように定義することができます。

グローバル変数を定義する



1. プログラムリポジトリを開き（*Shift+F3*）、メインプログラム（**#1**）に移動します。
2. ズーム（*F5*）してプログラムを開きます。
3. データビュータブをクリックします。
4. 他のタスクと同じように、データビューを作成するための操作を行います。

***ヒント :** メインプログラムでもデータソースをリンクすることができますが、メインプログラムは、アプリケーション実行中オープンされた状態のままになるため、多くのユーザが使用するデータソースをここに定義しないようにしてください。メインプログラム内で変数を定義し、タスク前やタスク後でデータを取り出したり、格納することでデータの内容を保持することができます。*

参照 : 「プログラムのデータビューを定義するには」（71 ページ）

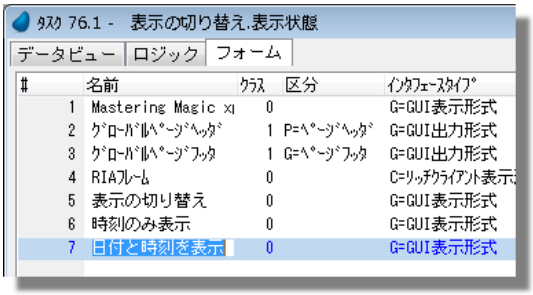
表示するメインフォームを動的に指定するには

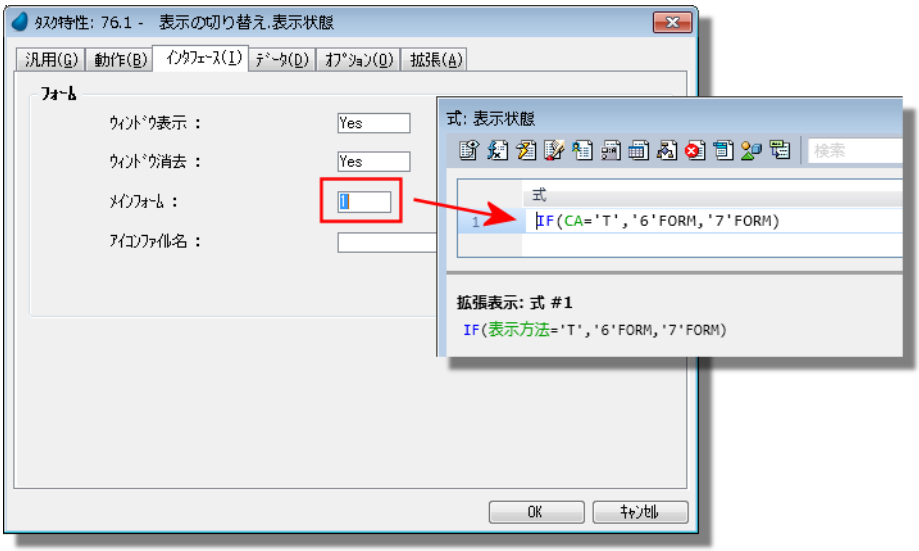
通常各タスクは、表示用の独自のフォームを持っています。しかし、実行時に別のフォームを表示させることができます。

実行時に表示するフォームを指定する

必要条件： フォームはすでに作成されおりて、このタスク（親タスクでない）に定義されているものとします。

1. フォームタブをクリックします。フォームの一覧が表示されているはずです。これらのフォームのいくつかがこのタスクで定義されたものであり、その他は親タスクで定義されたものです。この例では、下位の２つのフォーム（**#3** および **#4**）がこのタスクで定義されたものです。１つはログイン日付と時間を表示するものです。もう一つは、時間を表示するだけのものです。
2. タスク特性ダイアログを開きます（*Ctrl+P*、またはオプション→タスク特性）。インタフェースタブをクリックします。
3. メインフォーム特性でズームし実行時に表示させるフォーム番号を表す式を指定します。この場合に、3 または 4 のどちらかを選択するようにしています。



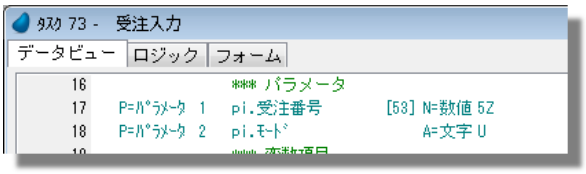


式はここに示されているように、**FORM** リテラルを使用してシングルクォートで囲むように指定する必要があります。これにより、フォームテーブル上のフォームが追加 / 削除された場合もフォーム番号が連動するようになります。

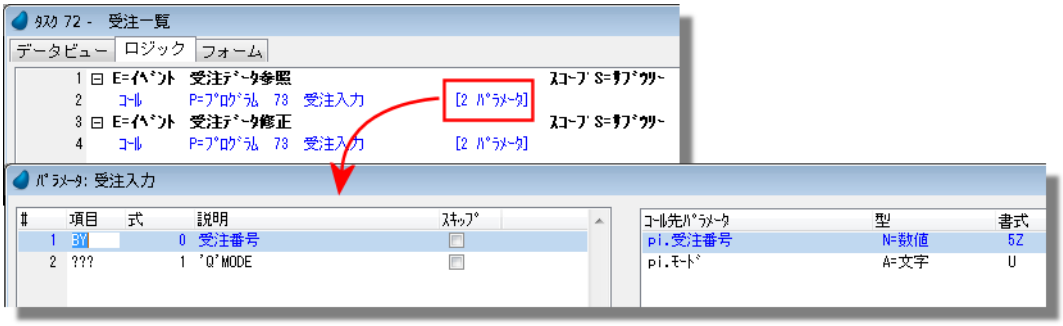
この例において、２つの異なるフォームが、ユーザの選択内容に応じて使用されることになります。

起動元と起動先のパラメータを同期させるには

データビューで**パラメータ**として項目を定義した場合、Magic xpa は起動先プログラムと起動元プログラムの間で自動的に値を同期させようとします。例えば、表示されているように３つのパラメータが定義されいるものとします。



このプログラムを起動している場合、**パラメータ特性**を**クリック**すると以下のように表示されます。



２つのリストが**パラメータ**ダイアログで合っていない場合、**構文チェック**でエラーメッセージが表示されます。渡すか渡さないかを任意に指定できるパラメータの場合、**スキップ**オプションをチェックすることでエラー扱いにならなくなります。

起動されるプログラムのパラメータは、項目名と型、書式が表示されます。

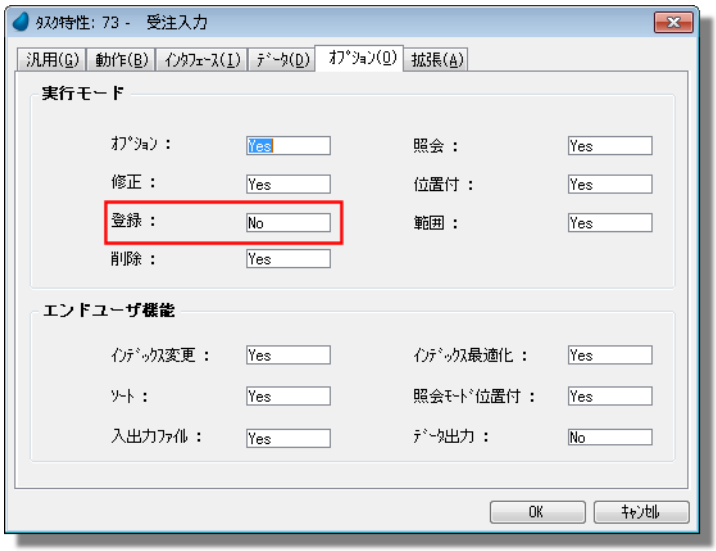
ヒント : *パラメータ項目が定義されていないプログラムにパラメータを渡すこともできます。これは、下位互換を保証するためのものでこのようなプログラムを作成することは推奨しません。これらの変数項目をパラメータ項目に変更し、Ctrl+F（編集→検索と置換→クロスリファレンス）でクロスリファレンスを起動し、このプログラムを起動しているプログラムが正しくパラメータを渡しているかを確認する必要があります。*

エンドユーザがレコードを追加することを防止するには

Magic xpa のタスクは、デフォルトでユーザがレコードの追加、削除、修正を行うことを可能にしています。

ユーザが新しいレコードを追加することを防止したい場合、以下のようにします。

ユーザが登録モードに切り替えることを防止する



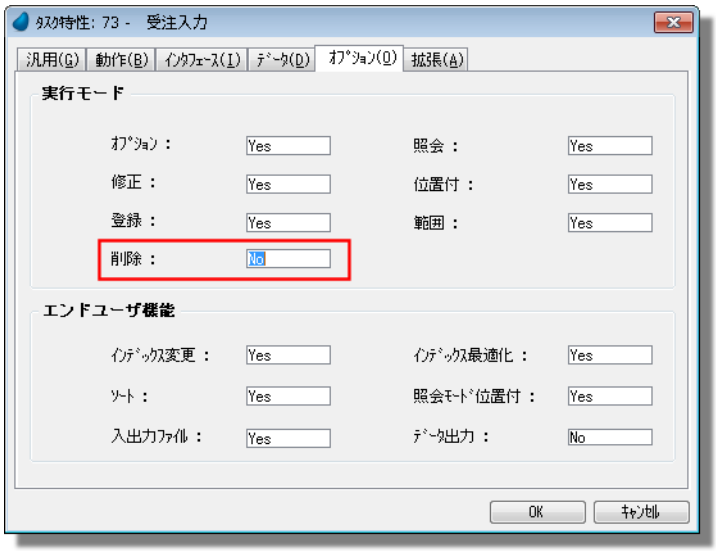
1. **タスク特性**（*Ctrl+P*）を開きます。
2. **オプション**タブを**クリック**します。
3. **登録**特性で、*No* を入力します。
4. ここで**ズーム**（*F5*、または**ダブルクリック**）して式を定義することもできます。実行時に定義された式が False と評価された場合、ユーザは登録モードに切り替えることができなくなります。

エンドユーザがレコードを削除することを防止するには

Magic xpa のタスクは、デフォルトでユーザがレコードの追加、削除、修正を行うことを可能にしています。

ユーザが既存のレコードを削除することを防止したい場合、以下のようにします。

ユーザが削除モードに切り替えることを防止する



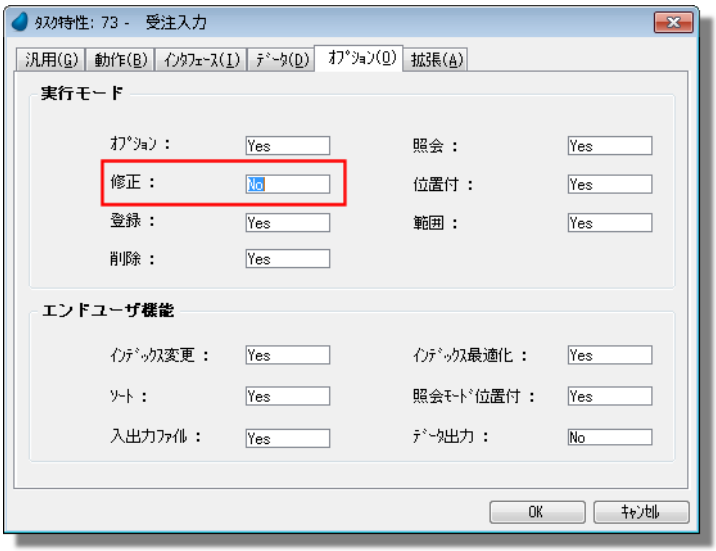
1. **タスク特性**（*Ctrl+P*）を開きます。
2. **オプション**タブを**クリック**します。
3. **削除**特性で、*No* を入力します。
4. ここで**ズーム**（*F5*、または**ダブルクリック**）して式を定義することもできます。実行時に定義された式が False と評価された場合、ユーザはレコードを削除することができなくなります。

エンドユーザがレコードを修正することを防止するには

Magic xpa のタスクは、デフォルトでユーザがレコードの追加、削除、修正を行うことを可能にしています。

ユーザが既存のレコードを修正することを防止したい場合、以下のようにします。

ユーザが修正モードに切り替えることを防止する



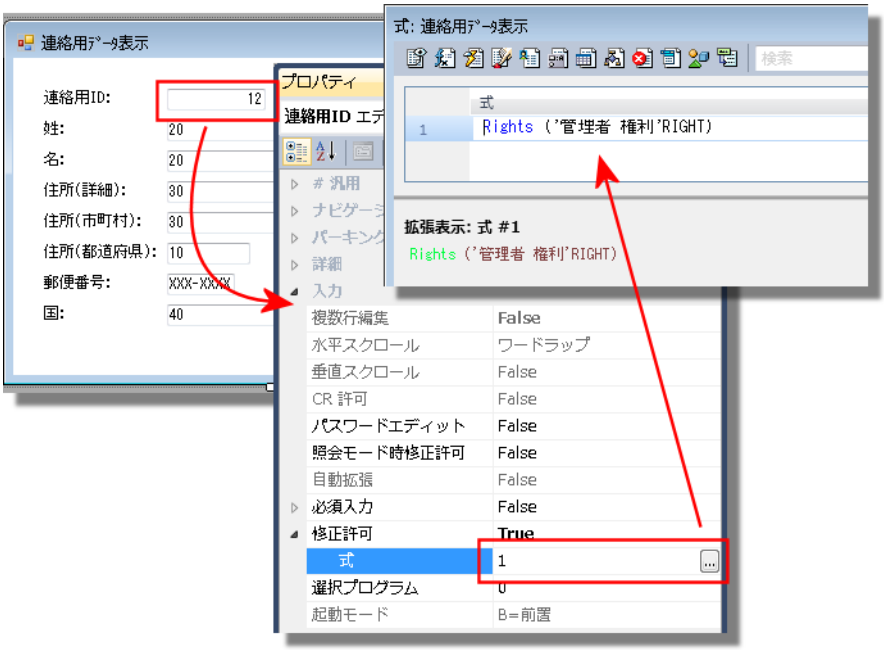
1. **タスク特性**（*Ctrl+P*）を開きます。
2. **オプション**タブを**クリック**します。
3. **修正**特性で、*No* を入力します。
4. ここで**ズーム**（*F5*、または**ダブルクリック**）して式を定義することもできます。実行時に定義された式が False と評価された場合、ユーザはレコードを修正することができなくなります。

注： これによって、ユーザが照会モードから修正モードに切り換えてレコードを修正することを防止します。しかし、これはタスクがアクセスするすべてのレコードが対象となります。より詳細な条件で制御したい場合（例えば、レコード単位で）、あるいは異なる方法を使用したい場合は、別の方法（例えば、レコードがオープンされる前に有効なモードを設定したり、フィールドレベルでの変更を無効にしたり）で制御する必要があります。

エンドユーザが特定の項目の内容を変更することを防止するには

タスクが修正モードの場合、デフォルトでは、ユーザはフォーム上のどのデータ項目も修正することができます。しかし、項目毎にデータを更新することを防止することができます。

項目を修正不可に設定する



1. 設定を変更したいコントロールにカーソルを置きます。*Ctrl+ クリック*を使用することで複数のコントロールを同時に選択することができます。
2. *Alt+Enter* を押下して**コントロール特性**を開きます（すでに開いている場合は、**特性**シートを**クリック**します）。

   実行中に修正されることがない場合は、**修正許可**特性にカーソルを置き *True* を設定します。左側の矢印 ▷ を

   クリックすると式特性が表示されます。式特性で**ズーム**し（または … ボタンを**クリック**して）**式エディタ**に論理値が返る式を定義することで、動的に切り替えることができます。この例では、**Right()** 関数を利用して管理者権利がある場合のみ修正できるようにしています。

これで、ユーザはこの項目にパークはできますが、定義式に設定された条件に合わない場合、修正できなくなります。

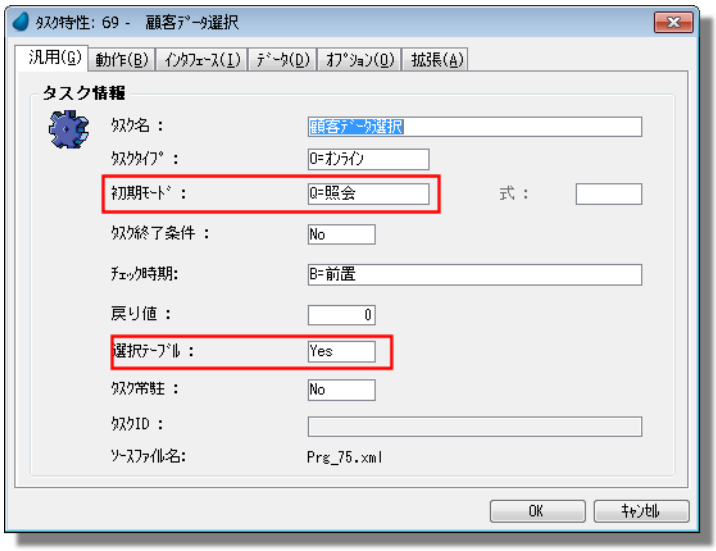
参照： 第 12 章：「マウスを使用した場合のみコントロールへのパークを可能にするには」（265 ページ）

選択プログラムを作成するには

よく作成するプログラムの１つにデータの選択用のプログラムがあります。このタイプのプログラムでは、項目一覧がユーザに対して表示されます。ユーザは、選択したい項目を見つけるために上下にスクロールしたり、キー操作を行うことができます。*Enter* を押下することで項目が選択され、現在パークしている項目の内容が起動元の項目に設定されます。

これらのプログラムは、簡単に作成できます。

選択用プログラムを作成する



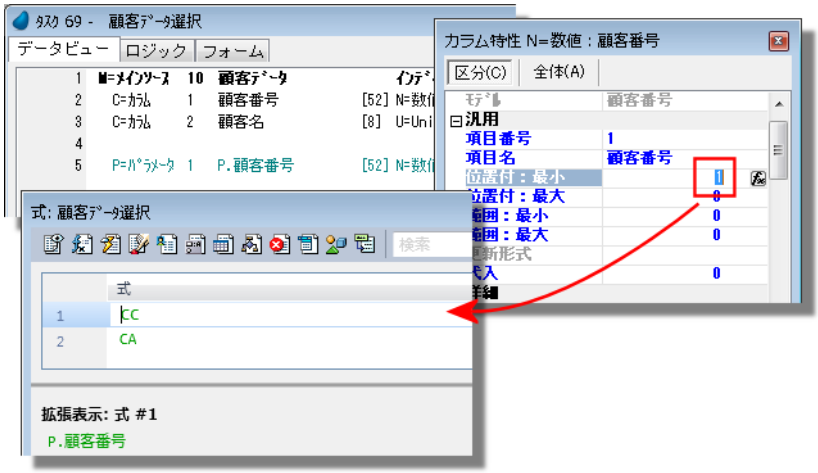
1. プログラムリポジトリで、*F4* を押下し一行追加します。
2. プログラムに名前を入力します。この名前はアプリケーション実行時、Magic xpa は使用しないため任意の名前を使用することができます。
3. **名前**カラムで**ズーム**（*F5*）します。新規プログラムとなるため、**タスク特性**ダイアログが表示されます。ここで以下のパラメータを設定します。

   初期モード：***照会***

   選択テーブル：***Yes***

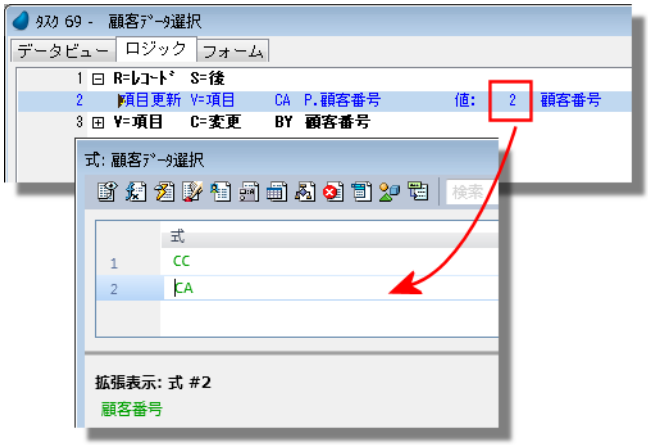
   選択処理ではデータの修正を行わないため、照会モードに設定してください。

   選択テーブル特性は、エンジンの動作を変更します。*Yes* に設定された場合、*Enter* が押下されると**レコード後**を実行しタスクが終了します。



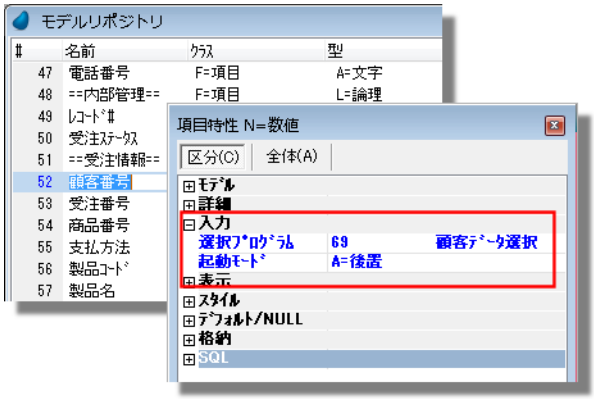
4. **データビューエディタ**では、一覧表示させたいデータソースを指定します。**データソース**特性では、アクセス中のレコードがロックされないように、**アクセス**特性を ***R= 読込***に設定します。
5. また、パラメータ項目を１つ定義します。このパラメータは、起動元に返す値としての**項目モデル**を使用します。この例では、顧客番号を使用しており、顧客番号モデル（**#52**）が使用されていますパラメータの名前が、選択する値の名前によく似ていること注意してください。間違った項目を更新しないように項目の名前の付け方を工夫してください。この場合、パラメータであることを表すため、接頭辞「P.」が使用されています。

6. データソースの**位置付：最小**特性にパラメータ項目を設定します。この例では、データソースの顧客番号カラムを位置付けるためにパラメータの P. 顧客番号を設定しています。



7. **ロジックエディタ**内では、**レコード後**ロジックユニットを作成します。この**ロジックユニット**では、データソースの値でパラメータを更新します。これは選択された値を起動元に返すための処理です。
8. 最後に、フォームを作成します。これは簡単な**テーブル**コントロールを使用して作成します。*Ctrl+G* を使用してフォームを作成することができます（「デフォルトのフォームレイアウトを自動的に作成するには」（132 ページ）を参照してください）。

選択プログラムを使用する



選択プログラムを最も簡単に使用する方法は、モデルやコントロールの**選択プログラム**特性にプログラムを設定することです。この場合、パラメータを明示的に指定する必要はありません。この特性が使用された場合は常に、顧客 ID は暗黙のうちに渡されるようになります。

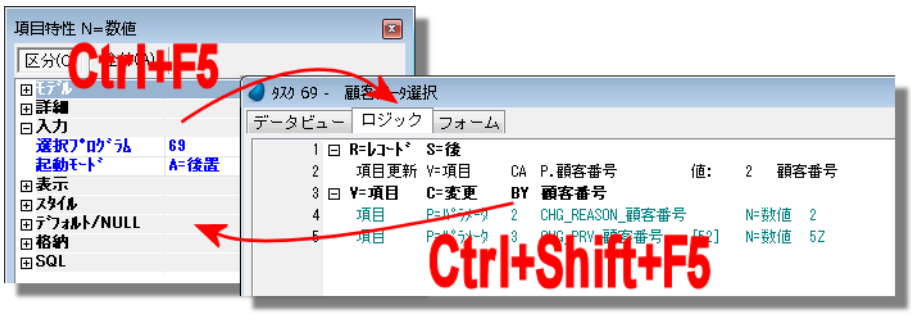
注： データ数が少ない場合は、データソースを割り当てたコンボボックスを使用することもできます。

注： [選択プログラム] 特性を使用して起動されたプログラム／タスクはモーダルウィンドウで表示されます。このため、MDI の外に表示される場合もある為、必要に応じて選択プログラム側のフォームの［開始時の位置］特性で調整してください。

指定したプログラムに移動する

選択プログラム特性に設定したプログラムを開くには、通常は、**モデル**リポジトリを抜け、**プログラム**リポジトリを開いて該当するプログラムを開く操作が必要です。しかし、**選択プログラム**特性にカーソルが位置付けられている状態で *Ctrl+F5*（**オプション→オブジェクトに移動**）を押下すると簡単に該当するプログラムを開くことができます。

この方法でプログラムを開いた場合、*Ctrl+Shift+F5*（オプション→オブジェクトから戻る）を押下すると元の**選択プログラム**特性に戻ります。



注： *Ctrl+F5* でプログラムを開いた場合、起動元の場所がスタックされます。このため、プログラムを開いた後、別のオブジェクトを開いた状態で *Ctrl+Shift+F5* を押下すると起動元に戻ります。

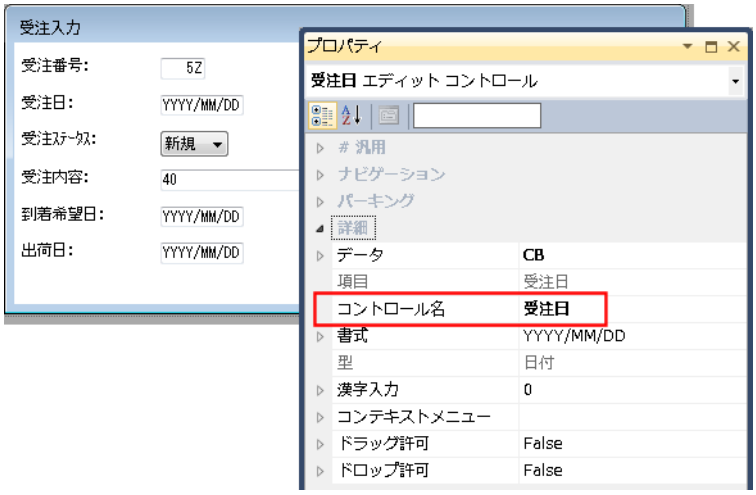
エンドユーザが入力したデータを検証するには

タスク

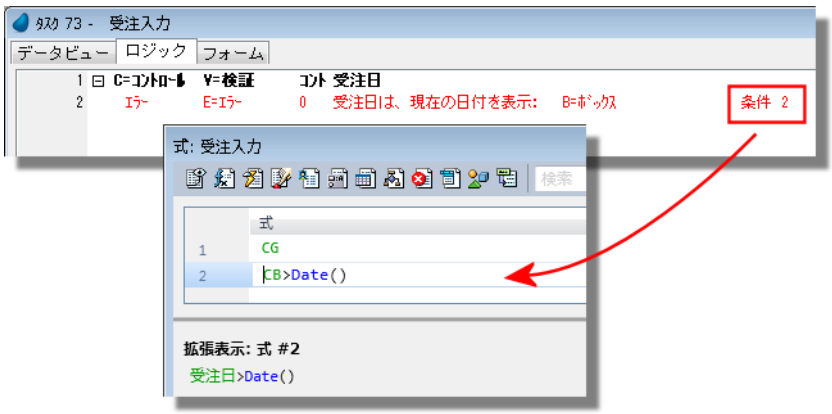
項目モデルを設定した時点で、Magic xpa によるデータの検証処理の大部分が行われることになります。例えば、入力範囲を、1 から 100 の間の数値に制限したり、「A」、「B」、「C」または「X」のみを入力可能にしたい場合は、モデル特性で設定でき、これ以外の検証処理は必要ありません。

しかし、さらに複雑な検証条件が必要な場合、コントロール検証ロジックユニットを作成する必要があります。

コントロール検証ロジックユニットを定義する



1. 最初に、データ内容を検証する対象となるコントロール名特性を確認します。この例では、*受注日*になります。
2. ロジックエディタで、コントロール検証ロジックユニットを作成します。検証対象となるコントロールを選択します。コントロール名特性からズームして一覧から選択することができます。



3. コントロール検証ロジックユニットでは、エラー処理コマンドを定義します。以下の手順で行います。
   - *F4* を押下して 1 行追加します。
   - *E* を入力するか、次のプルダウンリストから*エラー*を選択します。*Tab* を押下します。
   - ズームして表示するエラーメッセージをテキストやデータ項目で定義します。
   - 式を使用しない場合は、*Tab* を押下し、直接エラーメッセージを入力します。
   - ステータス行にエラーメッセージを表示しない場合は、表示カラムで *B= ボックス*（デフォルト）を選択します。
   - 条件カラムでズームして式を指定します。データがエラーの場合は True と評価される式を入力する必要があります。
   - 必要に応じて、エラー処理コマンドの特性を修正して表示内容をカスタマイズすることができます。

受注日が未来の場合、エラーメッセージが表示され、ユーザがエラー要因を修正するか、レコードの修正内容をすべてキャンセル（*Ctrl+F2*、または 編集→キャンセル）しない限り次の項目に移動できなくなります。

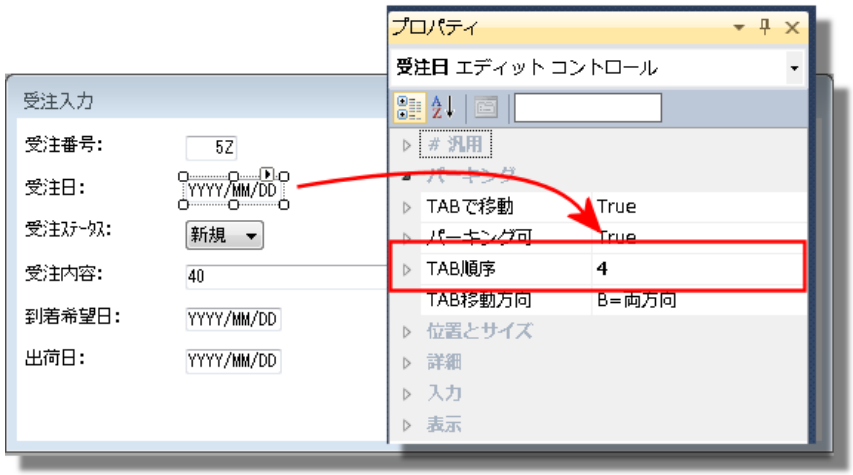
ヒント : 不適切なプログラミングを行うことで、取り消すことができないエラーメッセージが発生するようになる場合があります。例えば、条件カラムが Yes に設定された場合、エラーメッセージを止めることができません。このようなことが発生した場合、タスクを終了することは不可能になります。デバッグ中にこのような状態が発生したら、開発側のウィンドウに切り替えてデバッグ→停止を選択するか、ツールボックスの赤いボタンをクリックしてください。

Tab によるコントロールの移動順序を設定するには

Magic xpa のフォーム上のカーソルは、デフォルトでは上から下に、左から右に移動します。しかし、以下の手順に従うことで、必要に応じてデフォルトの設定を上書きすることができます。

TAB 順序を設定する

1. フォームの **TAB 順序指定**特性を ***False*** に設定します。
2. 各コントロールの **TAB 順序**特性に数値または数値が返る式を指定します。**TAB 順序**特性が灰色で表示されている場合は、修正できません。ステップ **#1** に戻って解除処理を行います。



注： 式が定義されたコントロールは、***Tab*** を割り付けることができないため灰色で表示されます。

3. レイアウトツールバーの **TAB オーダー**のアイコンをクリックすることで ***Tab*** の順番が表示されます。

コントロールの **TAB 順序**を変更した場合、フォームデザイナは連続性を維持した状態で保存します。例えば、**TAB 順序**を ***1*** から ***2*** に変更した場合、フォームデザイナは今まで ***2*** になっていたコントロールの値を ***1*** に変更します。

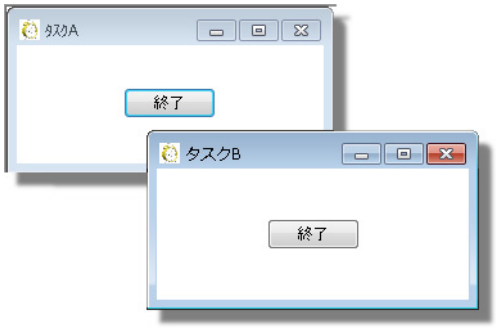
参照： 第 10 章：「指定コントロールにカーソルを移動させるには」（232 ページ）
第 10 章：「カーソルを一定の方向に移動させた場合のみ処理を実行させるには」（234 ページ）

サブタスクレベルからプログラムを終了するには

一緒に実行しているタスクに関連するスタックを持つことがよくあります。これらのタスクは、時々 1 つのウィンドウのように表示させることがあり、その際、終了ボタンをクリックするとタスクのスタック全体を終了させるように動作するのが一般的です。

このようなプログラムにする最もよい方法は、以下のようにイベントロジックユニットを使用することです。

終了イベントを伝播させる

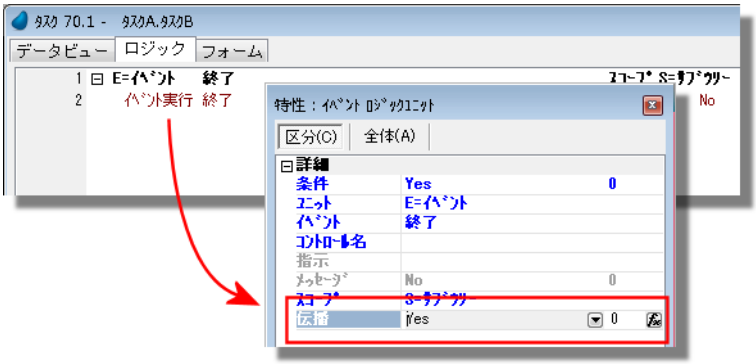


ここに、タスク A とタスク B があります。タスク A の終了ボタンをクリックすると通常は両方のタスクが閉じます。しかし、タスク B を閉じた場合、フォーカスがタスク A に移るだけです。

タスク B を閉じるとタスク A も閉じるようにするには以下のようにします。

1. 最初に、終了イベントを発行する終了ボタンを作成します。

   ユーザが  をクリックしたり、*Esc* を押下した場合と同じイベントのため、この方法ですべての状況がカバーされます。
2. 次に、終了イベントによって実行されるロジックユニットを作成します。このロジックユニットは終了イベントで実行されますが、タスク B に伝播されるようにも設定します。
3. ロジックユニット内では再度終了イベントが発行され、これによってタスク A は終了します。



２つのタスクは明確に分かれて表示されていますが、通常タスク B はタスク A のサブフォームとして使用されます。

参照： 第 8 章：「サブフォーム上のコントロールから親のコントロールに自動的に戻るには」（191 ページ）

タスクの編集中に編集内容を保存するには

PC が動作中に変更内容を保存する方が望ましいやり方です。これは複雑なプログラムを編集する時こそ必要です。間違いを犯したりシステムがクラッシュした場合、変更内容が保存されていれば、最後に保存した内容から作業が継続することができます。

プログラムの編集中に保存する

1. *Ctrl+S*（オプション→プログラム保存）を押下することで編集中のプログラムの内容を保存することができます。

これによってどのような変更内容もディスクに保存されます。

タスクにロジックを作成するには

タスクのロジックは**ロジックエディタ**で定義します。**ロジック**タブを**クリック**することで**ロジックエディタ**は開きます。

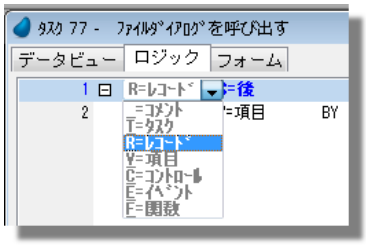
タスクのロジックは***ロジックユニット***と呼ばれるコードのブロック内に保持されます。**ロジックユニット**は、ハンドラを表しているヘッダ行と 0 個以上の*詳細行*（処理コマンド）から構成されています。

ロジックユニットはこれ自身は無手順です。すなわち、**ロジックエディタ**内の位置に依存せず何らかのイベントに対応して実行されます。

ロジックユニット内の***処理コマンド***は手続型で、上から下の順番で実行されます。これらの処理コマンドは、Magic プログラム内で実行される「コード」になります。これらは非常に強力で、他の言語で作成する行数よりはるかに少ない行数で開発することができます。

ヘッダ行を入力する

1. **ロジックエディタ**内のヘッダ行を作成したい行に移動します。
2. ***Ctrl+H*** を押下してヘッダ行を作成します。
3. **ヘッダタイプ**を選択します：***タスク、レコード、項目、コントロール、イベント、***または***関数***があります。
4. **ヘッダタイプ**に依存する**ヘッダ特性**を入力します。



処理コマンドを入力する

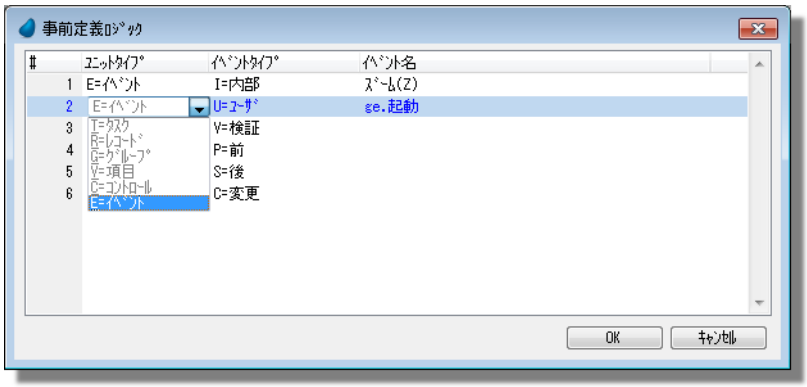
1. **ロジックユニット**内で ***F4*** を押下し、処理コマンドを定義することで必要なロジックを作成します。各処理コマンドの設定内容にはいくつかの違いはありますが、コンテキストヘルプを参照して作業することができます。処理コマンドについては、「タスクに処理コマンドを設定するには」（104 ページ）を参照してください。

ロジックヘッダを簡単に作成するには

事前定義ロジックテーブルにロジックヘッダのパターンを定義しておくことで、簡単にロジックヘッダを作成することができます。

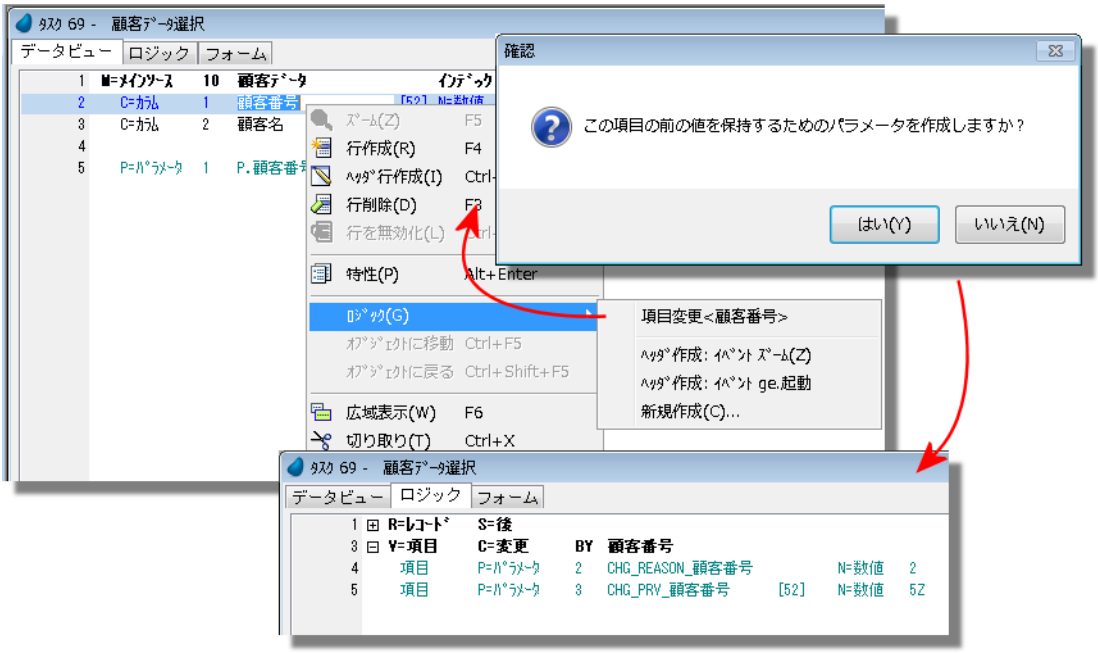
事前定義ロジックを追加する

1. 事前定義ロジックテーブル（オプション→設定→事前定義ロジック）を開きます。
2. よく作成するロジックヘッダのパターン（ユニットタイプ、イベントタイプ、イベント名）を登録します。



データ項目に対応したロジックヘッダを作成する

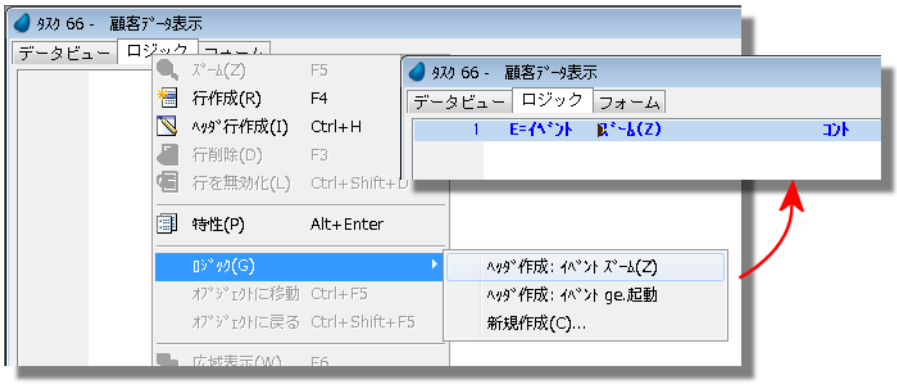
1. データビューエディタの任意のデータ項目にカーソルを移動します。
2. マウスで右クリックを行いコンテキストメニューを開きます。
3. メニューからロジックをクリックします。ここには、事前定義ロジックテーブルに登録されているヘッダの他に、項目変更や新規作成が表示されます。項目変更を選択します。
4. ロジックエディタに表示か切り替わり、項目変更ロジックユニットのヘッダが追加されます。さらに、ヘッダにパラメータを追加するかどうかを確認するダイアログが表示されます。ここで *Yes* をクリックするとパラメータ項目が追加されます。



任意ロジックヘッダを作成する

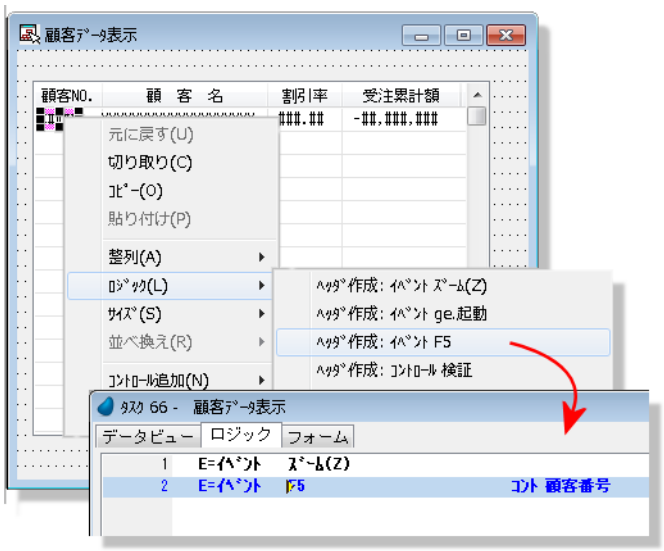
1. ロジックエディタの任意の行にカーソルを移動します。
2. マウスで右クリックを行いコンテキストメニューを開きます。
3. メニューからロジックをクリックします。ここには、事前定義ロジックテーブルに登録されているヘッダの他に、新規作成が表示されます。ヘッダ作成→イベント→ズームを選択します。

4. ロジックエディタの最終行にイベントロジックユニットのヘッダ行が追加されます。



コントロールに対応したロジックヘッダを作成する

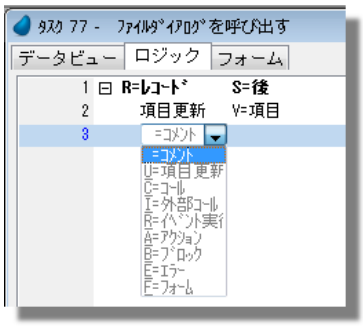
1. フォームエディタの任意のコントロールにカーソルを移動します。
2. マウスで右クリックを行いコンテキストメニューを開きます。
3. メニューからロジックをクリックします。ここには、事前定義ロジックテーブルに登録されているヘッダの他に、項目変更や新規作成が表示されます。ヘッダ作成→イベント→ズーム を選択します。
4. ロジックエディタに表示か切り替わり、イベントロジックユニットのヘッダ行が追加されます。このロジックユニットのコントロール名特性には、フォームで指定したコントロールのコントロール名が設定されます。



注： コントロール名が定義されていない場合、ロジックメニューは有効になりません。

タスクに処理コマンドを設定するには

各ロジックユニットには、処理コマンドを入力する行があります。以下のようにして処理コマンドを設定します。



1. 処理コマンドを入力したい行で *F4* を押下します。
2. 処理コマンドの**タイプ**を選択します。ショートカット文字を入力するか、プルダウンリストから選択します。
3. 選択されたタイプにもとづいた特性を設定します。

８つのタイプの処理コマンドを入力することができます。以下で各処理コマンドの概要を説明しています。

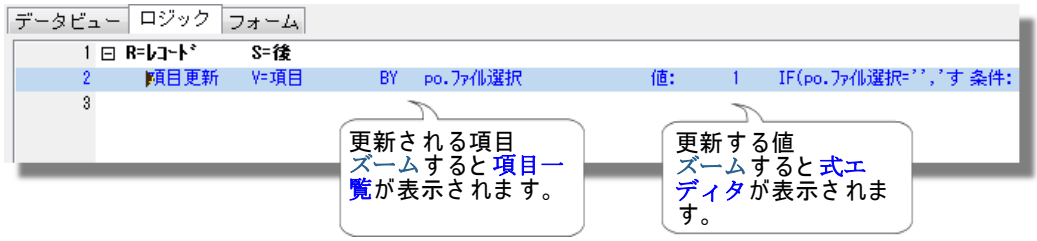
条件特性を使用する

すべての処理コマンドには、**条件**特性があります。基本的には「if」の構文に当たります。**条件**特性が *Yes* の場合、または定義されて式が実行時に True と評価された場合この処理コマンドは実行されます。

- **条件**が *Yes* の場合は、処理コマンドが常に実行されます。
- **条件**が *No* の場合は、処理コマンドは決して実行されません。
- **条件**が ***数値***の場合、式の番号を示しています。条件式を入力するには、ここからズームして式エディタに入り、式を入力します。*Enter* を押下してロジックエディタに戻ります。

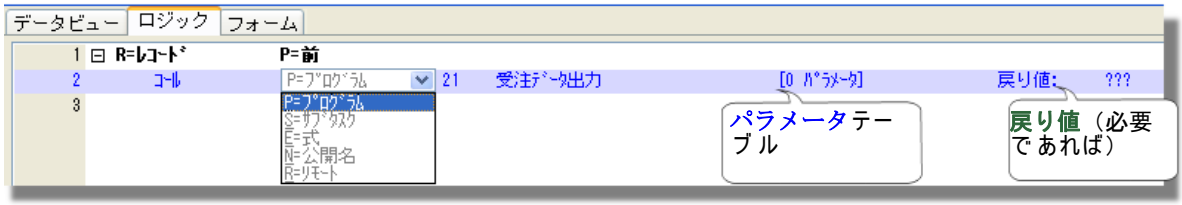
注：　編集メニューの行を無効化（*Ctrl+Shift+D*）を選択すると現在パークしている処理コマンドの実行がスキップされます（条件が *No* の場合と同じ動作になります）。再度、編集メニューの行を有効化（*Ctrl+Shift+D*）を選択することで有効になります。

項目更新



項目更新処理コマンドは、一般的な言語における代入演算子と同じような処理を行います。ここでは式の評価結果の値を変数項目にコピーしています。

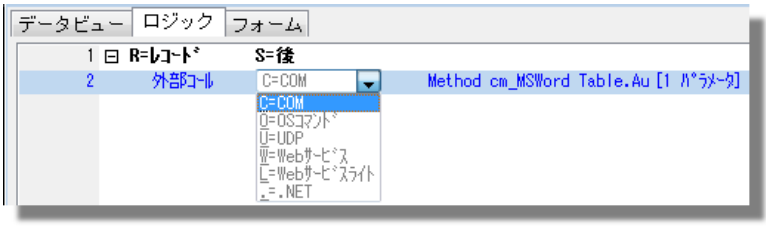
コール



コール処理コマンドは、別の Magic タスクを実行させます。何を起動するかによって設定する特性が異なります。例えば、コール処理プログラムはプログラムリポジトリ内の他のプログラムを番号指定で呼び出します。コールサブタスクは、現在のプログラムのサブタスクのみ呼び出すことができます。コールプログラム名は、公開名を指定してプログラムを呼び出します。

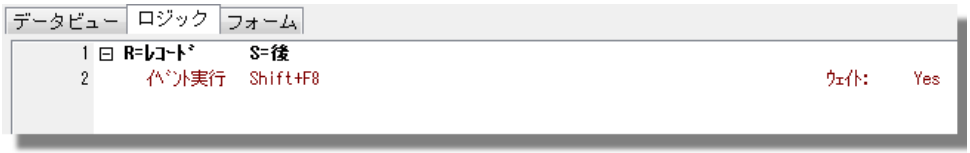
呼び出されるプログラムにパラメータが定義されている場合、パラメータテーブルが使用されます。同様に、**戻り値**特性は、起動したプログラムからの戻り値を受け取る項目を指定することができます。

外部コール



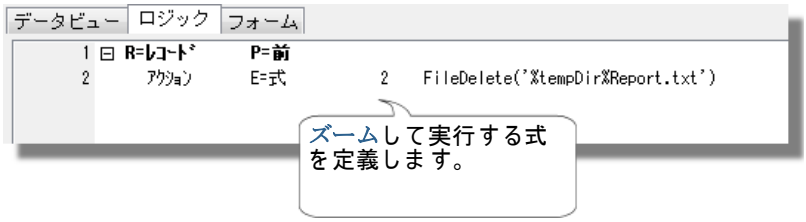
外部コール処理コマンドは、Magic xpa 以外のプログラムを呼び出す場合に使用します。OS のスクリプトや Web サービス、ActiveX オブジェクトなどを呼び出すことができます。これらは各々処理内容に設定内容も異なっています。

イベント実行



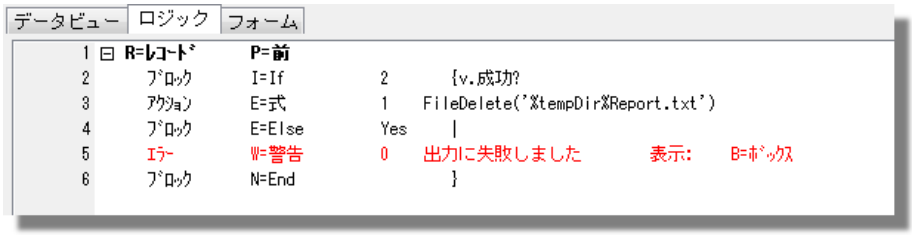
イベント実行処理コマンドは、イベントを発行する処理を行います。ユーザがキー（システムイベント）を押下したり、プッシュボタンによって発行された場合と同じようにイベントが発行されます。

アクション



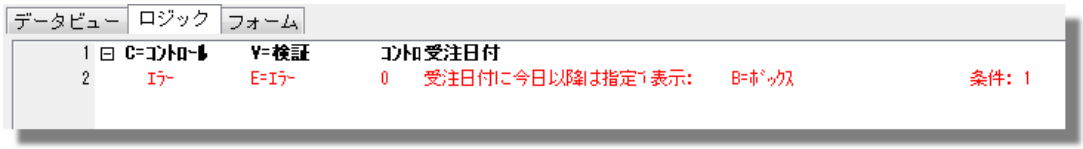
アクション処理コマンドは、データ項目を指定しないで式を実行するために使用されます。この例では、**FileDelete()** 関数を実行しています。

ブロック



ブロック処理コマンドは一般的な言語で行われるようなグループ処理で使用されます。この例では、処理の戻り値にもとづいた if/then/else が定義されています。

エラー



エラー処理コマンドには３種類のオプションあります。

- ***警告***は、ユーザにメッセージを表示するだけの機能があります。

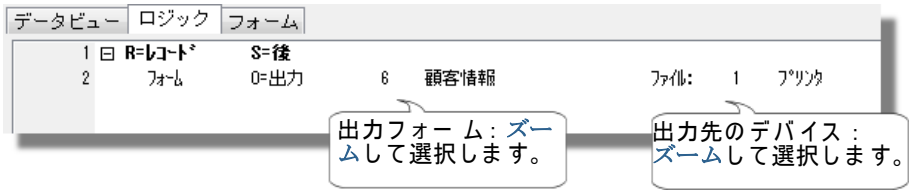
- ***エラー***は、ユーザにメッセージを表示し、エラーが解決するまでこれ以降の処理を継続させないようにします。
- ***復帰***は、コントロールロジックユニットとイベントロジックユニットでのみ有効です。エラーが発生した場合、ロジックユニット内を逆のフローで制御が移動します。

エラーはユーザが不正なデータを保存することを防止するため、処理を完全に停止させることができます。このことについての詳細は、「エンドユーザが入力したデータを検証するには」（97 ページ）を参照してください。

ヒント :*上図のようにエラーメッセージを直接入力した場合、メッセージ欄にマウスカーソルを置くとツールチップでメッセージのすべての内容が表示されます。*

Magic xpa では、エラーメッセージダイアログの表示内容をより詳細に定義することができます。詳細は、『リファレンスヘルプ』を参照してください。

フォーム



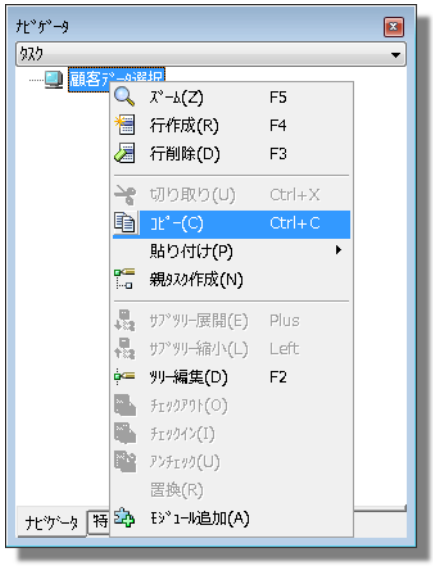
フォーム処理コマンドは、フォームの入出力を行う場合に使用されます。一般に、フォームを出力する場合、プリンタに帳票形式で、他のプログラムに渡すためにテキスト形式で、インターネット用画面として HTML 形式で出力されます。フォーム入力は、通常テキスト形式で使用されます。

サブタスクとしてタスクをコピーするには

他のプログラムによって呼び出されるだけのプログラムを作成する場合があります。このような場合、そのプログラムをサブタスクとすることで、プログラムの処理の流れを理解することが容易になることがあります。このような内容について説明します。

プログラムをサブタスクとしてコピーする

1. ナビゲータペインでコピーするタスクを選択します。
2. 右クリックでコンテキストメニューを表示し、コピーを選択します。
3. このタスクを貼り付けたいプログラムに移動します。
4. ナビゲータペインで、貼り付け先のタスクの親またはそのサブタスクを選択します。
5. 右クリックでコンテキストメニューを表示します。子タスクとして貼り付ける場合は貼り付け（*Ctrl+V*）、兄弟関係のタスクとして貼り付ける場合は、別のツリーに貼付を選択します。



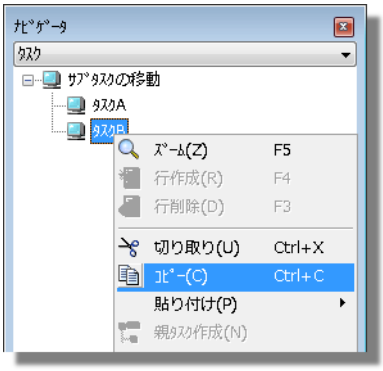
このような操作を実行すると、プログラムはサブタスクとしてコピーされます。この後、パラメータの設定内容やフォームを変更する必要が出てきます。

サブタスクをルートタスクにするには

独立したプログラムにしたいサブタスクが存在する場合があります。また、削除したいダミーの開始プログラムを持っているサブタスクがあるかもしれません。どちらの場合も、サブタスクを一番上のルートタスクとして移動する必要があります。ここではこの方法について説明しています。

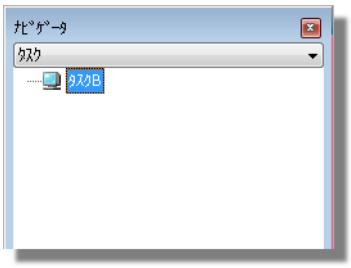
サブタスクを先頭レベルに移動する

1. ナビゲータペインで、移動したいタスクにカーソルを置きます。
2. 右クリックでコンテキストメニューを開き、コピーを選択します。



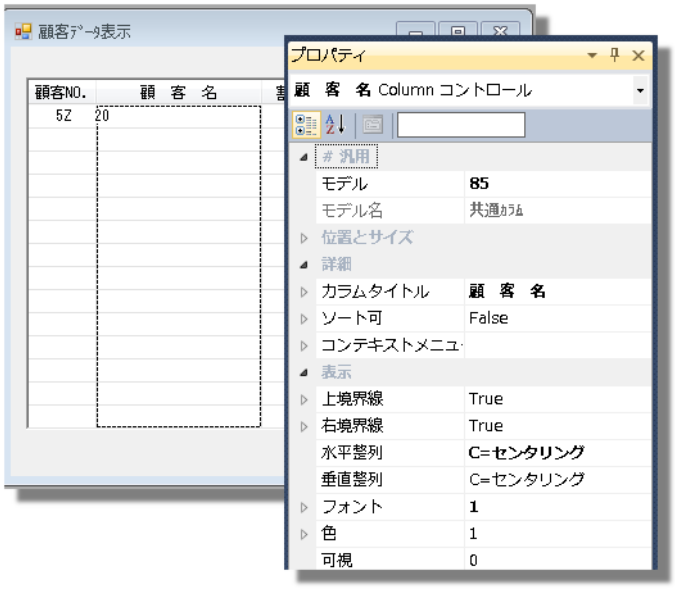
3. 先頭レベルのタスクにカーソルを置きます。
4. 右クリックでコンテキストメニューを開き、貼り付け→サブツリー上書を選択します。
5. 上書き確認ダイアログではいをクリックします。

6. これで、下位レベルのタスクが先頭レベルのタスクに上書きされました。



タスク

テーブルコントロールのカラムを選択するには



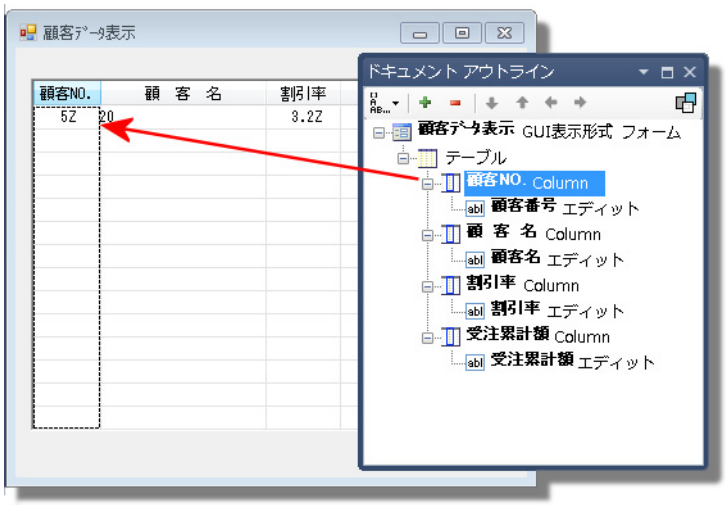
通常、テーブルコントロールをクリックと、テーブル全体が選択されます。これは、テーブルがグループ化されているためです。

- 表示フォームの場合は、カラムヘッダをクリックします。
- 出力フォームの場合は、カラム領域で *Alt+* クリック を実行します。

カラムが選択されると、*Tab* によって次のカラムに移動することができます。

ドキュメントアウトラインで選択する

ドキュメントアウトラインペインで顧客 No Column を選択することで、フォーム上のカラムコントロールが選択状態になります。この操作でテーブルコントロールや特定のエディットコントロールを選択することもできます。



ヒント : *以下の操作で複数のカラムを選択することもできます。同時に複数の ［カラム特性］ を変更したい場合に便利です。*

- 表示フォームの場合 …… *Ctrl* + クリック
- 出力フォームの場合 …… *Alt* + *Shift* + クリック

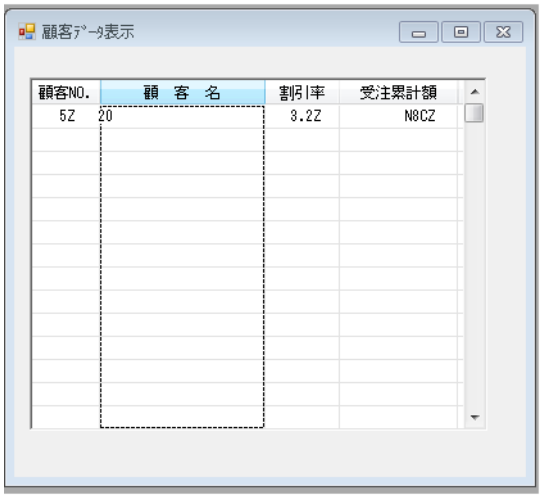
強調表示カラムの左側にコントロールや項目をドロップしてカラムを作成するには

通常、コントロールや項目をテーブルにドロップする場合、ドロップしたカラムの右側にカラムが追加されます。コントロールまたは項目を左側に追加させたい場合は、*Shift* を押下しながらドロップします。

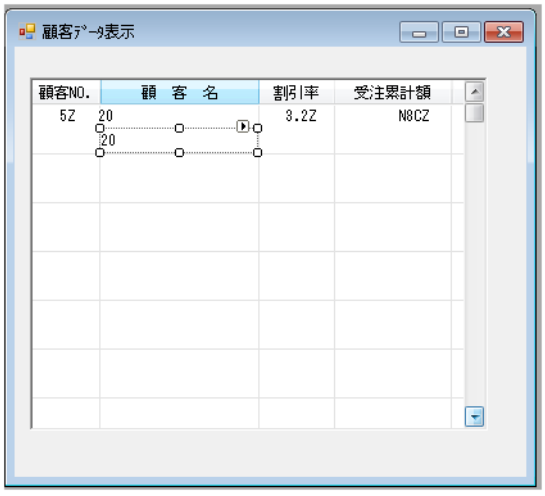
複数のコントロールを同じカラムに配置するには

通常、コントロールをテーブルコントロールに配置すると、ドロップしたコントロールの右側にカラムが追加されます。しかし、*Alt* を押下しながらドロップすることで、選択したカラム内にコントロールが追加されます。

Alt を押下しながら項目をドロップした場合の結果

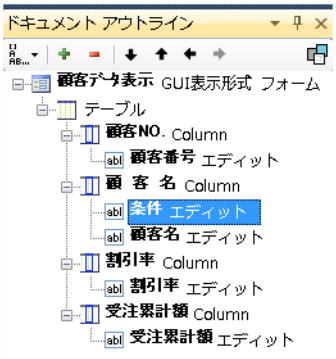


項目をドロップする前



ドロップした後

ドキュメントアウトラインペインでは、顧客名カラムに２つのカラムがリンクされていることが確認できます。

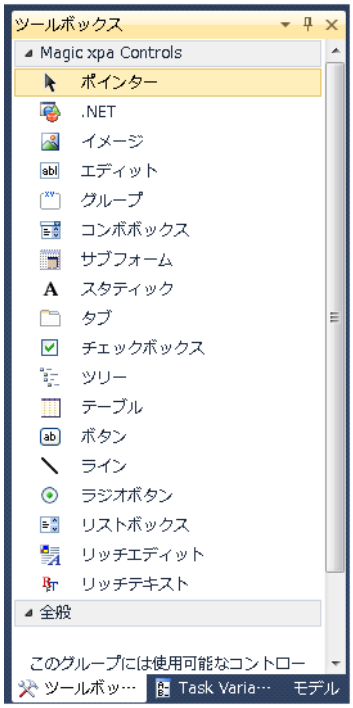


フォームにコントロールを追加するには

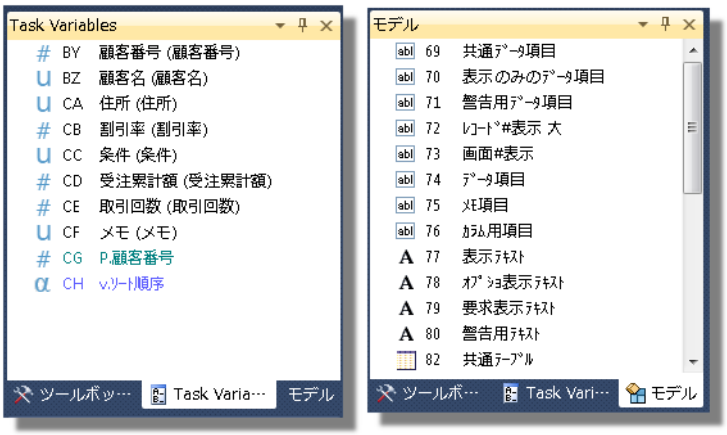
タスク

フォームにコントロールを追加するには、［ツールボックス］ペインからコントロールをドラッグ＆ドロップするだけです。

注： フォームの形式によって、扱うことのできるコントロールに制限があります。



タスク項目ペインまたは、**モデル**ペインから項目をドラッグすることもできます。これは、データ項目、または、モデルにリンクされたコントロールを定義することができます。



コントロールを連続して複数回ドロップするには

通常、コントロールをドロップすると、カーソルは通常の形に戻ります。この後、別のコントロールを選択する場合は、いったんメニューや**ツールボックス**ペインに戻る必要があります。

しかし、同じコントロールを複数回ドロップしたい場合は、***Ctrl*** を押下した状態でコントロールをドロップします。

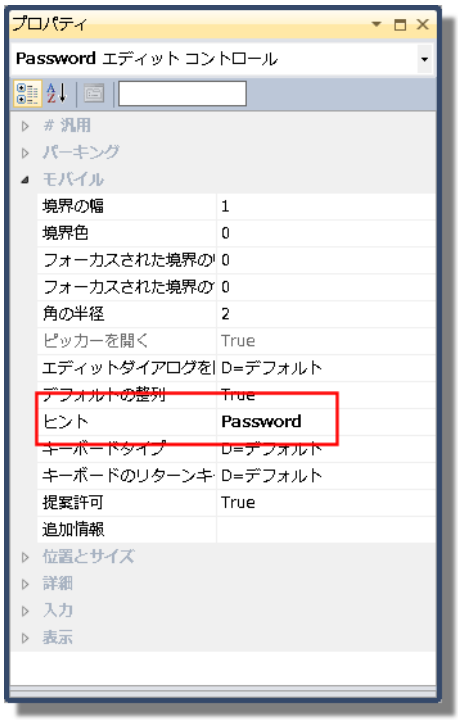
ヒント : *コントロールのドロップ処理を中断する場合は、**Esc** を押下します。*

エディットコントロールにヒントを定義するには

タスク

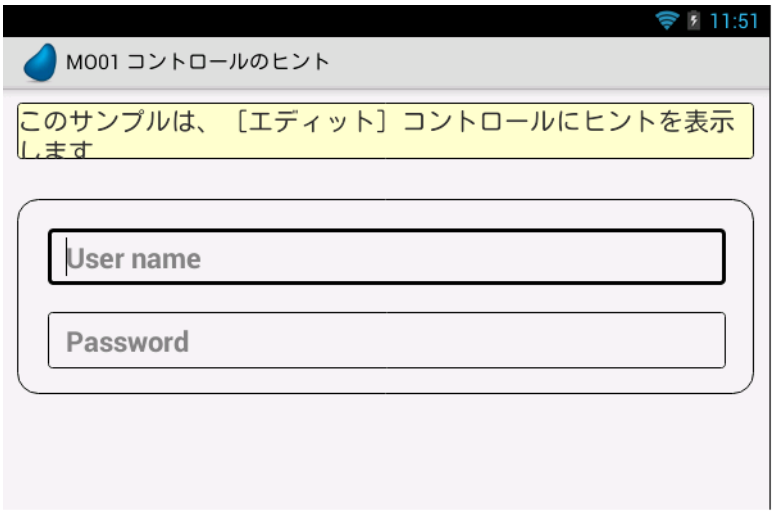
エディットコントロールにヒントを定義することができます。ヒントは、入力している間に自動的に削除されるエディットコントロール上で表示されルテキストです。

エディットコントロールのコントロール特性のヒント特性で以下のように設定します。



ユーザが最初に見た時にエディットコントロールに表示されテキストを入力するだけです。

モバイルデバイスの場合は、実行すると以下のように表示されます。



注： *Shift* キーを押下した状態でデータ項目をフォームにドラッグすると、モバイルデザインモードでは以下のように動作します。

- データ項目の名前がヒント特性の値として追加されます。
- 表示名特性に値が設定されている場合、この値がヒントとして表示されます。

サポートバージョン：2.5 モバイル

3.2 Windows

再利用のために帳票フォームの書式を保存するには

帳票フォーム（クラス >0）は Magic xpa では簡単に作成することができます。しかし、フォームを規格化したり、再利用するために複雑なフォームを保存したい場合は、フォームテンプレートを使用することができます。

フォームをテンプレートとして保存する

1. テンプレートとして保存したいフォームを開きます。
2. **オプション→テンプレート作成**を選択します。
3. ファイルダイアログが表示されたら、ファイル名を入力します。デフォルトの拡張子は ***.mft*** です。
4. **保存**を**クリック**します。

テンプレートフォームを使用する

1. テンプレートを読み込みたいフォームを開きます。
2. **オプション→テンプレート読込**を選択します。
3. **ファイル**ダイアログが表示されたら、読み込むテンプレートファイルを選択します。
4. **開く**を**クリック**します。

テンプレートがフォームに読み込まれます。フォーム上のすべてのコントロールは、オリジナルのフォームとなります。この後、これらのコントロールにタスクの項目を割り当てる作業が必要になります。

注： フォームの規格化を行う別の方法として、フォームとコントロールのモデルを使用することがあります。これらは、見た目が同じようなフォームを定義することを可能にするだけでなく、その規格自体が変更された場合にモデルを変更することで、モデルを使用しているすべてのフォームやコントロールが自動的に変更された規格に合うようになります。

注： Ver3 では、表示フォームとテキスト形式フォームでテンプレートを作成することができなくなりました。

タスク

複数のコントロールを同時に選択するには

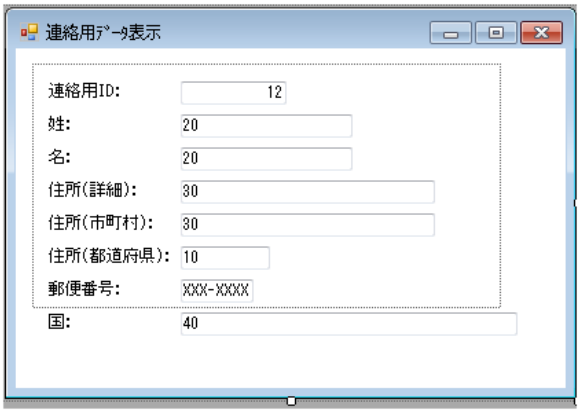
コントロールをクリックすることでそのコントロールを選択することができます。また、同時に複数のコントロールを選択する場合も考えられます。これは、グループでコントロールを移動する場合に便利です。また、コントロールの共通の特性を変更する場合にも役立ちます。例えば、同じようなコントロールのグループを選択し、設定されている同じモデルをすべてに変更したり、すべてのフォントを変更するようなことができます。

コントロールをまとめて選択する方法には、*ラバーバンド*（選択範囲を示す枠）と、*Ctrl+* クリックの２つの方法があります。

ラバーバンドを使用してコントロールを選択する

1. ラバーバンドを使用してコントロールを選択するには、選択するすべてのコントロールの外側のフォーム上をクリックします。
2. すべてのコントロールのまわりを矩形のボックス状にドラッグし、マウスを放します。

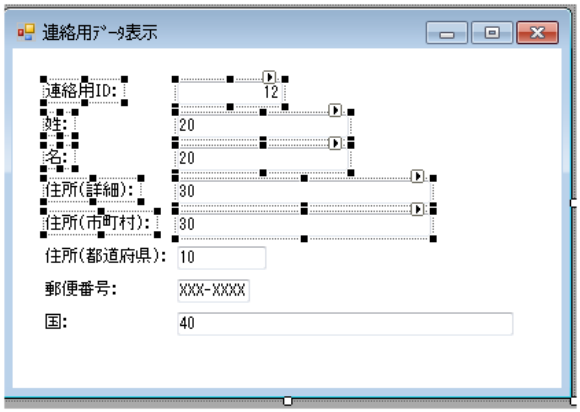
エリア内のすべてのコントロールが選択されます。コントロールがラバーバンドに交差した場合も選択されます。



Ctrl+ クリックを使用してコントロールを選択する

Ctrl+ クリックを使用して 1 つずつコントロールを選択することもできます。

1. *Ctrl* を押下した状態でコントロールをクリックします。
2. *Ctrl* を押下したまま、別のコントロールをクリックします。
3. **#2** の操作をすべてのコントロールに対して繰り返します。
4. 選択を解除したいコントロールがあった場合は、同じ状態でもう一度コントロールをクリックします。

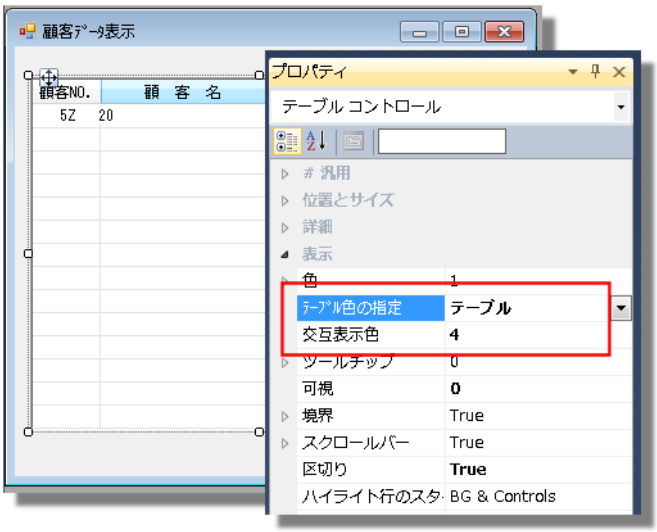


ヒント :*スペースバーを押下することでコントロールを選択解除することができます。*

テーブルコントロールに交互色を表示させるには

サイズの大きなテーブルの場合、行を目で追うのは大変な場合があります。このような場合の 1 つの解決策として、列の色を交互に変更する方法があります。

テーブルで交互色を使用する

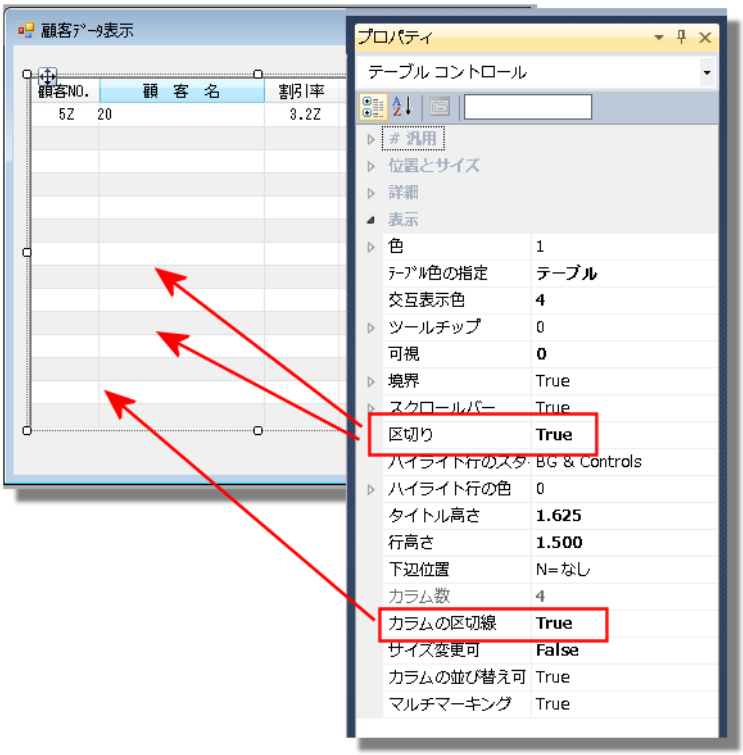


1. クリックを行って、テーブルコントロールを選択します。
2. コントロール特性において以下の操作を行います。
   - **テーブル色の指定**特性を *T= テーブルに依存*に設定します。
   - **交互表示色**特性に背景色となる色番号を選択します。番号を直接入力するか、ズームして基本色一覧から選択します。

テーブルは、交互に色が変わった状態で表示されます。基本色は**色**特性（この例では *1*）で指定された色になり、**交互表示色**特性（この例では *4*）で指定された色と交互に表示されます。

テーブルコントロールの区切り線の表示／非表示を行うには

タスク



テーブルの**コントロール特性**に値を設定することによって、行とカラムの区切り線を表示させるかどうかを制御することができます。

区切り特性を ***True*** に設定すると、水平の行区切り線が表示されます。***False*** に設定すると表示されなくなります。

カラムの区切線特性を ***True*** に設定すると、垂直のカラム区切り線が表示されます。***False*** に設定すると表示されなくなります。

ヒント : *アプリケーションのテーブル表示に区切り線が表示されなくても、テーブルの編集時にテーブル上のコントロールを整列させることでより使いやすいものになる場合があります。*

出力フォームのサイズを変更するには

フォーム特性シートの位置特性から、フォームの論理サイズを定義することができます。別の方法として、*Shift+ 矢印キー*を使用することでフォームのレイアウトを変更することができます。

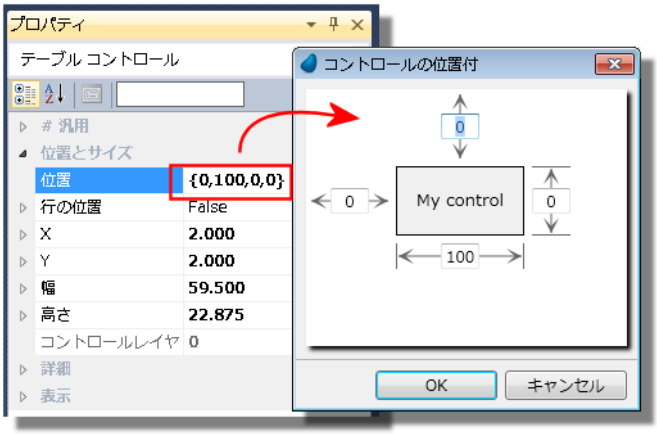
フォームを移動する場合は、*矢印キー*を使用するか、レイアウトのタイトルバーをドラッグしてください。

フォームのレイアウト領域の大きさを変更するには、レイアウト・フレームの右側と一番下の境界をクリックして、ドラッグしてください。

フォームのレイアウト領域の大きさを変更したり移動した後で、*Enter* を押下して変更を確定するか、*Esc* を押下してキャンセルします。

レイアウト・フレームをレイアウト領域より小さくすると、Magic xpa は自動的にフォームにスクロールバーを追加します。コントロールが紛失するような方法で、フォームのレイアウト領域を変更することはできません。

コントロールのサイズを表示テキストに合わせるには



GUI 環境の利点の一つは、ウィンドウがすべてユーザによってサイズ変更できることです。しかしこのことは、すべてのユーザに対して最良のフォームをデザインすることを難しくする要因でもあります。

位置特性は、このような問題の解決に役立ちます。**位置**特性は、フォームのサイズの変更内容に応じて各コントロールのサイズを指定することを可能にするものです。

- **幅**または、**高さ**で ***100*** を入力すると、ウィンドウの指定された方向に対して伸びた分の 100% の割合でサイズが変更されます。
- **X** または、Y で ***100*** を入力すると、ウィンドウの指定された方向に対して伸びた分の 100% の割合で移動します。

例 :

フォームのサイズを変更した場合以下のように動作します。

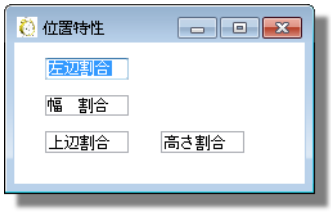
- **X**=***100***/Y=***100*** と定義した場合、コントロールの右下の位置は維持されます。これは、フォームのサイズを変更しても位置を維持させるボタンに利用できます。
- **幅** =***100*** と定義した場合、コントロールは水平方向にサイズが変更されます。これは、コントロールがデータのサイズより小さいような場合に利用できます。
- **X**=***100*** と定義した場合、コントロールは水平方向に移動します。これは、**幅**が ***100*** に設定されたコントロールの後に表示されるコントロールで利用できます。

テーブルコントロールで**位置**特性を使用すると動作の違いがよくわかります。テーブルの高さ割合が 100% の場合、フォームサイズを変更（拡張）すると、行が追加されます。幅割合が 100% の場合、カラムのサイズが変更され、広くなるためより多くのデータが表示されます。

ヒント : *コントロールは元のサイズより小さくはなりません。このため、フォームのサイズが変更されることを考慮して設計する場合、必要最小限のサイズで各コントロールを作成する必要があります。*

以下の図は、位置を 100% に設定した場合の効果を表しています。各**エディット**コントロールは、特定の方向に対して 100% に設定されており、ウィンドウのサイズを拡張すると、どのような結果になるかが確認できるようになっています。

コントロールの位置を 100% にした場合の効果



拡張する前のウィンドウ

右方向に拡張します。**左辺割合**と**幅割合**が両方とも 100% の場合、コントロールの右側がウィンドウの右端に合わせて動きます。

しかし、**左辺割合**が 100% に設定されているため、コントロールの左端も動きます。このためコントロールのサイズ自体は拡張されません。

下辺を拡張します。**上辺割合**と**高さ割合**の両方が 100% の場合、ウィンドウの下辺に合わせてコントロールの下辺が動きます。

しかし、**上辺位置**が 100% に設定されているためコントロールの上辺も動きます。このためコントロールの高さ自体は拡張されません。

タスク

コントロールを他のコントロールより前面に表示させるには

GUI 形式フォームを作成する場合、あるコントロールが別のものと同じ場所に配置されている場合、どのコントロールを全面に表示させるなどの問題が出てきます。これは、*Z オーダ*と呼ばれる機能を使用してプログラムで処理されます。これは、コントロールがフォーム上に表示される順番を参照する値です。下位の Z オーダを持つコントロールは、より高い **Z オーダ**を持つコントロールの背後に表示されることになります。

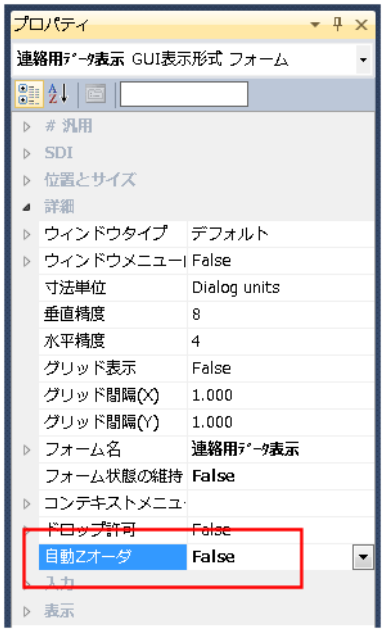
Magic xpa は、デフォルトで自動的にコントロールの Z オーダを設定します。何らかの理由で Z オーダを変える必要がある場合は、手動で設定する必要があります。

Z オーダを表示する

現在の Z オーダの値を確認するには、**表示**メニューか**レイアウトツールバー**から **Z オーダ**を選択します。この例では、複数のコントロールが**グループ**コントロールの背後に配置されています。**グループ**コントロールの **Z オーダ** は *1*、それ以外は、*2 ～ 17* が表示されています。

Z オーダを手動にする

1. Z オーダを確定するために、最初に、**自動 Z オーダ機能**を解除する必要があります。**フォーム**の**自動 Z オーダ**特性を *False* に設定します。。

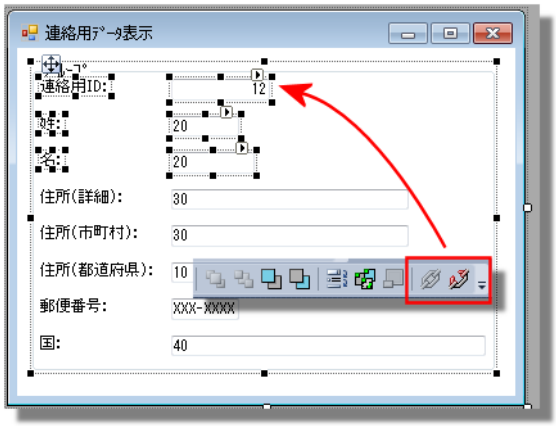


2. これにより、コントロールを選択すると、[**書式→順序**] メニューや**ツールバー**で利用可能ないくつかのオプションが有効になります。コントロールの表示を制御するために、**前面に移動**や**背面に移動**などが利用できます。

ヒント : *表示されていないコントロールを選択するには、[ドキュメントアウトライン] ペインでコントロールを選択してください。*

コントロールをリンクする

あるコントロール（例えば、グループコントロール上のテキスト）の一部として別のコントロールを表示させたい場合、一方が他方のコントロールに（親子）リンクさせることで実現できます。リンクした場合、リンクされたコントロールは常に表示され Z オーダの内容にかかわらず背後には隠れません。コントロールを他のコントロールにリンクさせるには以下のようにします。



1. *Ctrl* を押下した状態でコントロールをクリックし、リンクするコントロールを（複数）選択します。
2. ツールバーの アイコンをクリックします。
3. 背景用のコントロール（この例ではグループコントロール）をクリックします。

これで、コントロールはグループコントロールの子コントロールとしてリンクされます。

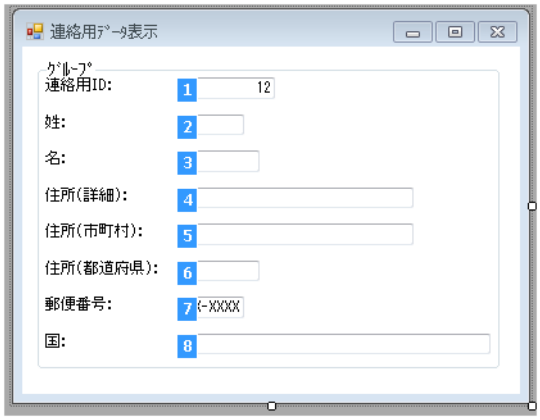
タスクのメインフォームに移動するには

GUI プログラミングでは、実行時に表示されるものだけでもタスク内にたくさんのメインフォームを使用することになります。フォームエディタを開き、該当するメインフォームをスクロールし、*F5* を押下して開くことでメインフォームにアクセスすることができます。

しかし、メインフォームを開く簡単な方法として *Ctrl+M*（オプション→フォーム表示）を押下する方法もあります。これによって、タスクのメインフォームを開き、編集を行うことができます。

ヒント : *メインフォームを閉じ、タスクに戻るには、F8 を押下してください。*

すべてのコントロールの TAB 順序表示するには



表示メニューの**タブオーダー**かツールバーの  アイコンを**クリック**することで、**TAB 順序**を表示することができます。

フォームデザイナの修正内容を取り消すには

Magic xpa を使用することで迅速にフォームの編集作業を行うことができますが、時として安易にミスを犯してしまうこともあります。このような場合に備えて、**フォームデザイナ**には複数レベルで修正内容を取り消す機能があります。

最新の修正内容を取り消すには、***Ctrl+Z***（**編集→変更取消**）を押下します。ツールバー上または**コマンド**パレット上の  アイコンを**クリック**することで取り消すこともできます。

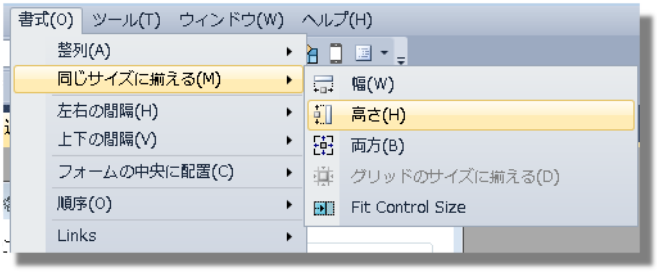
複数のコントロールの幅や高さを同じにするには

コントロールをフォーム上に整列させるため、高さや幅を同じにする必要があります。

例えば、入力項目に対する項目名表示のテキストが項目と異なる高さになっている場合、これらは上辺位置や下辺位置が同じでも関連性があるものとは認識されない可能性があります。

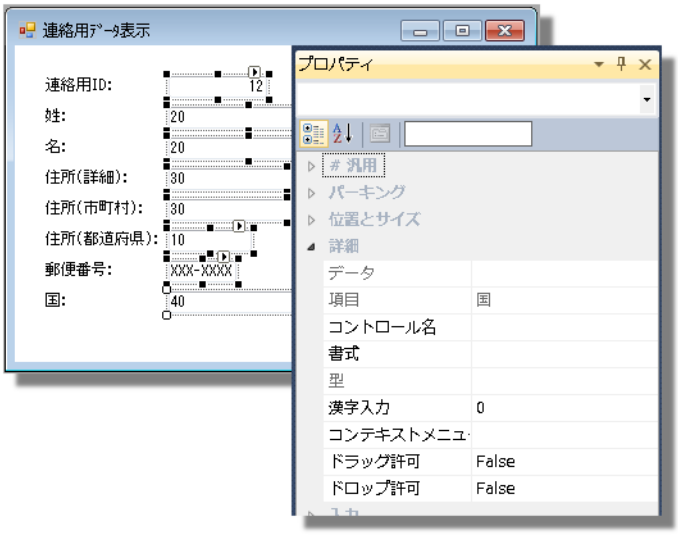
表示フォーム

これらはツールバーや**書式**メニューの**同じサイズに揃える**を選択して行うことができます。



複数のコントロールのプロパティを同時に設定するには

通常、同じフォーム上のコントロールは同じように表示させる必要があります。この場合、コントロールには同じプロパティを設定する必要があります。コントロールをクリックし、プロパティペインでプロパティを変更することで個別にプロパティを設定することができます。しかし、同時に複数のコントロールプロパティを変更することで開発工数を減らすことができます。ここではその方法について説明します。



複数のコントロールの特性を変更する

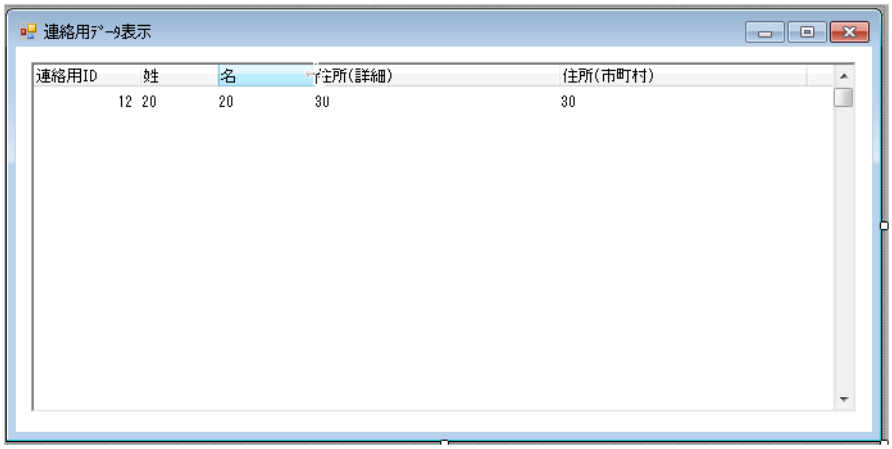
1. 変更したいコントロールを選択します。ラバーバンドや *Ctrl+* クリックを使用することで選択できます。
2. プロパティペインに移動し、プロパティ値を変更します。

複数のコントロールを選択すると、プロパティペインの表示が変わります。ペインのヘッダには、何も表示されず、選択したすべてのコントロールで共通のプロパティのみ設定可能に。異なる種類のコントロールを選択した場合（例：スタティックとエディットなど）、共通するプロパティが少なくなります。

ヒント：*これは特にモデルをコントロールに定義していて、独自のプロパティ値に設定していないような場合に有効です。同じコントロールであれば、同じモデルを定義することで簡単にプロパティ値を反映させることができます。*

テーブルコントロールの幅を変更するには

タスク



通常、テーブルコントロールのカラム上でドラッグするとカラム区切り線が移動し、カラム上のコントロールも移動します。カラムの幅を広げる必要があり他のカラムを右側に移動させたい場合は、以下のようにします。

1. テーブルのヘッダ領域上のカラム区切り線にカーソルを置きます。上図のようにカーソルは両側に矢印が表示される形状になります。
2. カラムを右側にドラッグすると、右側のカラムが移動します。

操作を実行すると、カラム上のコントロールも右側に移動されます。

注： カラムタイトルのテキストがカラムの幅より長い場合、（開発／実行時のどちらも）以下のように動作します。

- タイトルの末尾に省略記号（'...'）が表示されます。
- カラムコントロールの垂直整列特性が*上寄せ*の場合、タイトルがワードラップ（折り返し表示）されます。

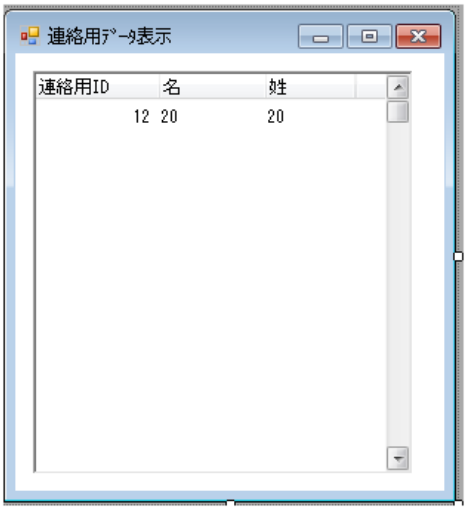
テーブルコントロールのカラムを移動するには

テーブルコントロールのカラムを別の位置に移動する必要がある場合、簡単に行うことができます。

テーブルコントロールのカラムを移動する

1. 移動したいカラムのヘッダ上にカーソルを置きます。
2. マウスボタンを置き、ヘッダを移動したい場所にドラッグします。カラムが表示される場所に若干黒い行が表示されます。
3. マウスボタンを離します。

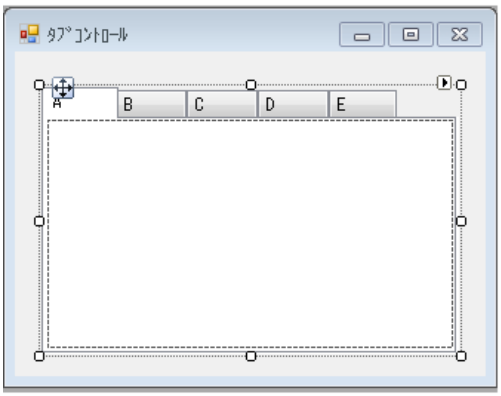
変更前



変更後

タブコントロールの編集中にタブの切り替えを行うには

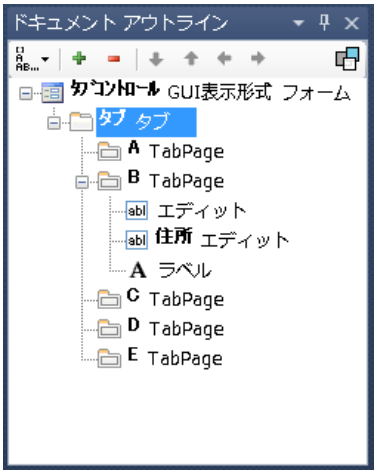
タブコントロールを編集中は、コントロールをクリックするとリンクされたコントロールを含むタブコントロール全体が選択されます。



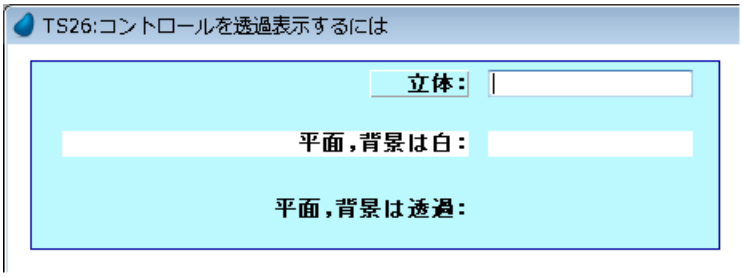
１つのタブを選択し、そこにどのようなコントロールがリンクされているか確認したい場合は以下の操作を行う必要があります。

1. タブコントロール全体が選択された状態で、表示させたいタブコントロールのヘッダをクリックします。この例では、C というヘッダをクリックすると C のタブが開きます。

2. 他の方法として、ドキュメントアウトラインペインで開きたいタブページを選択します。選択するとコントロールの表示も切り替わります。



コントロールを透過表示するには



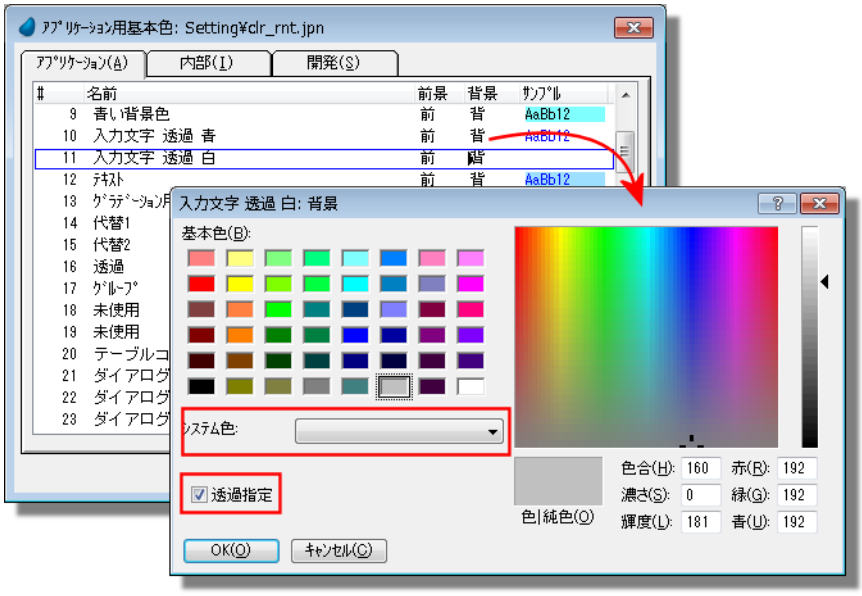
通常、コントロールの背景色は不透明か、Windows から継承された立体色です。例のように背景色を透過にすることで背景の画像を表示させたい場合があります。

二番目のテキストコントロールは、白い背景色を持つ平面のテキストコントロールです。三番目のコントロールは透過の背景色になっていて、前景色が異なっています。

このような表示にする方法は、基本的に、透過の背景色を持つ色をコントロールに定義することです。以下にどのように行うかを説明します。

背景色を設定する

1. オプション→設定→基本色を選択します。
2. 変更したい色をクリックするか、*F4* を押下して色を追加します。
3. 背景色カラムでズームします。



4. システム色を*空白*に設定します。ドロップダウンリストの先頭で空白行を選択することで設定できます。.
5. 透過のチェックボックスをチェックします。
6. OK をクリックします。
7. 基本色テーブルに戻り、（透過設定を除いた）同じ方法で前景色も設定できます。ここでテキストの表示色を選択します（通常は、黒か白を使用します）。
8. OK をクリックして色指定のダイアログボックスを終了します。

これで平面のコントロールでこの色を定義すると、背景色は透過になります。

フォームにデフォルトのプッシュボタンを設定するには

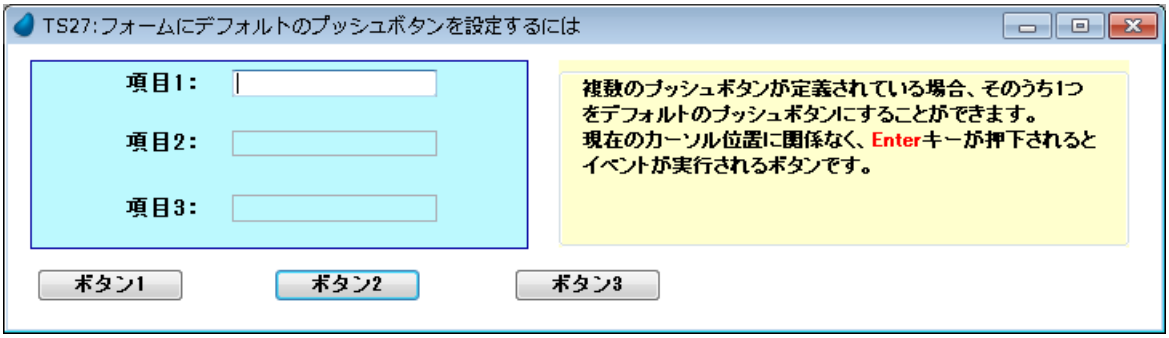
注： Magic ver3 のフォームデザイナでは、ボタンコントロールと表示されるようになりました。

フォーム上に複数のプッシュボタンが定義されている場合、ユーザが *Enter* を押下するときに押下されたように動作するデフォルトのプッシュボタンを割り当てる必要が考えられます。

例えば、このフォーム上には、ボタン２がデフォルトとして定義されています。

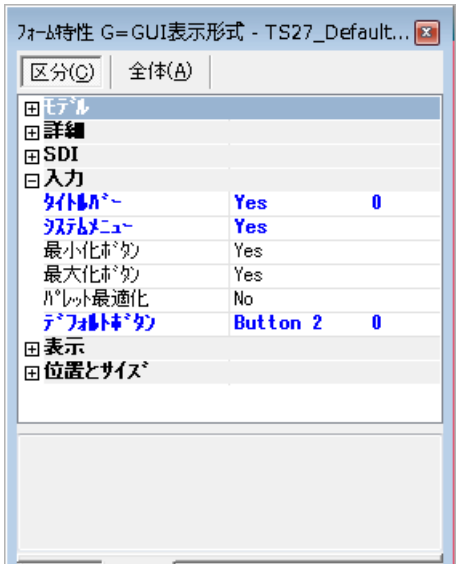
ウィンドウが開いた時点では、カーソルは項目１上にありますが、*Enter* を押下するとメッセージダイアログが表示されます。

ここでは、どのようにして設定するかを説明します。



デフォルトのプッシュボタンを設定する

1. ボタンコントロールを配置し、コントロール名を設定します。
2. フォーム特性（コントロールが選択されていなければ *Alt+Enter*）を開きます。
3. 入力セクションの**デフォルトボタン**特性にカーソルを移動します。
4. ここからズームしてプッシュボタンの**コントロール名**を選択します。
5. 右側の**式**特性でズームして、式エディタに実行時にコントロール名と表示される式を定義することもできます。



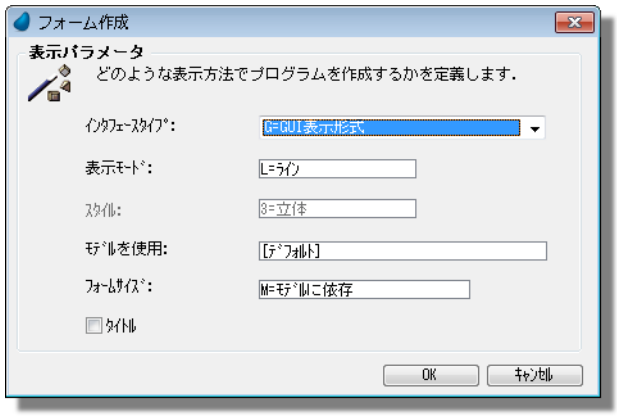
これで、プログラムを実行すると、選択されたボタンはデフォルトボタンになります。

デフォルトのフォームレイアウトを自動的に作成するには

タスクのデータビューをすべてフォームに配置するような場合、簡単な方法があります。これは*フォームジェネレータ*と呼ばれ、データビューとして定義された項目をすべてメインフォームに配置することができます。

フォームジェネレータを使用する

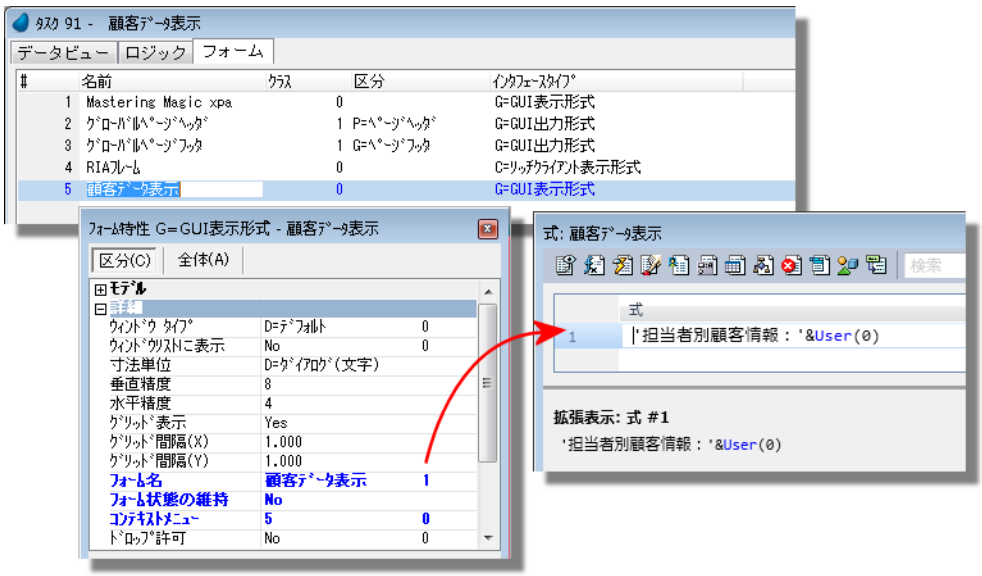
1. フォームを作成するタスクを開きます。
2. フォームタブをクリックします。
3. *Ctrl+G*（オプション→フォーム作成）を押下します。
4. 上書き確認のダイアログボックスが表示されます。はいをクリックします。
5. フォーム作成ダイアログボックスが表示されます。必要に応じて、オプションを設定します。
   - **表示モード：**データをテーブルコントロールで表示する場合はラインを選択します。これ以外はスクリーンを選択します。
   - **スタイル：**立体または平面を選択します。
   - **モデルを使用：**フォームモデルを使用する場合はここでモデルを選択します。
   - **フォームサイズ：**モデルに依存、表示内容に依存、MDI 内に納めるの３つのオプションがあります。
6. OK をクリックします。作成されたフォームにはデータビューがすべて配置されます。



ヒント :*Ctrl+G の押下は、フォームエディタ上で行ってください。データビューエディタやロジックエディタで行った場合、APG が実行されプログラムが上書きされてしまいます。*

ウィンドウタイトルを動的に表示させるには

一般にタイトルバーを持つウィンドウは、タイトル表示用のテキストを指定します。デフォルトでは、フォーム名がタイトルとして表示されます。しかし、式を使用することで実行時に動的にタイトルを変更することもできます。



フォームタイトルに式を使用する

1. フォームエディタを開きます。
2. フォーム特性（*Alt+Enter*）を開きます。
3. フォーム名特性に移動します。
4. 式特性でズーム（または、*fx* ボタンをクリック）します。
5. 実行時にタイトルとして評価される式を定義します。この例では、担当者毎のユーザ ID が表示されることになります。

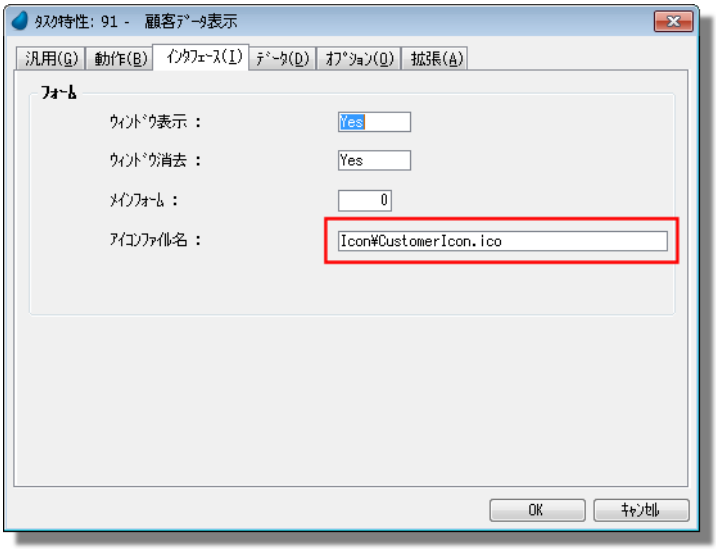
これで、実行時に式で指定された内容がタイトルバーに表示されます。修正前の（固定のフォーム名も残ります。これは開発時のみ表示されます。

ヒント : *タイトルが長すぎてフォーム名として入力できない場合も、この方法で指定できます。*

フォームのアイコンを設定するには

必要であれば、特定のフォームに表示するアイコンを設定することができます。このアイコンは、フォームの左上に表示され、フォームが最小化された場合もタスクバーに表示されます。この設定により、プロジェクトレベルで設定されたアイコンは上書きされます。

フォームのアイコンを選択する



1. **タスク特性**（*Ctrl+P*）を開きます。
2. **インタフェース**タブを選択します。
3. **アイコンファイル**特性で**ズーム**して、設定したいアイコンファイルを選択します。

ヒント : *アプリケーションがインストールされた場合、ユーザは開発時のセットアップ環境と同じとは限りません。このため、パス名を固定にしないように定義する必要があります。パス名が指定されていない場合、Magic xpa は .edp ファイルが存在する作業フォルダ（**%WorkingDir%**）を検索します。**アイコンファイル名**には、相対パスや論理名を使用することもできます。*

タスク

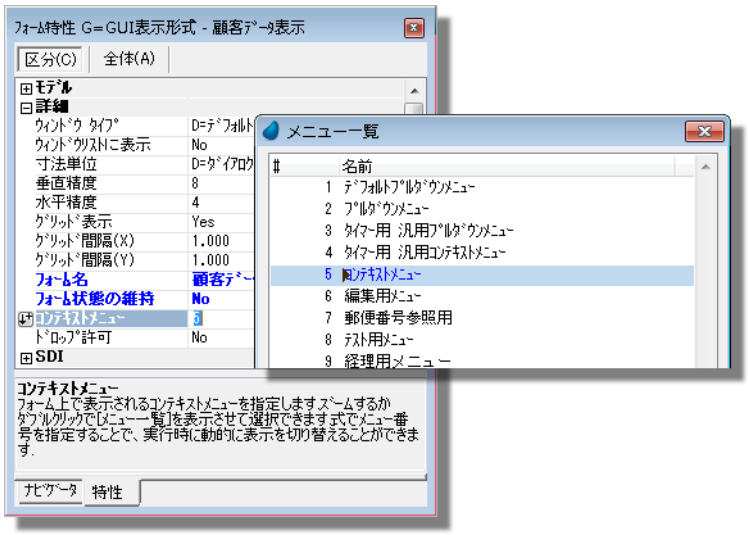
コントロールにデフォルトのコンテキストメニューを設定するには

メニューリポジトリでは、アプリケーション全体のコンテキストメニューを作成することができます。また、これをオーバライドし、各フォーム毎のデフォルトメニューを作成するもできます。

フォームにデフォルトのコンテキストメニューを設定する

必要条件： メニューがすでに作成されているものとします。

1. フォーム特性（コントロールが選択されていない場合は、*Alt+Enter*）を開きます。

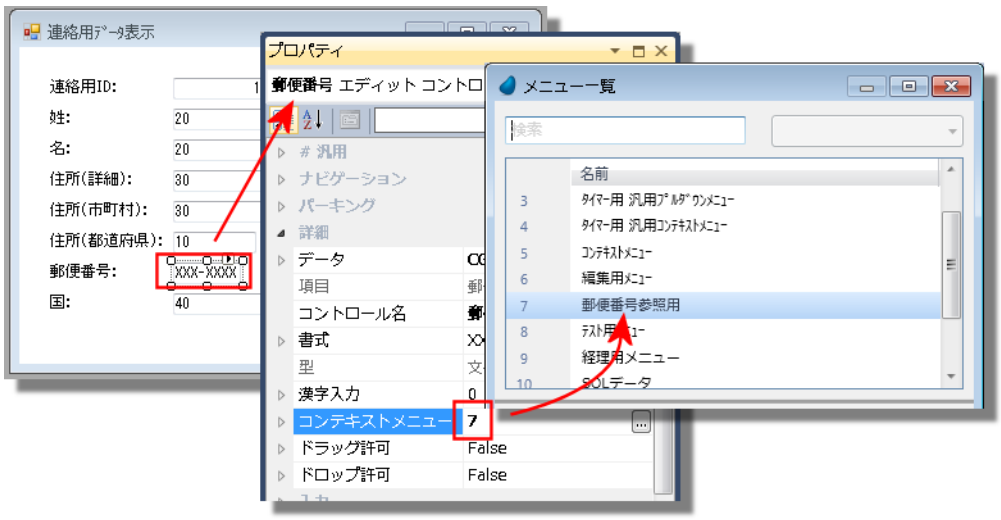


2. **コンテキストメニュー**特性で**ズーム**してメニューを選択します。
3. 代わりに、右側の**式**特性で**ズーム**（または ***fx*** ボタンの**クリック**）して実行時に式番号として評価される式を定義することもできます。

これで、ユーザがフォーム上でマウスの**右クリック**を行うと、コンテキストメニューが表示されます。

個別のコントロールにコンテキストメニューを設定するには

メニューリポジトリでは、アプリケーション全体のコンテキストメニューを作成することができます。また、これを上書きし、各フォーム毎のデフォルトメニューを作成するもできます。さらにコントロールレベルのコンテキストメニューを作成することもできます。この例では、郵便番号を検索するための特別なコンテキストメニューを作成していきます。



コントロールレベルのコンテキストメニューを作成する

必要条件： メニューがすでに作成されているものとします。

1. **コントロール特性**（コントロールが選択されている場合は、*Alt+Enter*）を開きます。
2. **コンテキストメニュー**特性でズームしてメニューを選択します。
3. 式で定義することもできます。左側の矢印 ▷ をクリックすると式特性が表示されます。式特性でズームし（または … ボタンをクリックして）式エディタに実行時にメニュー番号として評価される式を定義することもできます。

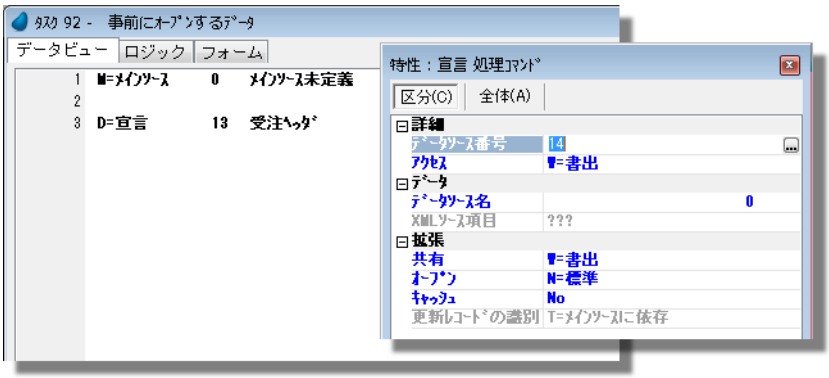
これで、ユーザが郵便番号の入力欄でマウスの右クリックを行うと、コンテキストメニューが表示されます。

バッチタスクから繰り返し呼ばれるタスクのパフォーマンスを改善するには

あるバッチタスクが繰り返し別のバッチタスクによって呼ばれる場合、パフォーマンスが悪くなる場合があります。これは、サブタスクが使用するデータソースを開いたり閉じたりするために必要なオーバヘッドによるものと考えられます。このような場合、以下の方法によってパフォーマンスを改善することができるかもしれません。

バッチのサブタスクのパフォーマンスを改善する

1. 最初に行う最も重要なステップは、サブタスクが使用するデータソースを親タスクであらかじめオープンされているように定義することです。異なるファイルが複数使用する場合、オープンするためのオーバヘッドはファイル数分増えることになります。レコードを読み込むためのオーバヘッドよりファイルをオープンするための方が一般的に大きくなります。

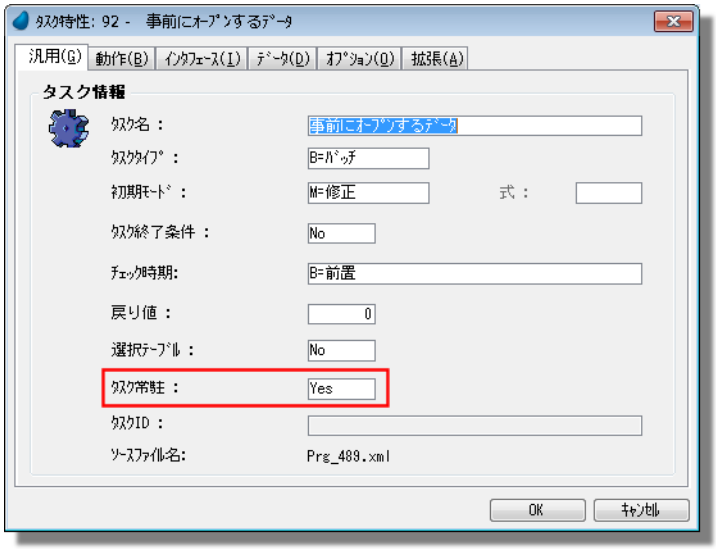


親タスクでデータソースが使用されない場合、**データビューエディタ**で**宣言**のヘッダ行を作成することで、事前にこのデータソースをオープンしておくことができます。宣言指定の**アクセス**特性は、サブタスクがデータソースをオープンする場合に使用される**アクセス**特性の内容と合っていなければなりません。

2. 次に、バッチタスクを常駐タスクとして設定します。この設定により起動されるたびに、バッチタスクが再読込されることを防ぎます。

   サブタスクを常駐タスクとして定義するには以下のようにします。

   - **タスク特性**（*Ctrl+P*）を開きます。
   - **汎用**タブを開きます。
   - **タスク常駐**特性を *Yes* に設定します。

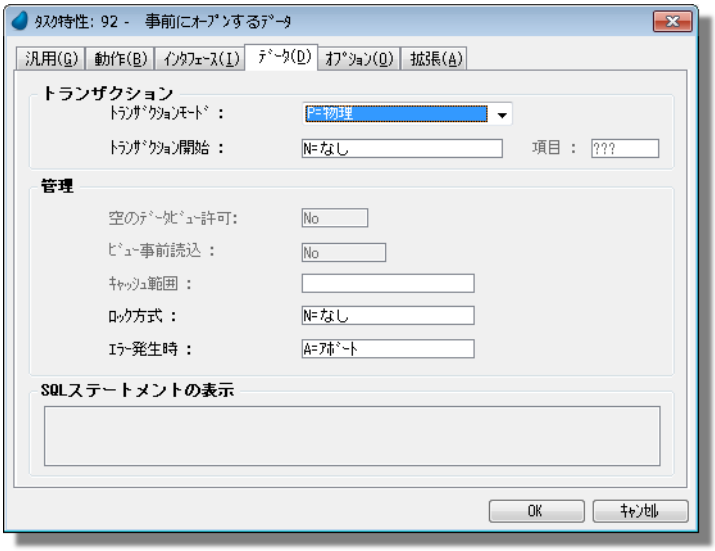


注： 常駐タスクの使用は現在推奨しておりません。使用される場合も、あまり多用しないようにお願いします。

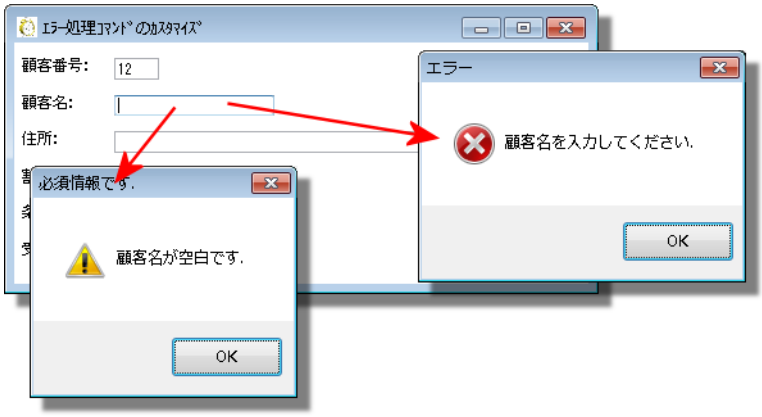
3. 最後に、親タスクのトランザクションを**遅延モード**にしないように設定します。トランザクションモードが遅延に設定されていると、すべてのサブタスクトランザクションは蓄積され、パフォーマンスを低下させる要因になります。トランザクションモードを確認するには、以下のようにします。

   - 親タスクに移動します。
   - **タスク特性**（*Ctrl+P*）を開きます。

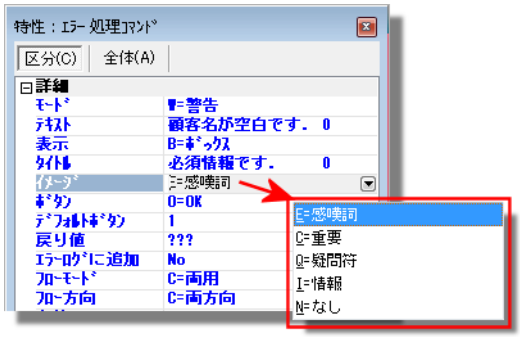
- データタブを開きます。
- **トランザクションモード**特性を、タスクとして必要な値に設定します。この例では、トランザクション処理を行わない設定になっています。大規模なバッチタスクのほとんどは帳票用のためデータの更新は行われないのでトランザクション処理を行う必要はありません。



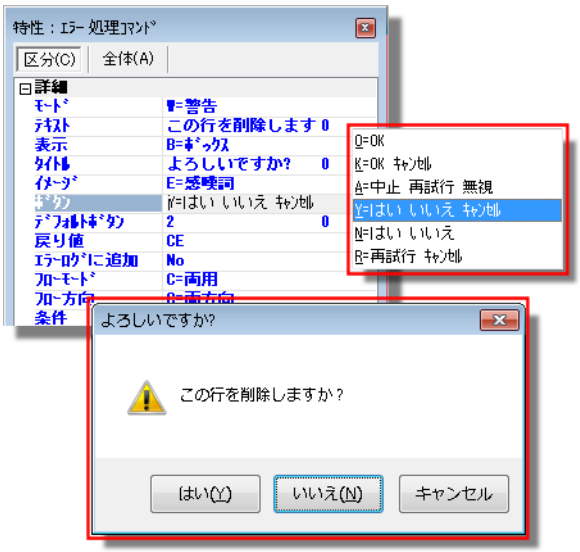
エラー処理コマンドの表示内容をカスタマイズするには



エラー処理コマンドは、ユーザに情報を簡単に伝える為に利用できます。また、ユーザの選択内容をプログラムに返すこともできます。エラー処理コマンドには、***エラー***と***警告***の 2 つのモードがあります。各々、プログラムでの必要性に応じてカスタマイズすることができます。



エラーの場合、値を入力するか、式で指定することでテキストやタイトルバーを変更することができます。また、表示するイメージを変更することもできます。この場合、5 つのイメージの中から選択することができます。



警告の場合は、プッシュボタンの設定を変更することができます。また、ユーザがどのボタンをクリックしたかをプログラムに返すことができます。この警告ウィンドウは、Microsoft のオブジェクトを使用しています。このため、オブジェクト自体や戻される結果は、OS によって設定されます。

ここでは、**ボタン**特性は、***はい / いいえ / キャンセル***が設定され、**デフォルトボタン**は、2 番目のボタンに設定されます。数値型変数 Z は、結果を保存するために使用されます。

返される値は、以下の通りです。

- はい……6
- いいえ……7

- キャンセル……2

選択内容や使い方についての詳細は、リファレンスヘルプのエラー処理コマンドの特性の説明を参照してください。

サブフォームの表示内容を実行時に差し替えるには

通常、各サブフォームは、1 つのプログラムまたはタスクを表示します。しかし、サブフォームの内容を指定した目的で動的に変更することができます。たとえば、この例では、ラジオボタンから起動するタスクを選択するようにしています。そして、選択内容に応じて異なるタスクを呼び出しています。

1. **サブフォーム特性**に、サブフォームの**コントロール名**を定義します。
2. 表示したいプログラムまたはサブタスクを呼び出すために、**コール**処理コマンドを定義します。
3. **出力**特性にサブフォームの**コントロール名**を設定します。この例では、左側のサブフォームが注文書かユーザを表示するために使用されます。

これで、プログラムを実行すると、要求したタスクが適切なエリアで動作するようになります。

サブフォームは、同様の目的でタブコントロールと組み合わせて使用することもできます。この場合、サブフォームを各タブに割り当てます。**コール**処理コマンドは必要ありません。サブフォームよりもフレームでこの方法を利用することができます。

第 6 章： ロジックの拡張

電子メールを送信するには

Magic xpa には電子メールを送受信するための機能が組み込まれています。

電子メールを送信するには、以下の手順でプログラムを作成します。

1. サーバと接続します。
2. 電子メールを送信します。

Magic xpa ではこれらを個別に処理します。

サーバと接続する

MailConnect() 関数を使用してサーバと接続します。この場合、あらかじめメールサーバの名前を知っておく必要があります。それが何なのかわからない場合は、電子メール用のソフトウェアの設定オプションで確認してください。構文は、以下の通りです。

MailConnect(*type*, *server*, *user*, *password*)

パラメータ：

- *Type* ：接続するサーバのタイプ（1=SMTP, 2=POP3, 3=IMAP）
- *Server* ：サーバのアドレス
- *Userid* ：サーバに接続する際に必要なユーザ ID
- *Password* ：ユーザのパスワード

例えば、juno.olympus.com というアドレスの SMTP サーバに接続する場合は以下のように設定します。

```
MailConnect(1,'juno.olympus.com','','')
```

戻り値は、接続しているサーバの種類に依存します。関数の詳細は『リファレンスヘルプ』を参照してください。

電子メールを送信する

サーバに接続したら、**MailSend()** 関数を使用することで電子メールを送信することができます。構文は以下の通りです。

MailSend(*From*, *To*, *Cc*, *Bcc*, *Subject*, *Message*, *Attachment*)

パラメータ：

- *From* ：送信元のアドレス。内容はチェックされません
- *To* ：宛先のアドレス。正しくない場合メールは送信されません。
- *Cc* ：CC へのアドレスのリスト
- *Bcc* ：BCC へのアドレスのリスト
- *Subject* ：メールの件名を表すテキスト
- *Message* ：メールの本文となるテキスト
- *Attachment* ：添付ファイルのファイル名

以下は設定例です。

```
MailSend ('jenny@frigates.com', 'fred@aaawidgets.com', '', '', 'May Invoice', 'Attached is
your May invoice', 'F:¥Invoices¥aaa_11_2007.pdf')
```

通常は、ハードコーディングされた文字列ではなくデータ項目を使用することになります。アドレスは、カンマ区切りにすることで複数指定することができます。また添付ファイルも複数指定することができます。

戻り値は数値で、**0** が成功を意味しています。戻り値が **0** でない場合、**MailError()** 関数を使用してエラーメッセージに変換することができます。戻り値を 'BP' という項目に格納する場合は、以下のような式が定義できます。

```
'エラーコードは：'&Str (BP,'5NC')&' '&MailError (BP)
```

この式によってエラーコードとメッセージの両方を表示させることができます。

電子メールにファイルを添付するには

Magic xpa の関数で電子メールを送信する場合、**MailSend()** 関数を使用します。構文は以下の通りです。

MailSend(*From*, *To*, *Cc*, *Bcc*, *Subject*, *Message*, *Attachment***)**

添付ファイルを送信する場合は、**MailSend()** 関数の *Attachment* パラメータを指定する必要があります。カンマ区切りでファイル名を連結することで複数のファイルを送信することができます。

以下は設定例です。

```
MailSend ('jenny@frigates.com', 'fred@aaawidgets.com', '', '', 'May Invoice', 'Attached
is your May invoice', 'F:¥Invoices¥aaa_11_2007.pdf')
```

この例では、F:¥Invoices¥aaa_11_2007.pdf が添付ファイルになります。

参照： 「電子メールを送信するには」（143 ページ）

電子メールを受信するには

電子メールを受信する場合、以下の手順で行います。

1. メールを受信するには、POP3 または IMAP サーバに接続（送信する場合は、SMTP サーバでした）する必要があります。この接続は、通常ユーザ ID とパスワードが必要となるため、以下の例のような式を定義します。

   ```
   MailConnect (3,'juno.olympus.com','FredZ','rabbit16')
   ```

   この例で設定されている *3* は、IMAP サーバに接続することを示すサーバタイプです。その他の３つのパラメータは各々サーバ名、ユーザ ID、およびパスワードです。
2. 戻り値が正の数値なら、キュー内の電子メールの数を示します。この値を保存しておきます。
3. 次に、メッセージを取得する処理を実行する必要があります。メッセージ数と同じ数だけのインデックスを使用してメッセージを取得します。各メッセージ毎に、電子メールの様々な情報を取得する関数を使用することができます。例えば、Y をインデックスとします。この場合、以下の関数を実行できます。

   ```
   MailMsgDate(Y)
   MailMsgFrom(Y)
   MailMsgSubj(Y)
   MailMsgText(Y)
   MailMsgFiles(Y)
   ```

   これらの関数によって取得された情報をすべて保存します。

   メッセージを読み込んだら、以下の関数で削除できます。

   ```
   MailMsgDel(Y)
   ```

   メッセージが削除できない場合、メッセージはメールボックス内に存在していることになります（メールボックスに保存することは、テスト中であれば良い考えです）。
4. 終了する場合は、以下の関数でサーバ接続を切断します。

   ```
   MailDisconnect(2, 'TRUE'LOG)
   ```

   これでセッションが切断され、メールがサーバから削除されます。この例の *2* はサーバタイプで、受信サーバからの切断を意味しています。サーバと切断されメールが削除されると、論理値 'TRUE'LOG が返ります。

サーバのタイプ

メールを受信する場合、２種類のサーバタイプがあります。POP3 サーバに接続する場合は、新規メールのみ受信します。IMAP サーバに接続した場合は、サーバに存在するメールをすべて受信します。

参照： これらの関数の詳細情報は、『リファレンスヘルプ』を参照してください。

添付ファイルを受信するには

電子メールに添付されているファイルを取得するには、以下の手順でロジックを定義します。

1. 最初に **MailMsgFiles()** 関数を実行して添付付されているファイルの数を取得する必要があります。構文は以下の通りです。
   MailMsgFiles(*Y*)

パラメータ：

- *Y* ：メールのインデックス

この関数は、メールに添付されているファイルの数を返します。

2. 次に、各添付ファイル毎に **MailMsgFile()** 関数を実行してファイル名を取得します。構文は以下の通りです。
   MailMsgFile(*Y*, *Z*)

パラメータ：

- *Y*：メールのインデックス
- *Z*：添付ファイルのインデックス

この関数は、添付ファイルのファイル名が返ります。

3. 最後に、各添付ファイル毎に **MailFileSave()** 関数を実行してファイルを取得します。構文は以下の通りです。
   MailFileSave(*Y*, *Z*, *FileName*, *Flag*)

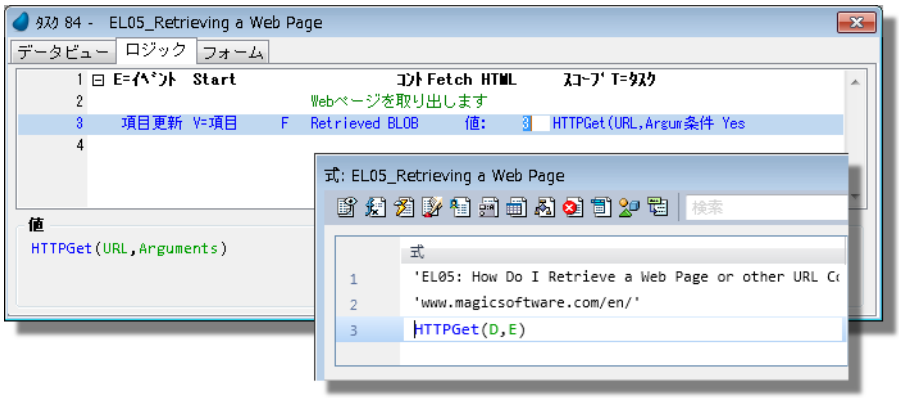
パラメータ：

- *Y*：メールのインデックス
- *Z*：添付ファイルのインデックス
- *FileName*：保存する添付ファイル名
- *Flag*：ファイルを上書きするかどうかを指定するフラグ

この関数は、ファイルの保存が成功したかどうかを示す数値が返ります。

Web ページやその他の URL コンテンツを受信するには

HTTPGet() 関数や **HTTPCall()** 関数を使用することで URL を指定して、Web ページやその他の URL コンテンツを取得することができます。



HTTPGet() 関数を使用する

1. **HTTPGet()** 関数を使用して Blob 項目を更新します。構文は以下のとおりです。
   HTTPGet(*URL*, *Arg1*, *Arg2*, . . .)

パラメータ：

- *URL*：Web サイトの URL
- *ArgX*：追加で指定するヘッダ情報

2. **Blb2file()** 関数を使用することで Blob 項目の内容をファイルに返還できます。
3. BLOB の内容を拡張子 **.htm** のファイルに保存すると、Web ブラウザで参照することができます。

参照： **HTTPGet()** 関数の詳細情報は、『リファレンスヘルプ』を参照してください。

HTTPCall() 関数を使用する

1. **HTTPCall()** 関数を使用して同じ処理が可能です。構文は以下のとおりです。
   HTTPCall(*Method*, *URL*, *Arg1*, *Arg2* . . .)

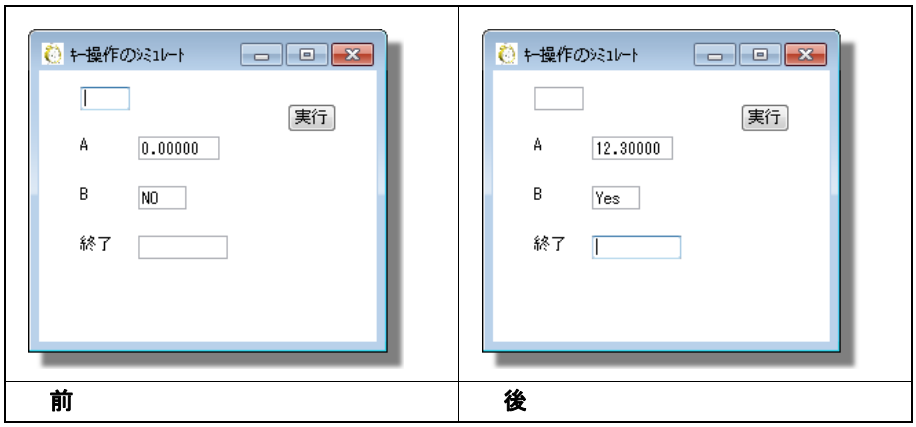
パラメータ：

- *Method*：HTTP メソッドのメソッド名、ここでは、”Get” を指定します。
- *URL*：Web サイトの URL
- *ArgX*：追加で指定するヘッダ情報

参照： **HTTPCall()** 関数は Ver1.9 で追加されました。今後は、こちらの関数を使用するようにしてください。

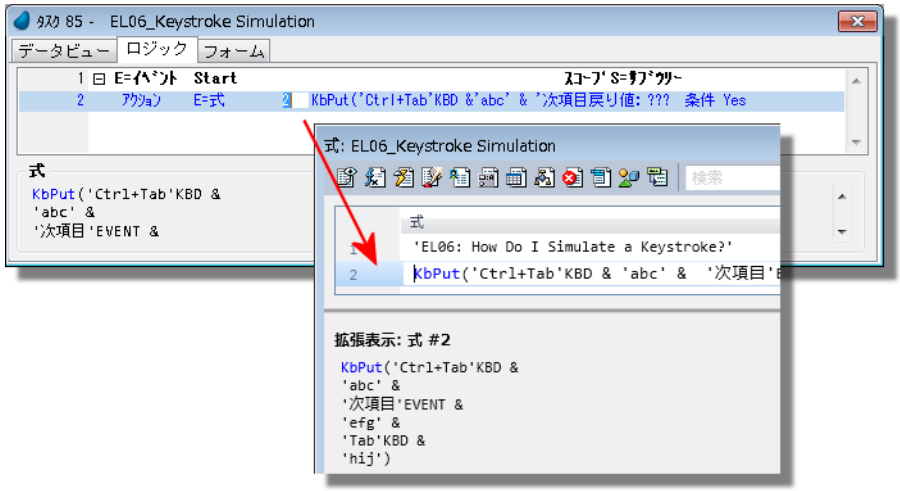
キー操作をシミュレートするには

Magic xpa でキー操作をシミュレートするには２つの方法があります。１つは、**KBPut()** 関数を使用する方法で、もう１つはイベント実行処理コマンドを使用する方法です。**KBPut()** は汎用的な関数で、文字を入力したり、長いマクロ文を作成することができます。他方、イベント実行処理コマンドは、色々な種類のイベントを発行したりするなど、より柔軟性を提供します。ここで、**KBPut()** 関数について説明します。



前 後

KbPut() 関数を使用する



KbPut() 関数の構文は以下のとおりです。

KbPut(*String*)

パラメータ：

- ***String***：以下に示す書式が指定された文字列です。連結演算子（&）は、文字列を結合する際に使用されます。

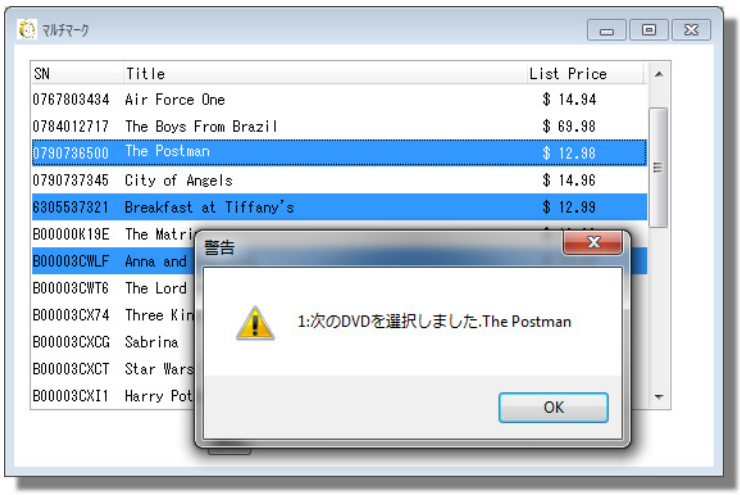
KbPut での文字内容	構文	例	簡単な入力方法
文字 , 数字	クォーテーションで囲まれたテキスト	‘123.2’	
キー操作	**KBD** リテラルを使用したキー名	‘**Tab**’KBD	コンテキストメニューから**キーボード**を選択し、キー設定ダイアログから指定します。
イベント	**EVENT** リテラルを使用したイベント名	‘ **次項目** ’EVENT	コンテキストメニューから**イベント**を選択し、イベントテーブルから指定します。

この例では基本的には、‘*Tab*’KBD と ‘ **次項目** ’EVENT は同じ動作になります。**キーボード割付**テーブルで設定が変更される可能性があるため、EVENT による指定を使用したほうが安全です。

注： **KBPut()** 関数は、.NET コントロールや起動された Windows のダイアログには影響しません。

参照： 第 4 章：「Magic エンジンをイベント駆動型で動作させるには」（50 ページ）

複数のレコードをマークしたり、マークしたレコードを処理させるには



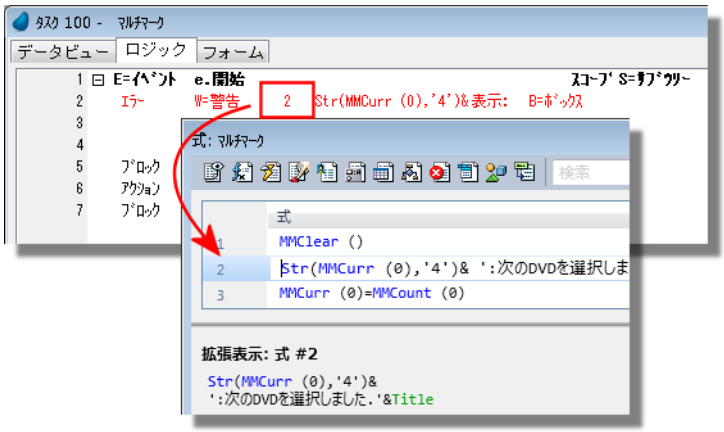
ユーザがリストから複数の項目を選択できるようにすることは便利な機能です。Magic xpa にはこのような機能が ***マルチマーキング機能***として組み込まれています。

テーブルをマルチマーキング対応に設定する

テーブルコントロールのコントロール特性で、マルチマークを利用可能になるように設定する必要があります。

1. テーブルコントロール特性で、**マルチマーキング**特性を ***True*** に設定します。
   これで、ユーザはこのテーブルから複数のレコードを選択することができるようになります。

マークされたレコードを処理する



レコードが選択されたら、これらを処理する必要があります。このような場合、一般的に別のテーブルに書き込んでから処理することになりますが、この例では、ただ単にユーザにメッセージを表示するだけの処理を説明します。

1. マークされたレコードを処理する時点で実行するイベントを作成します。この例では、e. 開始イベントを実行するプッシュボタンを使用します。
2. イベントに対応するロジックユニットでは、現在のレコードであるかのようにレコードを処理します。この例では、各レコード毎に３つのメッセージが用意され、そのうち１つを選択するようになっています。ブロック処理コマンドによるループはありませんが、これによってループしたような処理が実行されます。

マルチマーク関数を使用して処理する

1. イベント内で **MMStop()** 関数を使用することで、マークされたすべてのレコードが処理される前に処理を終了することができます。**MMStop()** 関数が実行された場合、エンジンは **MMStop()** が実行された時点で処理しているレコード上にパークします。

2. 以下の式を指定したブロック処理コマンドを使用することで、すべてのレコードが処理された後に、ロジックを実行させることができます。

```
MMCurr()=MMCount(0)
```

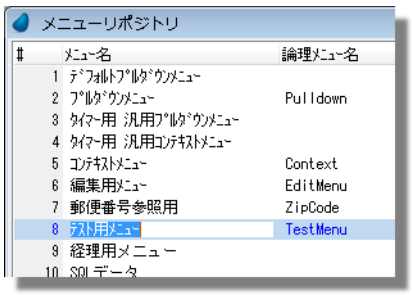
MMCurr() 関数は、現在までに処理されたレコード数が返り、**MMCount(0)** 関数はレコードの総数が返ります。この例では、最後のレコードが処理された後にマーキングが解除されます。

3. **MMClear()** 関数は、レコードのマークを解除します。

メニューを非表示 / 無効 / チェック表示に設定するには

Magic xpa では、エンドユーザ用のメニューを作成 / 制御することができます。プルダウンメニューやコンテキストメニューを作成することができ、フォームやコントロールにコンテキストメニューを設定することができます。また、実行時に有効にしたり無効にしたりすることができます。ここではこのような処理について説明します。

論理メニュー名を指定する



メニューを個別に有効化 / 無効化するには、ユニークな**論理メニュー名**を指定する必要があります。この設定は、メニューリポジトリで行います。設定したいメニュー項目に対して、**論理メニュー名**を入力します。この名前は実行時は表示されないため、どのような名前も入力できます。この例では、２つの論理メニュー名、EditMenu と TestMenu が定義されています。

論理メニュー名を定義すると、**MnuCheck()** と **MnuEnable()** 関数を使用して有効化 / 無効化やチェック表示の切り替えを実行時に行うことができます。

メニューパス関数を使用する

メインメニューのサブメニューにも、**論理メニュー名**が設定されており、これらは、メニューパスを文字列で取得することができます。例えば、デフォルトのプルダウンメニューの下に、UtilityMenu と名付けられたメニューが定義されており、その下には SetupMenu という名前のメニューが定義されている場合は、パスは以下の通りになります。

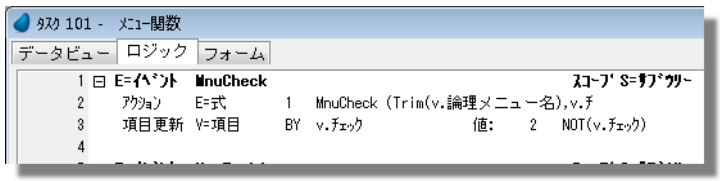
```
UtilityMenu¥SetupMenu
```

この値は、**MnuAdd()** 関数で使用することができます。

メニュー番号を指定する

メニュー名に加え、メニューは **MENU** リテラルによって参照することもできます。上記の例では、Edit menu は **'2'MENU**、Testing Menu は **'3'MENU** と指定できます。**MnuAdd()** 関数で **MENU** リテラルを使用します。

MnuCheck() 関数を使用する

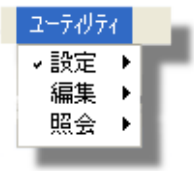


MnuCheck() 関数は、メニュのチェック表示の切り替えを行います。構文は以下の通りです。

MnuCheck(*EntryName*, *Boolean***)**

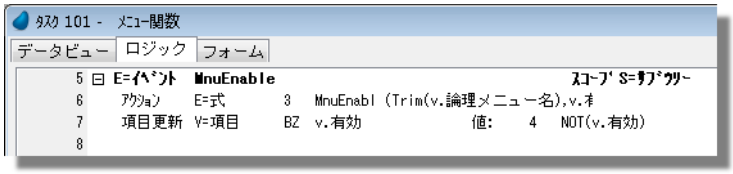
パラメータ：

- ***EntryName*** ：**メニュー定義**テーブルに定義されている**論理メニュー名**です。**論理メニュー名**は、実行時にメニューとして表示されないので任意の名前が定義できます。
- ***Boolean***：True が指定された場合、メニューにチェックが表示されます。False の場合、チェックは表示されません。



この例では、式が実行されるとメニューの設定がチェックされます。

MnuEnabl() 関数を使用する



MnuEnabl() 関数は、メニューの有効 / 無効を切り替えます。構文は以下の通りです。

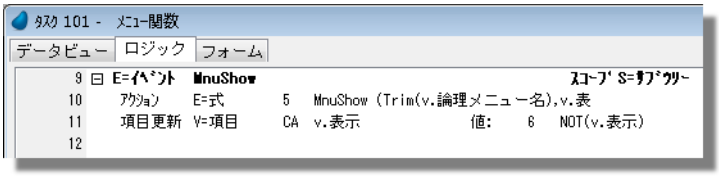
MnuEnabl(*EntryName*, *Boolean***)**

パラメータ：

- *EntryName* ：メニュー定義テーブルに定義されている**論理メニュー名**です。**論理メニュー名**は、実行時にメニューとして表示されないので任意の名前が定義できます。
- *Boolean*：True が指定された場合、メニューは有効になります。False の場合、無効になります。

この例では、式が実行されるとメニューの設定が無効になります。

MnuShow() 関数を使用する



MnuShow() 関数は、メニューを表示するか否かを指定します。構文は以下の通りです。

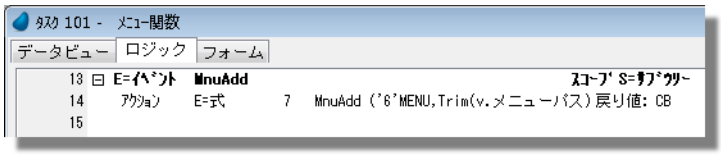
MnuShow(*EntryName*, *Boolean***)**

パラメータ：

- *EntryName* ：メニュー定義テーブルに定義されている**論理メニュー名**です。**論理メニュー名**は、実行時にメニューとして表示されないので任意の名前が定義できます。
- *Boolean*：True が指定された場合、メニューは表示されます。False の場合、表示されません。

この例では、式が実行されるとメニューの設定は定義はされていますが、表示されていません。

MnuAdd() 関数を使用する



MnuAdd() 関数は、メニュー全体（サブメニューを含む）をメニューに追加することができます。メニューリポジトリにメニューを作成し、メニュー番号を **MENU** リテラルを付加して指定することで追加することができます。

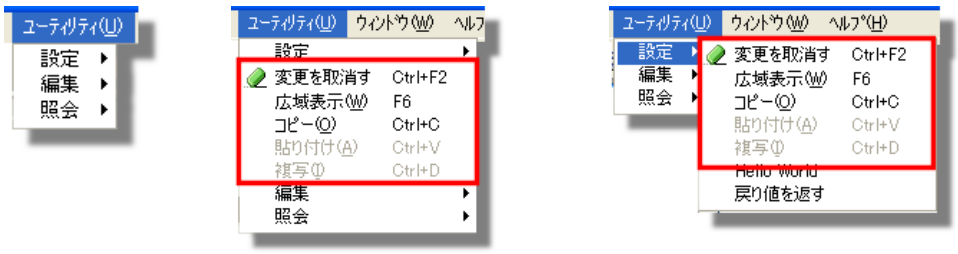
さらに、新しいメニューがどこに位置するかを指定するメニューパスを使用します。構文は以下の通りです。

MnuAdd(*MenuEntry*, *MenuPath***)**

パラメータ：

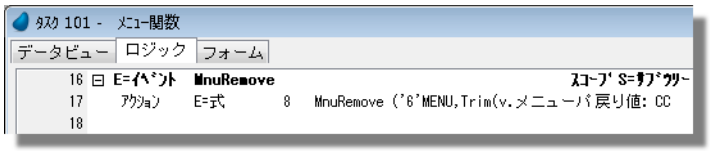
- *MenuEntry* ：メニュー番号です。この例で指定されている、**'3'MENU** は、メニューリポジトリの３番目のメニューを示しています。
- *MenuPath*：メニューの追加先のメニューパスです。この例では、'UtilityMenu¥SetupMenu' が指定されており、設定メニューの下にメニューリポジトリの３番目のメニュー内容が追加されます。'UtilityMenu¥SetupMenu¥' と指定した場合、設定メニューのサブメニューとして追加されます。

注： メニューパスの後尾にバックスラッシュ（¥）が付いた場合と付かない場合で、結果が異なることに注意してください。



元のメニュー　　UtilityMenu¥SetupMenu の場合　　UtilityMenu¥SetupMenu¥ の場合

MnuRemove 関数を使用する



MnuRemove() 関数は、**MnuAdd()** 関数で追加されたメニューを削除します。構文は以下の通りです。

MnuRemove(*MenuEntry*, *MenuPath***)**

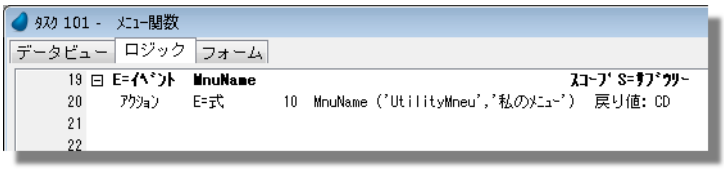
パラメータ：

- *MenuEntry* ：メニュー番号です。この例で指定されている **'3'MENU** は、メニューリポジトリ上の３番目のメニューを示しています。
- *MenuPath*：メニューの削除先のメニューパスです。前述の **MnuAdd()** 関数で指定されたメニューパスが指定されています。

MnuReset() 関数を使用する

MnuReset() 関数は、メニューをデフォルト状態にリセットします。メニュー全体を初期化するだけのため、パラメータはありません。

MnuName() 関数を使用する



MnuName() 関数は、メニューの名前を変更することができます。メニューの表示が変わるだけで、動作には影響しません。構文は以下の通りです。

MnuName(*EntryName*, *EntryText***)**

パラメータ：

- *EntryName* ：メニュー定義テーブルに定義されている**論理メニュー名**です。**論理メニュー名**は、実行時にメニューとして表示されないので任意の名前が定義できます。
- *EntryText* ：実行時に表示されるメニューを表すテキストです。この例では、ユーティリティという名前を私のメニューに変更するように指定されています。

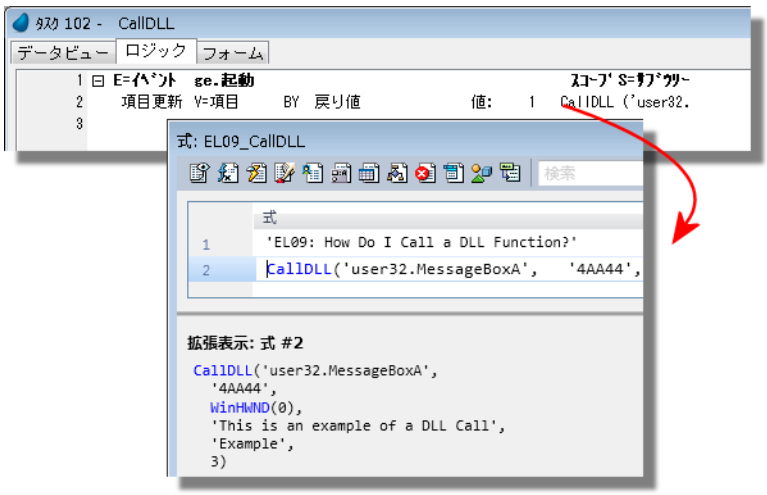
DLL 関数を呼び出すには

Windows 環境や、購入したパッケージには、たくさんの利用可能な DLL があります。独自に作成したものを使用する場合も考えられます。各々の DLL の呼び出し方は異なっていますが、呼び出し先に合わせてパラメータを設定する必要があります。ここでは、Windows API を呼び出してメッセージボックスを表示させる簡単な例を元に説明します。

Magic xpa では、**CallDll()** 関数や**コール UDP** 処理コマンドを使用して、DLL を呼び出すことができます。ここでは両方の呼び出し方で説明しています。

CallDLL() 関数を使用する

CallDll() 関数には 3 種類あり、呼び出す DLL に応じて使い分けます。Windows API を呼び出す場合は、stdcall 規約で呼び出すため **CallDLLS()** 関数を使用します。



1. コール処理を定義したい行で *F4* を押下します。
2. *U* を押下して**項目更新**処理コマンドを作成するか、プルダウンリストから処理コマンドを選択します。
3. 戻り値を格納するデータ項目を選択します。この場合、呼び出す DLL は数値を返すため、戻り値には数値型項目を使用します。数値は、ユーザからの応答を表します。
4. **値**カラムで**ズーム**して、起動する DLL を定義します。ここでは、**CallDLLS()** 関数を使用して呼び出します。構文は以下の通りです。

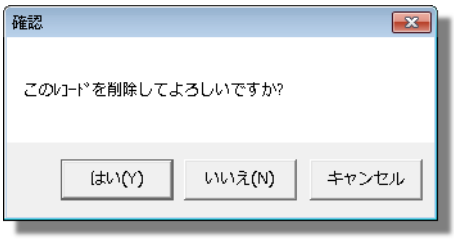
   CaLLDLLS(*DLLName*, *ArgString*, *Arg1*, *Arg2*. . . **)**

パラメータ：

- *DLLName* ：DLL の名前です。この例では、user32.MessageBoxA になります。
- *ArgString* ：パラメータのデータタイプを表す文字列です。各パラメータに対して 1 文字が対応します。最後の 1 文字は、戻り値を表します。
- *ArgX* ：渡されるパラメータです。この例では、4 つのパラメータが渡されます。(Window ハンドル値、2 つの文字列、およびボックスのスタイルを表す数値)

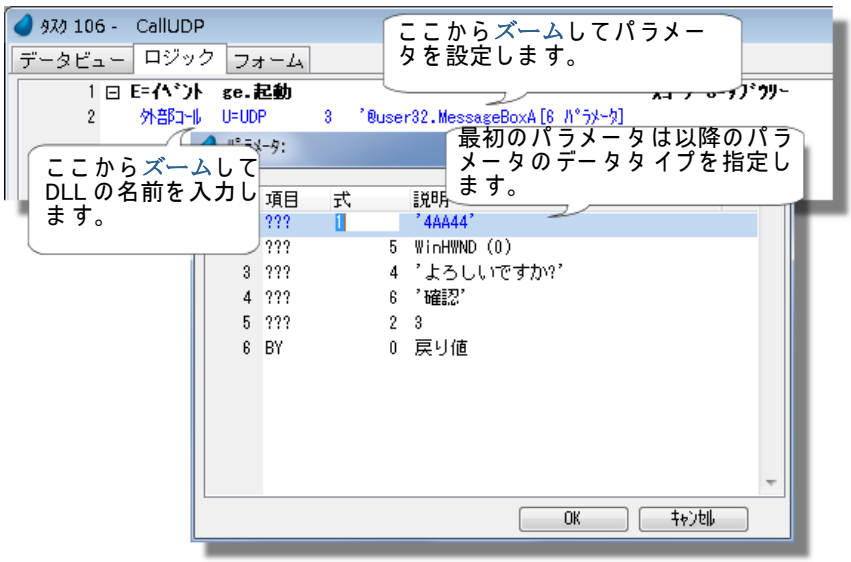
文字	データのタイプ
1	Char
2	Short
W	Wchar
4	Long
F	Float
8	Double
D	Double pointer
E	Float pointer
L	Long pointer
A	Null で終了する String のアドレス
U	LPCWSTR
V	Void pointer
0	Void

CallDLLS() 関数を定義し実行すると、Windows のメッセージボックスが表示されます。ユーザがボタンをクリックすると、クリックされたボタンに応じた戻り値（6= はい、7= いいえ、2= キャンセル）が返ります。



ヒント : このように、OS に依存する関数を使用する場合、Magic xpa のタスク内でカプセル化することを推奨します。このようにすると、別の方法で呼び出したりインストール内容が変わったりした場合に変更が簡単になります。

コール UDP 処理コマンドを使用する



コール UDP 処理コマンドを使用して DLL を呼ぶ場合も、**CallDLL()** 関数を使用した場合と同じように利用できます。

1. コール処理を定義したい行で *F4* を押下します。
2. *I* を押下して**外部コール**処理コマンドを作成し、次のカラムに移動します。
3. *U* を押下して **UDP** を選択します。次のカラムに移動します。
4. ズームして**式エディタ**を開き、起動する DLL の名前を入力します。DLL 名の前に @ を指定します。この例では、Windows の user32.MessageBoxA を呼び出すため以下のようになります。
   ```
   @user32.MessageBoxA
   ```
   Enter を押下して**ロジックエディタ**に戻り、次のカラムに移動します。
5. ズームして**パラメータ**テーブルを開きます。ここでは **CallDLL()** 関数の場合と同じ書式で設定します。最後の行には、戻り値を返す変数項目を定義します。
   - 最初のパラメータには、DLL が必要な各パラメータのデータタイプを示す文字を指定します。MessageBoxA 関数の場合は、５つの文字を指定します。*4* は long integer、*A* は Null で終わる文字列を表します。
   - その他のパラメータに対しては必要に応じて項目や**式**特性からズームして設定します。最後の 1 文字（4）は、戻り値を表しています。
6. *Ctrl+Enter* を押下して**外部コール**処理コマンドの**特性**シートを開きます。**規約**特性に ***STDCALL*** と入力します。

構造体のパラメータを DLL に渡すには

Magic xpa は、BLOB 項目を利用したバッファを扱うことができます。バッファ関数を使用することで構造体を構築することができ、構造体にデータを渡したり、受け取ったりすることができます。

構造体を作成する

1. BLOB 項目を定義します。ここには、バイナリデータが格納されます。
   バッファは、バイナリデータとして格納されます。BLOB 自体はデータ長を意識していませんので、データを格納したり取り出す際は、該当データの格納位置とデータ長を明示的に指定する必要があります。
2. 数値データをバッファに追加する場合は、**BufSetNum()** 関数を使用します。
   例： BufSetNum ('H'VAR,1,B,3,8)
   BufSetNum() 関数のパラメータは以下の通りです。
   - バッファ項目……全ての バッファ関数で指定します。値が追加されるバッファ項目として BLOB 型の項目を指定します。
   - 位置…… 値が追加される位置です。

   この例では、位置を「1」にしています。構造体の最初であることを示しています。(Magic xpa では、位置指定は 1 から始まります。)
   バッファは、連続したデータの集まりです。そのため、バイナリデータのどこに追加するかを指定することはとても重要です。
   値を連続して指定する場合、値を設定する位置と長さを考慮に入れて位置を指定してください。
   例えば、「位置 =1」、「長さ =2」のデータの後に続くデータの位置は、「3」になります。
   - 値…… 構造体に追加する値です。関数のタイプに合っていなければなりません。例えば、**BufSetNum()** 関数や **BufSetBit()** 関数の場合は、数値であり、**BufSetAlpha()** 関数の場合は、文字データにしなければなりません。

   ここには、項目や固定値を組み合わせた定義式でも指定できます。
   - 記憶形式……**BufSetVariant()** 関数以外のすべての **BuffSet()** 関数の四番目のパラメータで、数値によって記憶タイプを指定します。

   例えば、数値を float タイプでバッファに追加するといった指定を行います。有効な記憶タイプは、別表で示しています。
   - 長さ…… 最後のパラメータは、追加するデータの長さです。記憶タイプの長さと合っていなければなりません。

   この例では、「8」に設定されています。

参照： バッファ関数の記憶形式タイプの詳細は、Magic xpa の『リファレンスヘルプ』を参照してください。

Magic xpa からバッファを送る

コール UDP 処理 コマンドを使用して外部の dll を呼び出す際、バッファを使用してパラメータの受け渡しをすることができます。以下の例では、PC のシステム日付を取得する Windows の API を呼び出しています。

変数項目を準備する

1. オンラインタスクを作成し変数項目を定義します。

シンボル名	名前	型	書式
A	バッファ用変数	B=BLOB	
B	年数格納用変数	N= 数値	4
C	月数格納用変数	N= 数値	2
D	週の何日目かを格納する変数	N= 数値	1
E	日数格納用変数	N= 数値	2
F	時数格納用変数	N= 数値	2
G	分数格納用変数	N= 数値	2
H	秒格納用変数	N= 数値	2
I	ミリ秒格納用変数	N= 数値	3

API を呼び出すロジックユニットを定義する

1. タスクのロジックエディタを開き、ユーザイベントに対応したロジックユニットを作成します。
2. アクション処理コマンドで構造体の初期化を行います。

```
BufSetNum ('A'VAR,1,0,2,2)
BufSetNum ('A'VAR,3,0,2,2)
BufSetNum ('A'VAR,5,0,2,2)
.....
BufSetNum ('A'VAR,17,0,2,2)
```

3. コール UDP 処理コマンドを定義し外部 dll を呼び出すようにします。呼び出す関数を定義式で指定します（規約特性は、*STDCALL* に設定します）。

   通常、これらの Dll は、Magic xpa 用にコンパイルされているわけではありません。このため、文字列は以下のように「@」で始まるようにしてください。

```
'@kernel32.GetSystemTime'
```

   これは、kernel32.dll 内の GetSystemTime 関数を呼び出しています（kernel32.dll は、Windows に標準で添付されているモジュールです）。dll ファイル名にパスを指定することもできます。例えば、

```
'@C:¥Windows¥System32¥kernel32.GetSystemTime'
```

   パスが指定されていない場合は、Magic xpa は最初に作業フォルダを探し、存在しない場合は、環境変数の「PATH」で指定されているディレクトリを検索します。

   System32 ディレクトリは、OS で認識できるパスのため、通常指定する必要はありません。

4. コール UDP 処理コマンドの最初のパラメータとして、関数に渡す引数や戻り値を表す文字列を指定します。

   GetSystemTime 関数は、以下のような仕様になっています。

```
VOID GetSystemTime(
  LPSYSTEMTIME lpSystemTime
);
```

   GetSystemTime 関数を呼び出すには、lpSystemTime という SYSTEMTIME 構造体を必要とし、VOID と定義されていて戻り値がないことを示しています。このため戻り値 は、「T0」となります。

注： 関数の説明は、dll のベンダから取得するか、そのソースファイルを見てください。

パラメータのタイプは、以下の文字で表されます。

- 1 …… Char
- W ……Wchar
- 2 …… Short
- 4 …… Long
- F …… Float
- 8 …… Double
- D …… Double pointer
- E …… Float pointer
- L …… Long pointer
- A …… Null で終了する String のアドレス
- U …… LPCWSTR
- V …… Void pointer
- 0 …… Void
- T …… Structure

C 言語での構造体のデータ型と上記のタイプを対応させるには、Windef.h の定義内容を参照してください。

SYSTEMTIME 構造体は次のように定義されています。

```
typedef struct _SYSTEMTIME {
    WORD wYear;
    WORD wMonth;
    WORD wDayOfWeek;
    WORD wDay;
    WORD wHour;
    WORD wMinute;
    WORD wSecond;
    WORD wMilliseconds;
} SYSTEMTIME;
```

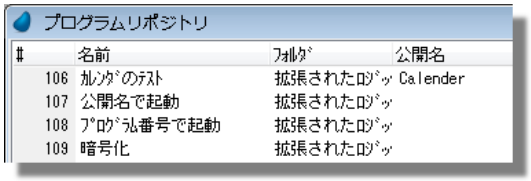
上記のデータ型の "WORD" は、Windef.h において 16 ビット符号なし整数（unsigned short）と定義されています。このため、BufSetNum() および BufGetNum() の記憶形式パラメータは 2（Unsigned Integer）となります。

5. **コール UDP** 処理コマンドの二番目のパラメータとして、GetSystemTime 関数に渡す引数を指定します。この例では、構造体 SYSTEMTIME をもつ BLOB 項目を渡します。
   パラメータの値は、「参照渡し」で渡され、（外部）関数はその内容を必要に応じて更新します。つまりパラメータは以下のようになります。
   - ??? **式** # ‘T0’
   - A 0 （**バッファ用変数**）
6. **コール UDP** 処理コマンドの後に、バッファから値を取り出す処理を定義します。**項目更新**処理コマンドで取得したデータを変数項目に格納します。
   - **年数格納用変数**（B）……BufGetNum ('A'VAR,1,2,2）
   - **月数格納用変数**（C）……BufGetNum ('A'VAR,3,2,2）

   ...
   - **ミリ秒格納用変数**（I）……BufGetNum ('A'VAR,17,2,2）

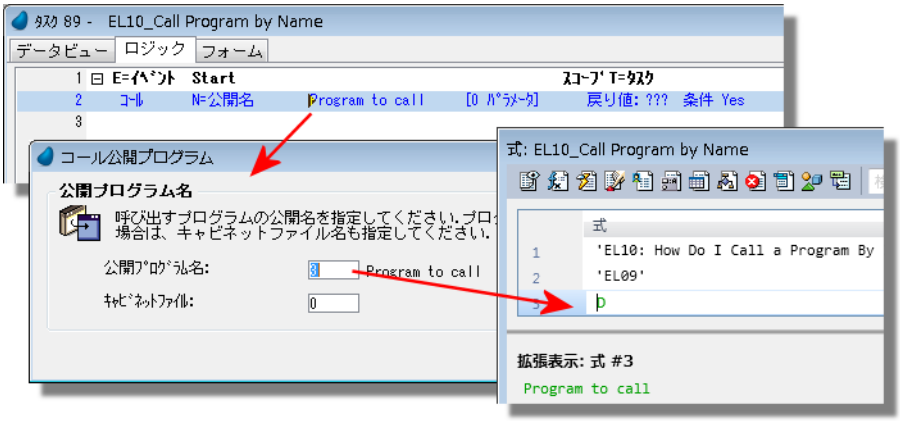
公開名でプログラムを呼び出すには

プログラムに公開名が定義されている場合、この公開名を指定してプログラムを呼び出すことができます。通常プログラムを呼び出す場合は、このような方法は必要ありません。



しかし、データソース内に起動するプログラム名が格納されている場合は便利です。また、実行時にアクセス可能なキャビネットファイル名を指定して呼び出すような場合にも利用できます。

公開名でプログラムを起動する



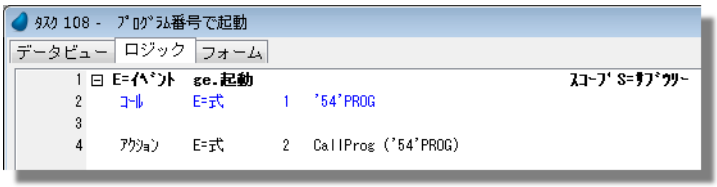
1. *F4* を押下して処理コマンド行を追加します。
2. *C* を押下して**コール** 処理コマンドを選択します。
3. *N* を入力するかプルダウンリストから**コールタイプ**として***公開名***を選択します。次のカラムに移動します。
4. **ズーム**して**コール公開プログラム**ダイアログボックスを開きます。
5. **公開プログラム名**で**ズーム**して**式エディタ**を開き公開名を定義するか実行時に公開名として評価される項目を定義します。
   起動したいプログラムが別のキャビネットファイルに定義されている場合は、**キャビネットファイル**で**ズーム**して同じようにキャビネットファイル名を指定します。

   これで実行時に項目に設定されたプログラム名が呼び出されます。

プログラム番号を指定して動的にプログラムを呼び出すには

プログラム番号を指定して呼び出すことができます。これには２つの方法があります。コール式処理コマンドを使用する方法と **CallProg()** 関数を使用する方法です。

CallProg() 関数は、式を使用してプログラムを呼び出す場合や戻り値を他の関数のパラメータとして使用する場合に便利です。



コール式処理コマンドを使用する

1. *F4* を押下して処理コマンド行を追加します。
2. *C* を押下してコール 処理コマンドを選択します。
3. *E* を入力するかプルダウンリストから**コールタイプ**として***式***を選択します。次のカラムに移動します。
4. ズームして式を定義します。式は、プログラム番号を **PROG** リテラルで指定したものです。例えば、プログラム **#54** を指定する場合、‘54’PROG と入力します。

CallProg() 関数を使用する

1. 式エディタで以下のように関数を入力します。
   CallProg(*'n'PROG*)

   n は起動するプログラム番号です。
   これは、初期値としてフォーム上に表示したり、他の式の一部として使用することができます。プログラムに戻り値がない場合は、アクション処理コマンドでも使用できます。

プログラムの公開名を使用して、式からプログラムを呼び出すために **ProgIdx()** 関数と組み合わせて **CallProg()** 関数を使用することができます。

呼び出すプログラムの名前と番号を保存したい場合は、この方法が便利です。例えば、ユーザが独自のメニューシステムを作成するためのテーブルを作成することができます。

注：　PROG リテラルを使用すると、プログラムリポジトリ内でプログラムが移動してもプログラム番号が同期をとることができます。**CallProg(*54*)** と定義した場合、この関数は実行できますが、プログラム **#54** が **#55** に移動しても、関数は今まで通りプログラム **#54** を呼び出してしまします。

ヒント：この方法は* ProgIdx() *関数と組み合わせ使用する場合に便利です。この関数は、公開名を指定することでプログラム番号が返ります。

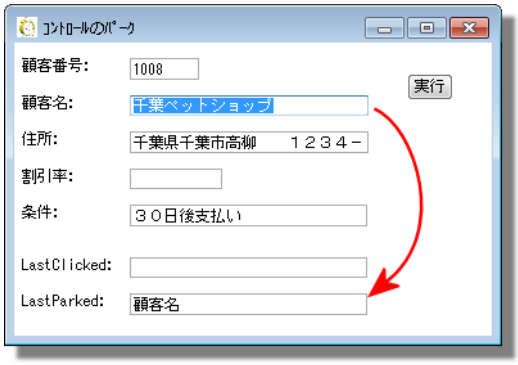
プログラムを設定する

プログラム番号を選択する場合、あらかじめ記憶しておく必要はありません。式に追加する項目を選択するには、式エディタの右側にプログラムリストを表示させます。

上記の例のプログラム番号を選択するには以下のようにします。

1. 式エディタで、**CallProg()** 関数を選択後、ツールバーからプログラムアイコンをクリックするか、右上のドロップダウンリストから ***プログラム***を選択します。式エディタの右側にプログラムリストが表示されます。
2. 必要なプログラムを探します。右側の検索フィールドに移動し、プログラム名の先頭の文字を入力するか、プログラムリストをスクロールすることでプログラムを検索することができます。
3. 該当するプログラム名で***ダブルクリック***すると、プログラム番号が式に反映されます。

ユーザがパークしたコントロール名を取得するには



処理を行う際に、ユーザがどこで最後にパークしたかを知る必要があるかもしれません。**Lastpark()** 関数を使用することで、これを実現することができます。これは、最後にクリックしたコントロールではありません。上の例では、*Tab* によってある入力項目から別の入力項目に移動した場合を示しています。最後にパークしていた項目は顧客名になります。しかし、どの項目もクリックしていないため **LastClicked()** 関数からは空白が返ります。

LastPark() 関数は、コントロールを含んでいる現在のタスクや子タスクからコントロール名を取得します。しかし、以下のような動作になります。

- 現在パークしている項目のコントロール名は返りません。
- 親タスク内のトリガによって実行されたロジックユニットから起動されたタスクでは値が返りません。

HandledCtrl() 関数を使用して現在のタスク、または親タスクからパークしていたコントロール名を取得することができます。現在の項目の情報を取得するには **This()** 関数を使用することができます。

LastPark() 関数を使用する

LastPark() 関数の構文は以下のとおりです。

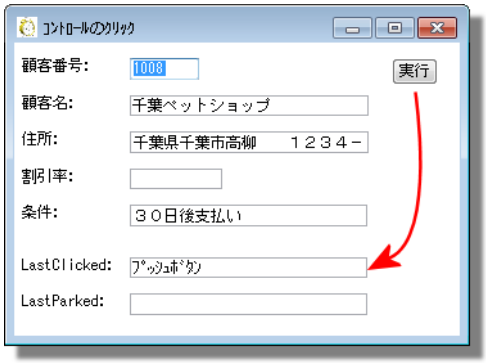
LastPark(*Generation***)**

パラメータ：

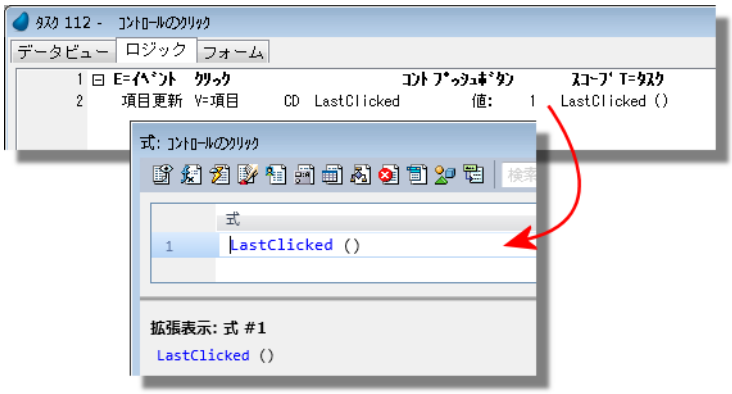
- *Generation* ：タスクツリー上のタスクの位置を示す番号です。**0** は自タスク、**1** は親タスクといったように指定します。

注： 現在のタスクの 1 つの項目に対して何らかの処理を行う場合、そのコントロールに対するためにコントロールイベントを使用することで、これを行うことができます。

ユーザがクリックしたコントロール名を取得するには



処理を行う際に、ユーザどこで最後にクリックしたかを知る必要があるかもしれません。プッシュボタンのようにカーソルがパークできないコントロールも含まれるため、最後にパークした場所とは異なります。



LastClicked() 関数を使用する

必要条件： コントロールには、すでにコントロール名が定義されているものとします。定義されていない場合は何も返りません。

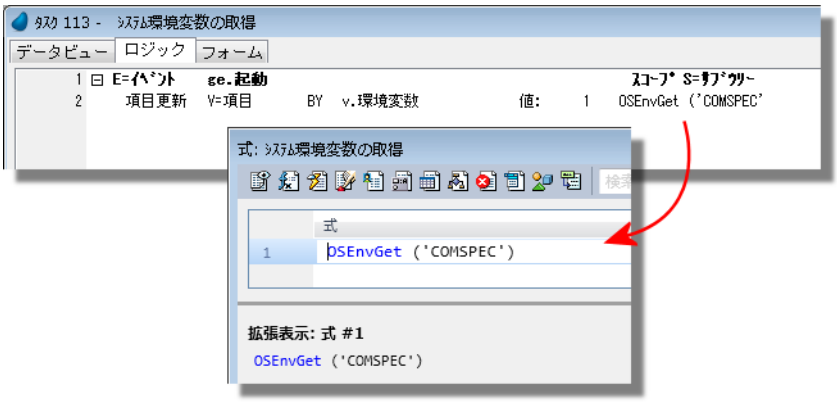
LastClicked() 関数を使用することでユーザが最後にクリックしたコントロール名を取得します。パラメータは指定しません。戻り値として、コントロール名を表す文字列が返ります。この値は、コントロール名を確認するために、他の関数に渡したり、式で使用することができます。

注： ユーザがクリックしたコントロール名にもとづいた処理を実行する場合、クリックイベントのロジックユニットに対してフィールド（コントロール）を設定することでも同じようなことができます。例えば、上記の例において、イベントロジックユニットを定義した場合は、以下のように定義します。

設定されたコントロールがクリックされた場合だけ、このロジックユニットは実行されます。

システムの環境変数から値を取得するには

OS の環境変数には、システムのユーザ ID、OS のタイプ、バージョン、コンピュータ名、およびコマンドシェルのパスなどのような利用可能な多くの情報があります。**OSEnvGet()** 関数を使用して環境変数を取り出すことにより Magic プログラムでこれらを利用することができます。



OSEnvGet 関数を使用する

OSEnvGet 関数の構文は以下の通りです。

OSEnvGet(*Variable*)

パラメータ：

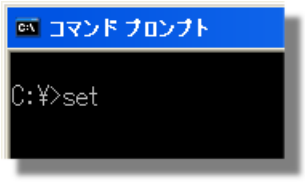
- *Variable*：環境変数の名前を表すテキストです。

この例では、コマンドシェルへのパスを格納するため、**項目更新**処理コマンドを使用しています。使用すると便利な環境変数名には以下のようなものがあります。

- COMSPEC …… シェルのパス名
- OS …… OS の種類
- USERNAME …… 現在のユーザのユーザ名
- USERDOMAIN …… 現在のユーザのログオンドメイン
- LOGONSERVER …… ドメインコントローラにログオンした時に表示される名前

しかし、ユーザ独自の環境変数を定義することもできます。ユーザ独自に環境変数を定義することで、便利なこともあります。

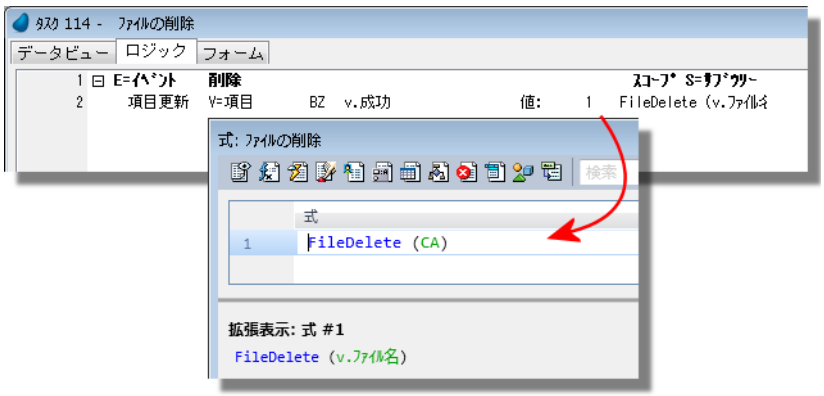
ヒント :*現在の環境変数を確認するには、コマンドプロンプトを起動し、SET コマンドを実行することで表示されます。*



ディスク上のファイルを削除するには

ディスク上のファイルを削除したい場合が考えられます。例えば、一時ファイルを作成した後に、一時ディレクトリからこのファイルを消去させたい場合があります。またファイルを再作成する場合、処理を実行する前に古いファイルを削除させる必要があります。

OS コマンドを使用するよりも Magic xpa の関数を使用した方がより便利です。これは、OS コマンドを使用すると、OS のバージョンに依存する場合があるからです。Magic xpa の関数を利用した方がより確実です。また、Magic xpa の関数は、戻り値を簡単に確認することができます。



FileDelete() 関数を使用する

FileDelete() 関数を使用することでディスク上のファイルを削除することができます。構文は以下の通りです。

FileDelete(*FileSpec*)

パラメータ：

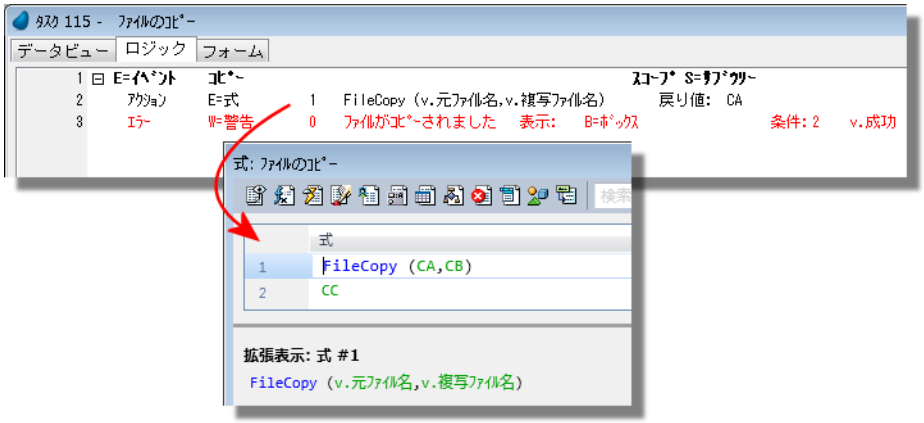
- *FileSpec*：削除するファイル名を表す文字列です。削除が成功した場合、True が返ります。

注： 削除ができなかったり、ファイルが存在しない場合は、False が返ります。しかし **FileExist()** 関数と組み合わせて使用することで、ファイルが削除されたかどうかを確認できるためより確実になります。

参照： 「FileExist() 関数を使用する」（166 ページ）

ディスク上のファイルをコピーするには

ディスク上のファイルのコピーしたい場合があります。例えば、バックアップや保存のためのコピーを作成するような場合が考えられます。



OS コマンドを使用するよりも Magic xpa の関数を使用した方がより便利です。これは、OS コマンドを使用すると、OS のバージョンに依存する場合があるからです。Magic xpa の関数を利用した方がより確実です。また、Magic xpa の関数は、戻り値を簡単に確認することができます。

FileCopy() 関数を使用する

FileCopy() 関数を使用してディスク上のファイルをコピーすることができます。構文は以下の通りです。

FileCopy(*origin*, *target***)**

パラメータ：

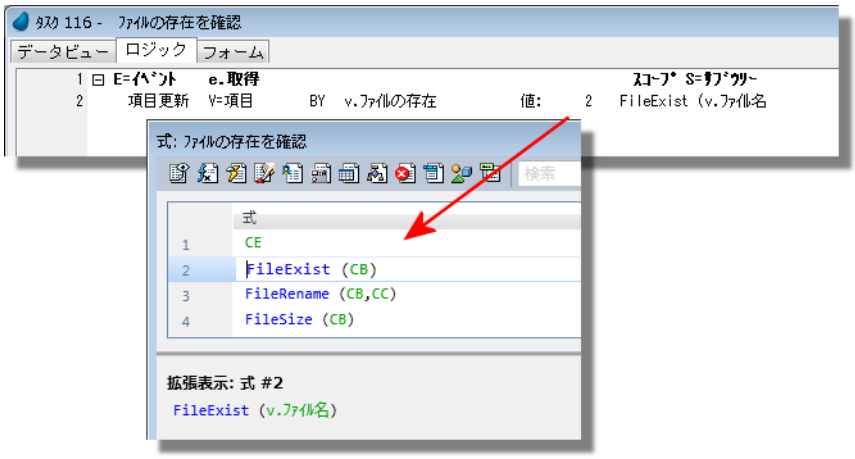
- *origin*：コピー元のファイルを表す文字列です。
- *target*：コピー先のファイルを表す文字列です。コピーが成功した場合、True が返ります。

注： **FileCopy()** 関数は、同じ名前のファイルが存在している場合上書きします。このような場合に警告メッセージを表示させるには、事前に **FileExist()** 関数を実行してチェックする必要があります。

参照： 「FileExist() 関数を使用する」（166 ページ）

ディスク上のファイルの存在をチェックするには

必要な共通関数の 1 つに、ディスク上にファイルが存在しているかどうかをチェックする関数があります。例えば、ユーザが Excel からデータを読み捨てている場合、データが見つからなかったり、処理が何らかの理由で成功しなかった場合、メッセージを表示させたい場合などが考えられます。



OS コマンドを使用するよりも Magic xpa の関数を使用した方がより便利です。これは、OS コマンドを使用すると、OS のバージョンに依存する場合があるからです。Magic xpa の関数を利用した方がより確実です。また、Magic xpa の関数は、戻り値を簡単に確認することができます。

FileExist() 関数を使用する

FileExist() 関数を使用してディスク上のファイルをチェックすることができます。構文は以下の通りです。

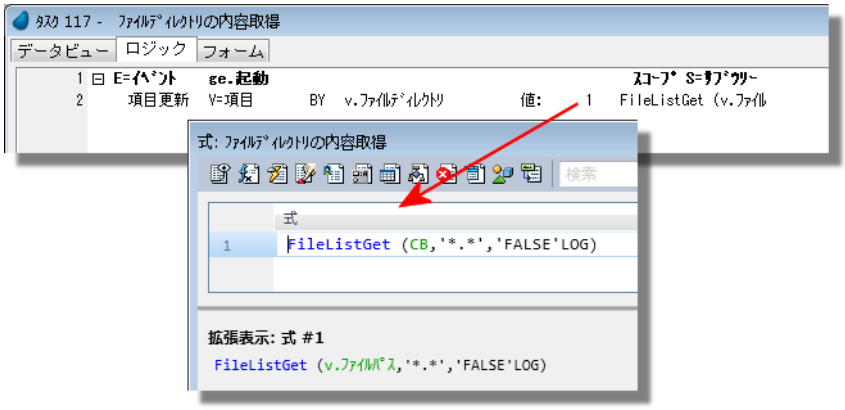
FileExist(*FileSpec*)

パラメータ：

- *FileSpec*：チェックするファイル名を表す文字列です。ファイルが見つかった場合は、True が返ります。

参照： 『リファレンスヘルプ』の入出力関数に関するトピック

ファイルディレクトリの内容を取得するには



プログラムでファイルディレクトリの内容を読み込む必要があるかもしれません。別のプログラムから、現在のプログラムで処理するためにファイルをドロップした場合（例えば、電子メールや Fax 用プログラム、または Web から送られたファイルなど）、その内容を確認する必要があります。

FileListGet() 関数を使用する

FileListGet() 関数を使用してディレクトリの内容を取得することができます。構文は以下の通りです。

FileListGet(*DirectoryName*, *Filter*, *SubDirSearch***)**

パラメータ：

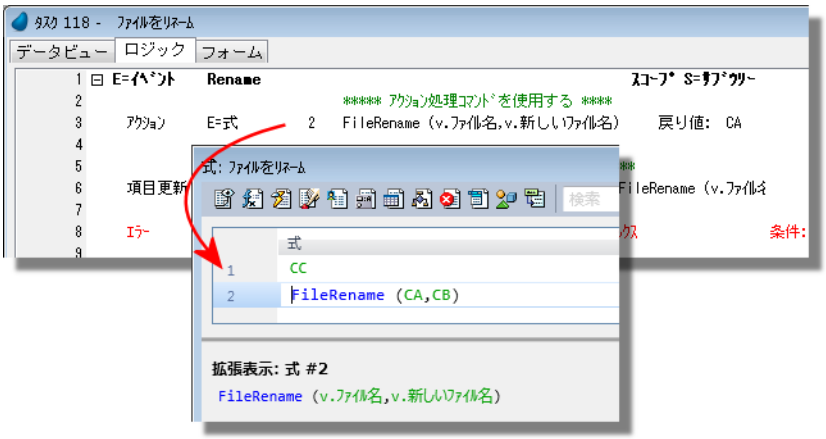
- *DirectoryName* ：検索するディレクトリ名を表す文字列
- *Filter*：フィルタ用マスク文字
- *SubDirSearch*：サブディレクトリも検索するかどうかを指定する論理値

この関数は、文字列をセルとしたベクトル値が返ります。

参照： 第 14 章：「COM オブジェクトに配列を渡す」（349 ページ）
『リファレンスヘルプ』の入出力関数に関するトピック

ディスク上のファイルをリネームするには

ディスク上のファイルの名前を変更する必要があるかもしれません。例えば、新しファイルを作成する前に既存のファイルをバックアップのためにリネームする場合が考えられます。



OS コマンドを使用するよりも Magic xpa の関数を使用した方がより便利です。OS コマンドは、OS のバージョンに依存する場合があるからです。Magic xpa の関数を利用した方がより確実です。また、Magic xpa の関数は、戻り値を簡単に確認することができます。

FileRename() 関数を使用する

FileRename() 関数を使用してファイルの名前を変更することができます。構文は以下の通りです。

FileRename (*origin*, *target*)

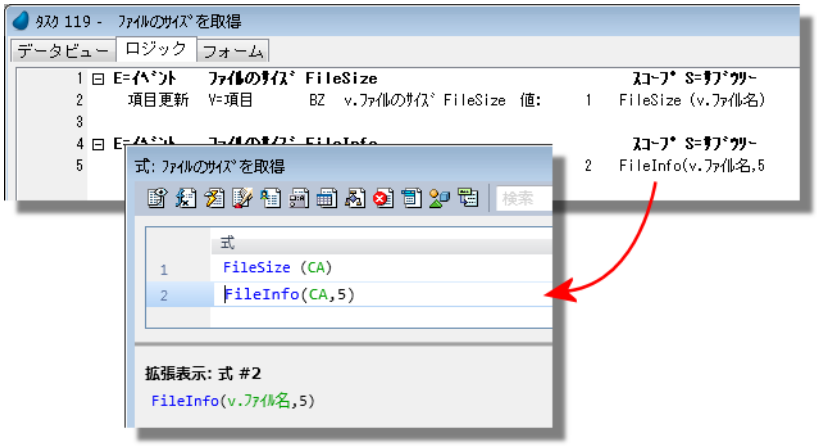
パラメータ：

- *origin*：元のファイル名を表す文字列です。
- *target*：変更後のファイル名を表す文字列です。リネームが成功した場合、True が返ります。

上の図では、２つの例が示されています。最初は７行目です。**アクション**処理コマンドの**戻り値**カラムを使用して戻り値を取得しています。２番目は、１０行目です。**項目更新**処理コマンドを使用して同じ処理を定義しています。どちらも正しく動作します。

参照： 『リファレンスヘルプ』の入出力関数に関するトピック

ディスク上のファイルのサイズを取得するには



ディスク上のファイルのサイズを知る必要があるかもしれません。データを作成したり、電子メールに添付する際に大きすぎないかどうかをチェックする場合などが考えられます。

FileInfo() 関数を使用する

FileInfo() 関数を使用してファイルのサイズを取得することができます。構文は以下の通りです。

FileInfo (*FileSpec,InfoType*)

パラメータ：

- *FileSpec*：チェックするファイル名を表す文字列です。
- *InfoType*：チェックする情報のタイプを数値で指定します。ファイルサイズの場合は、5 を指定します。

参照： 『リファレンスヘルプ』の入出力関数に関するトピック

他の Windows アプリケーションにフォーカスを移すには

他のアプリケーションと連携したシステムを運用する場合、そのアプリケーションにフォーカスを移す必要があるかもしれません。

SetWondowFocus() 関数を使用する

SetWindowFocus() 関数を使用することでウィンドウ名を指定したアプリケーションにフォーカスを移すことができます。構文は以下の通りです。

SetWindowFocus (*Window Name*)

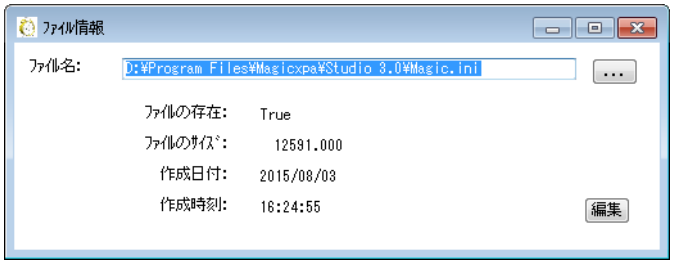
パラメータ：

- ***Window Name***：フォーカスを移したいアプリケーションのウィンドウ名

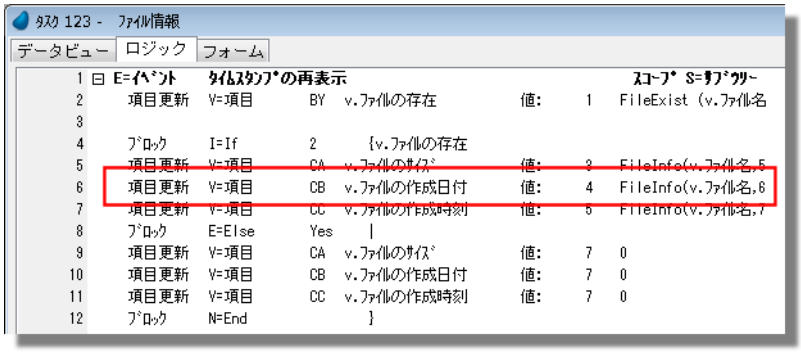
例えば、Sample.doc という文書ファイルを開いた Micorsoft Word のウィンドウ名は、「Sample.doc - Microsoft Word」になります。ウィンドウ名は、アプリケーションウィンドウの左上に表示されています。また、タスクバーや Windows タスクマネージャでも確認できます。

参照： 『リファレンスヘルプ』の入出力関数に関するトピック

ディスク内のファイルの作成日付を取得するには



ディスクファイルにアクセスする場合、ファイルに関する情報（例えばファイルの作成日）を取得すると便利な場合があります。このような情報は、**FileInfo()** 関数を使用することで取得できます。



1. 作成日を取得したいファイルを決めます。この例では、ファイル名を変更したり、内容を編集した場合に作成日付が更新されます。
2. **FileInfo()** 関数を使用して作成日付を取得します。この関数は、以下の 2 つのパラメータを指定します。
   - ファイル名……ファイル名には、（例えば %Temp% のような）論理名や相対パスを含めることができます。
   - 数値 (6)……作成日を取得することを指定する番号。これ以外の数値は、作成時刻や、更新日／時刻等を取得するための番号が指定できます。パラメータの詳細は、Magic xpa のリファレンスをヘルプを参照してください。

［このページは意図的に空白にしています。］

第 7 章： 実行

アプリケーションを実行するには

開発しているプロジェクトが完成した場合、どのようにアプリケーションを実行させるかを検討することになります。これは使用するエンドユーザに依存する場合があります。アプリケーション毎にユニークなアイコンとアプリケーション名を定義する必要があります。また、アプリケーションが売れ筋の製品の場合、スプラッシュスクリーンを表示させたり、独自のインストール処理を必要とすることになるかもしれません。

以下は、アプリケーションの実行に関係する手順と、その詳細情報が記述されている箇所について記述しています。

#	内容	記述箇所
1	**キャビネットファイルの作成** キャビネットファイルは修正できないプロジェクトの内容がパッケージ化されたファイルです。	「キャビネットファイルを作成するには」（174 ページ）
2	**クライアントに Magic xpa Client をインストール** Magicxpa Client は、アプリケーションを実行するための製品です。	『インストールガイド』
3	**エンドユーザ用のショートカットやメニューの作成** エンドユーザがアプリケーションを実行しやすくします。アプリケーションによっていろいろな方法があります。	「アプリケーション用のショートカットを作成するには」（175 ページ）
4	**自動起動環境の作成** アプリケーションを自動的に起動するための環境設定を行います。	開始アプリケーションは Magic.ini ファイルで指定できます（「動作環境でデフォルトアプリケーションを設定する」（177 ページ）を参照）。複数のアプリケーションに対して、1 つのデフォルトアプリケーションだけを実行させたい場合は、この設定が必要となります。 しかし、異なるアプリケーションを異なる時間で開始させたい場合は、アプリケーションファイルに対するショートカットを作成することで、直接アプリケーションを開始させることができます（「アプリケーション用のショートカットを作成するには」（175 ページ）を参照）。

キャビネットファイルを作成するには

Magic xpa Studio で作業を行っている場合、Magic xpa の**プロジェクトファイル**（***.edp**）にアクセスしていることになります。この **.edp** はソースファイルのコレクションを参照しています。これは一般的に XML フォーマットので、¥Source サブディレクトリ内に保存されています。

しかし、プロジェクトが完成した場合、インストール用にそれをパッケージ化することになります。パッケージ化されたファイルは、**キャビネットファイル**（***.ecf**）と呼ばれます。

Magic xpa のインストールによって、**.edp** と **.ecf** のどちらのファイルも**クリック**することで実行することができるように設定されます。これらのファイルの違いは以下の通りです。

.edp	.ecf
eDeveloper Project File	eDeveloper Cabinet File
Magic xpa Studio でオープンします。	アプリケーションを実行します。
開発者のみ	主にユーザ、または開発時にコンポーネントとして使用されます。
XML フォーマットのソースファイルが必要	XML ソースがパッケージ化されています。

プロジェクトを実行する場合、**.ecf** ファイルとその他のサポートファイル（フォントや基本色などの定義ファイル、コンポーネント）のみ必要です。

また、実行するには、ユーザは Magic xpa の実行版（**Magic xpa Enterprise Client** や **Magic xpa Enterprise Server**、**Magic xpa RIA Server**）がインストールされており、実行できる状態になっていることが必要です。

キャビネットファイルの作成は非常に簡単です。

キャビネットファイルを作成する

1. プロジェクトを開きます（**ファイル→プロジェクトを開く**）。
2. プルダウンメニューから**ファイル→キャビネット作成**を選択します。
3. キャビネットファイル名の入力要求が表示されます。ファイル名を入力するか、ファイル名を選択し保存を**クリック**します。
4. ファイルがすでに存在している場合、**上書き確認**のダイアログボックスが表示されます。上書きする場合は、はいを**クリック**します。

ダイアログボックスが閉じると、**.ecf** ファイルは作成（更新）され、使用できるようになります。

参照： 「アプリケーション用のショートカットを作成するには」（175 ページ）

アプリケーション用のショートカットを作成するには

アプリケーションを実行する最も一般的な方法は、Windows のショートカットをクリックして起動する方法です。これらのショートカットは、簡単に作成できます。ショートカットは、実行するプロジェクト（**.edp**）ファイルに対して作成するか、**Magic xpa Studio** を起動するショートカットにデフォルトプロジェクトを指定するようにする作成する方法の２通りあります。

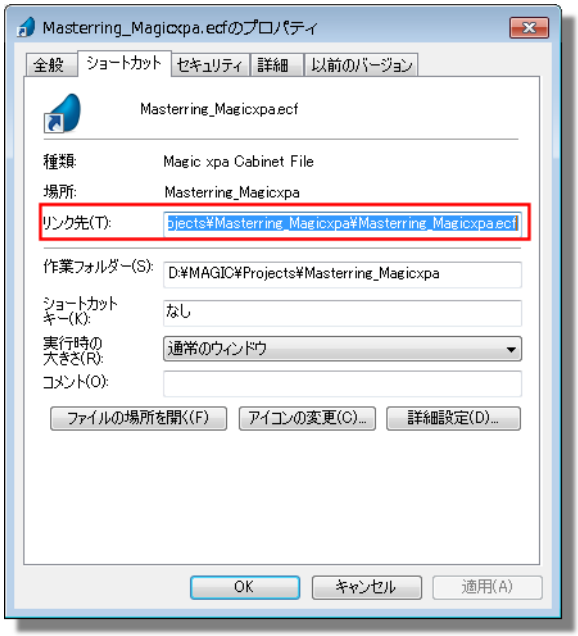
注： ショートカットの作成は必須ではありません。ユーザが Windows エクスプローラを使用して **.ecf** ファイルをクリックすることで、Magic xpa が自動的に起動されます。

アプリケーションファイルのショートカットを作成する

1. ショートカットを作成したいフォルダ内で、コンテキストメニューを表示させ新規作成→ショートカットを選択します。ショートカット作成ウィザードが起動されます。
2. 項目の場所を入力する要求が表示されます。ここでキャビネットファイルのファイル名とパスを入力します。参照ボタンをクリックして選択することもできます。次へをクリックします。
3. ショートカットの名前を入力する要求が表示されます。ここに任意の名前を入力します。
4. 完了をクリックしてダイアログを閉じます。

これでアプリケーションを起動するためのショートカットが作成されます。デフォルトでは、Magic xpa のアイコンが表示されますが、独自なアイコンを定義することもできます。「独自のアイコンを使用する」（175 ページ）

ショートカットのプロパティを開くと、右の例ように表示されます。**.ecf** は、リンクに設定されています。



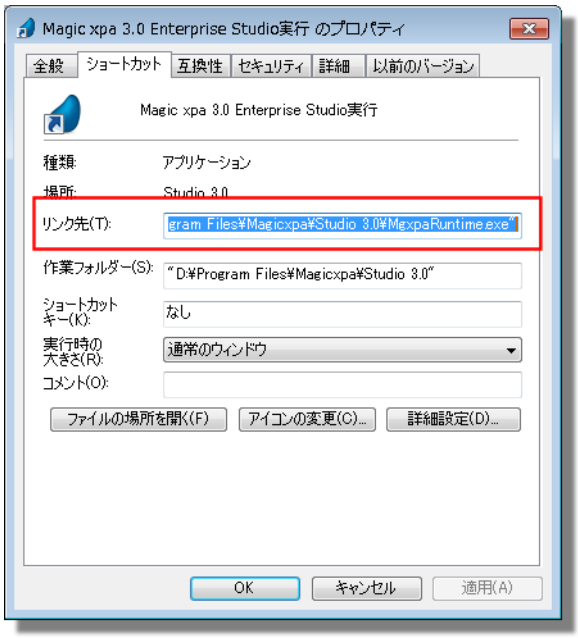
Magic エンジンのショートカットを作成する

1. ショートカットを作成したいフォルダ内で、コンテキストメニューを表示させ新規作成→ショートカットを選択します。ショートカット作成ウィザードが起動されます。
2. 項目の場所を入力する要求が表示されます。ここで Magic xpa の実行モジュールのファイル名とパスを入力します。デフォルトは以下のようになります。

   ```
   "C:¥Program
   Files¥Magic_xpa¥Studio¥MgxpaRuntime.exe"
   ```

3. ショートカットの名前を入力する要求が表示されます。ここに任意の名前を入力します。
4. 完了をクリックしてダイアログを閉じます。

ここでは Magic xpa の実行エンジンを起動するだけの設定を説明しています。アプリケーションを自動的に起動させる場合は、「実行エンジンがキャビネットファイルを自動的に読み込むように設定するには」（177 ページ）を参照してください。



独自のアイコンを使用する

上記のどちらの方法でショートカットを作成しても、デフォルトアイコンは Magic xpa アイコンになります。しかし、独自のカスタムアイコンを使用することもできます。

必要条件： 最初にカスタムアイコン用のファイルを作成する必要があります。

1. ショートカットのコンテキストメニューからからプロパティを選択します。

2. **アイコンの変更**ボタンを**クリック**します。**アイコンの変更**ダイアログが表示されます。最上段の入力欄には、「このファイル内のアイコンを検索」と表示されています。.
3. **参照**ボタンを**クリック**して使用するアイコンファイルを選択します。選択すると、表示されていたアイコンがデフォルト Magic xpa アイコンから選択されたアイコン表示に変わります。

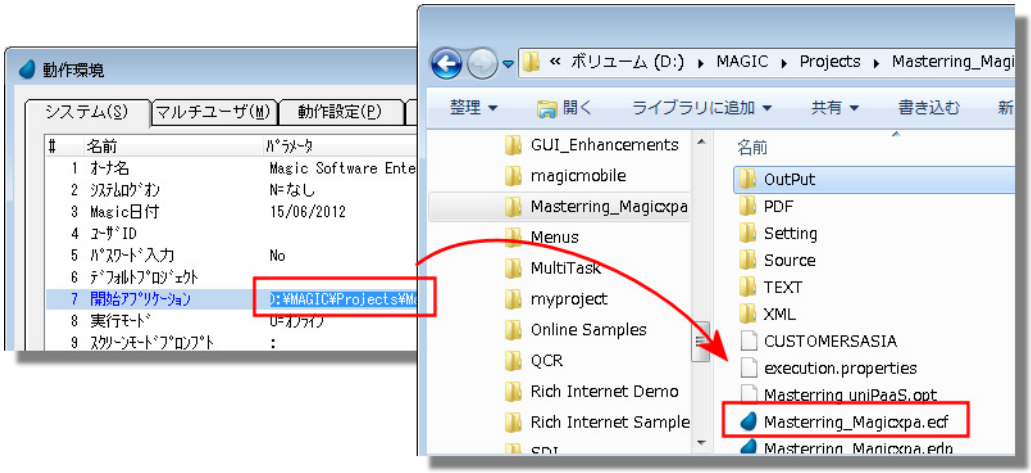
カスタムアイコン用のファイルがない場合は、Windows が持っているアイコンから選択したり、汎用のアイコンファイルを購入して使用することもできます。

実行エンジンがキャビネットファイルを自動的に読み込むように設定するには

動作環境に指定することで、実行エンジンが起動時に自動的にキャビネットファイルを読み込むようにすることができます。ここにキャビネットファイルを指定すると、Magic xpa が起動時にデフォルトで指定されたプロジェクトが実行されます。

２つのデフォルトが存在するので注意してください。１つはデフォルトのプロジェクトです。**Magic xpa Studio** が起動されたときに自動的にオープンされるプロジェクトファイル（***.edp**）です。もう１つは開始アプリケーションです。これは実行エンジンが起動時に読み込まれるキャビネットファイル（***ecf**）です。

動作環境でデフォルトアプリケーションを設定する

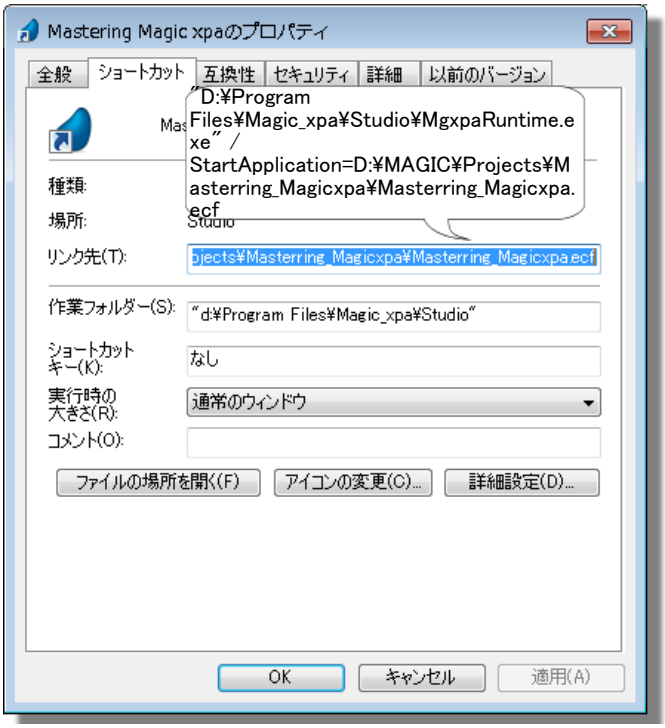


1. プロジェクトを開きます（ファイル→プロジェクトを開く）。
2. **動作環境**ダイアログを開き（オプション→設定→動作環境→システム）、**開始アプリケーション**を選択します。
3. ここからズーム（*F5*、または編集→ズーム）して**プロジェクトを選択**ダイアログを開きます。ここで **.ecf** ファイルを選択します。直接ファイル名を入力することもできます。

これで、実行エンジンが起動すると指定されたキャビネットファイルがオープンされます。

注： 設定された**開始アプリケーション**は Magic.ini ファイルに保存されます（Magic.ini ファイルは、デフォルトでは Magic xpa のインストールディレクトリ内にあります）。自動的に起動させたいアプリケーションが 1 つしかない場合は、この設定で十分ですが、異なる複数のアプリケーションも同じように起動させたいような場合は、ショートカットを作成して対応する必要があります。詳細は、「アプリケーションファイルのショートカットを作成する」（175 ページ）を参照してください。

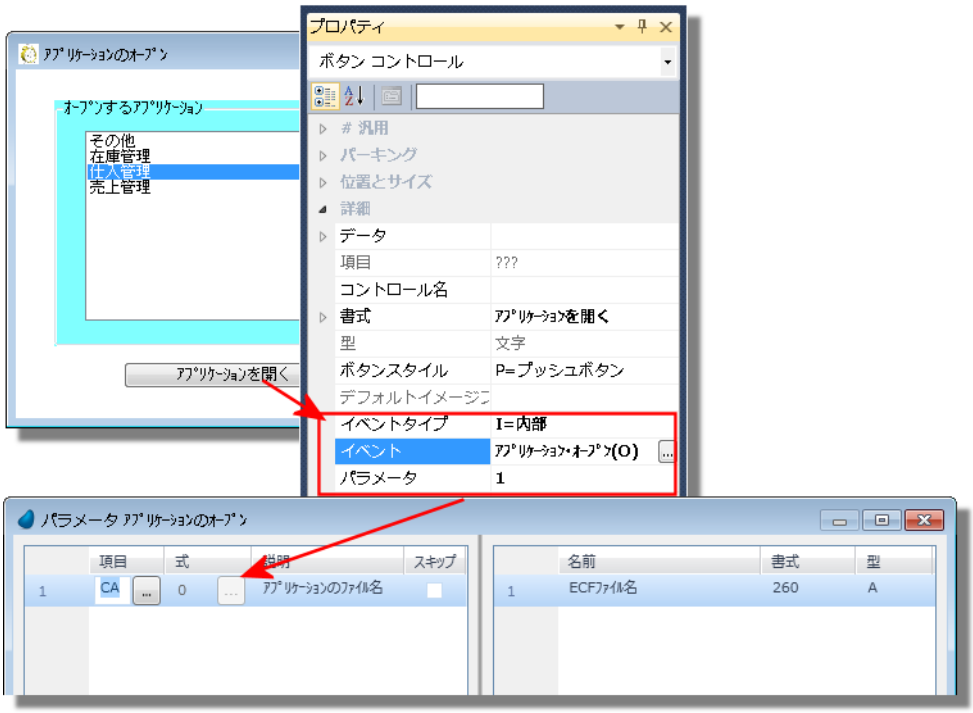
ショートカットにキャビネットファイルを設定する



Windows のショートカットの**リンク先**にキャビネットファイルを指定することもできます。

1. **リンク先**で Magic 実行エンジンのパスとファイル名を入力します（パス名にスペースがある場合は、ダブルクォーテーションで囲む必要があります）。
2. 空白とパラメータを指定するためのスラッシュ（/）、パラメータ名（**StartApplication=**）を付加します。
3. キャビネットファイルのパスとファイル名を入力します。

アプリケーションリストのウインドウや他のアプリケーションを開くには



異なるモジュールを異なる顧客に販売できるようにするたや、扱いやすさ、メンテナンス上の理由で、大きな Magic アプリケーションを、分割する場合があります。このような場合、エンドユーザがどのアプリケーションを実行するべきかを選択できるように、単純なアプリケーションリストを用意しておくと便利です。

これを実現するための簡単な方法は、有効なアプリケーションのリストを作成して、ユーザがどちらを実行するべきかを選択できるようにすることです。この例では、アプリケーション・リストはデータベーステーブルに保持され、リストボックスで表示されます。実際の ECF の格納場所は、エンドユーザには表示されません、しかし、テーブルには保存されています。

これで、アプリケーションを開ける場合、アプリケーションオープンの内部のイベントが発行されます。イベントは、プッシュボタンから直接発行されますが、これはイベントロジックユニットで使用することもできます。パラメータは、単に ECF のパスを渡すだけです。ユーザがボタンをクリックすると、選択された ECF が開きます。

新しいアプリケーションを開くと、それが唯一の実行アプリケーションとなります。つまり、ユーザが新しいアプリケーションを終了すると、全てのセッションが終了します。アプリケーションリストが、全ての有効なアプリケーションに対してドライバーとして動作することを望む場合は、タスク後で処理コマンドを定義することで、アプリケーションの終了時に制御をアプリケーションリストに返すようにする必要があります。または、アプリケーションを開くのメニューを追加することでユーザが簡単にアプリケーションを切り替えることができます。

注： **このイベントは、Magic xpa の実行エンジンを使用した場合のみ動作します。つまり、ECF から実行する必要があります。また、リッチクライアントプログラムでは利用できません。**

インストーラを使用しないで Web アプリケーションを配備するには

ここでは、Magic xpa のインストーラを使用することなく、複数の PC に他の PC からコピーした Magic xpa バージョンのフォルダを使って Web Client/ リッチクライアントの実行環境を構築する場合に役立ちます。

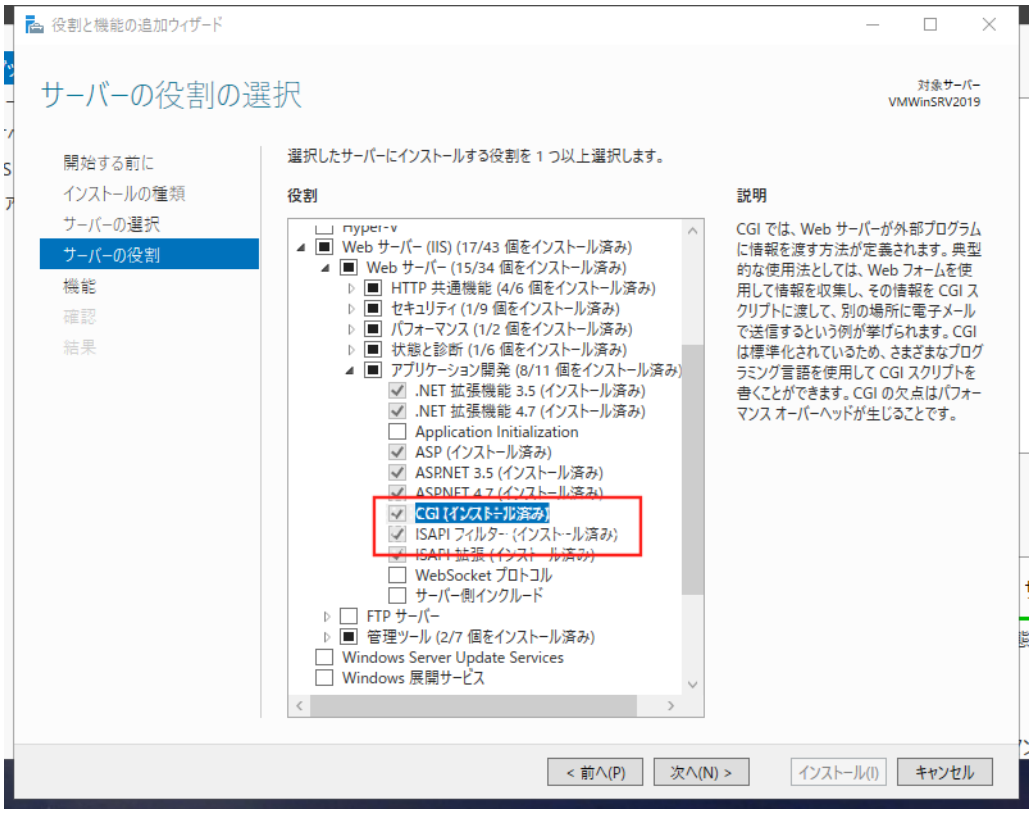
以下のことができます。

- iis_web_client.bat（Web Client アプリケーション用）または iis_web_client.bat (Web Client アプリケーション用) を実行して、コピーした Magic xpa のバージョンに対応したエイリアスを作成します。
- BrokerSettings.exe を使用して Broker サービスを作成します。

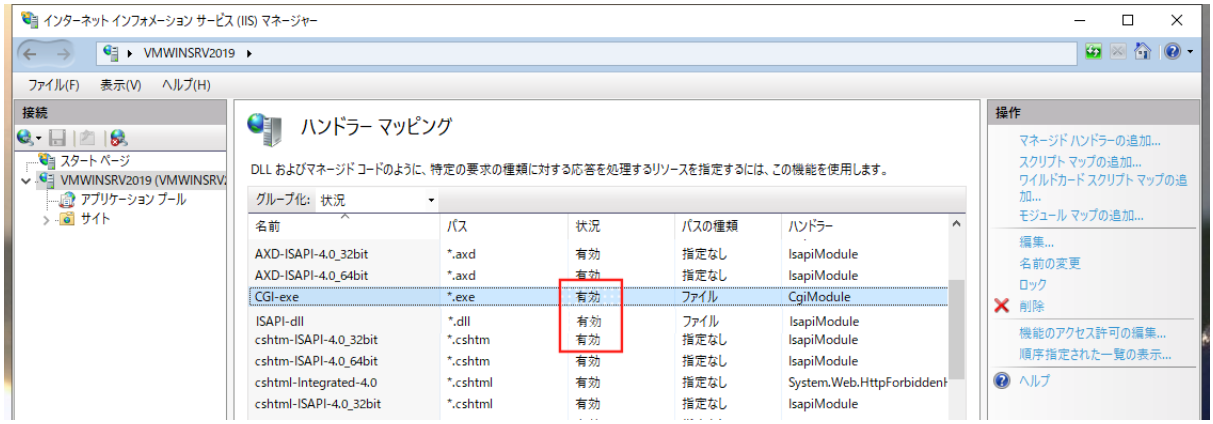
まず、以下の Web サーバ関連の前提条件がインストールされていることを確認してください。

Web サーバ関連の前提条件

1. インターネットインフォメーションサービスや World Wide Web サービス、アプリケーション開発機能の一部として、CGI および ISAPI 機能がインストールされていることを確認してください（下図参照）。



2. 選択したコンポーネントをインストールして、ウィザードをします。
3. ［スタート］メニューから［インターネットインフォメーションサービス（IIS）マネージャ］を開きます。
4. ［ハンドラマッピング機能］を開きます。
5. 以下の 2 つのエントリを有効にします。



Web Client アプリケーションの配置環境の作成

Web Client の導入環境を新規に作成するには、以下の手順に従ってください。

前提条件

- マシン A …… このマシンに新しい展開環境を作成します。新しい配置環境は、このマシンに他の作業用 Studio がインストールされているかどうかに関係なく動作します。そのようなインストールがあったとしても、その設定は影響を受けません。
- マシン B …… Magic xpa のインストーラを使用して、このマシンに配備するサーバ製品をインストールします。このインストールの内容をマシン A にコピーします。

マシン A で行うべき変更:

1. マシン B の Magic xpa のインストールフォルダをマシン A の任意のフォルダにコピーします。例えば、c:Magic\Server とします。
2. 以下のファイルで Server フォルダ内の MRB のポートを変更します。

- Magic.ini …… [MAGIC_SERVERS] セクションの [Default Broker]
- mgrb.ini …… [Broker Port] セクションのエントリ
- Scripts\mgreq.ini …… [MessagingServer] セクションのエントリ。セミコロンが入っている場合は削除してください。
- EngineDir 内の mgreq.ini …… [MessagingServer] セクションのエントリ。セミコロンが入っている場合は削除してください。

たとえば、デフォルトのポート "6115" を新しいポート "6200" に変更します。

3. mgrb.ini の [APPLICATIONS_LIST] セクションにアプリケーションエントリを追加します。.

例えば：

4. Scripts フォルダ内の BrokerSettings.exe を使用して、手順 2 で指定した新しいポートで動作する Broker Service を作成します。BrokerSettings.exe の構文と使用例は以下のとおりです。

```
BrokerSettings.exe <MRB の実行パス> <MRB.INI のパス> <サービス名>
```

例えば：

```
BrokerSettings.exe "c:¥Magic¥Server¥MgBroker.exe" "c:¥Magic¥Server¥mgrb.ini" "Deployment
Broker"
```

注： Broker Service が作成されていることを確認してください。

5. Web サーバを設定するには、Scripts フォルダ内の iis_web_client.bat の以下の引数の値を変更します。

```
SET ApplicationPoolName=MagicXpaAppPool 4.8
SET ScriptsDirectory=c:¥magic¥¥server¥Scripts
SET ScriptsAliasName=Magic48Scripts
SET WebClientCacheDirectory=c:¥magic¥¥server¥Web_Client_Cache
SET WebClientCacheAliasName=Magic48WebCCache
SET IsxpaServer32bit=false
```

6. 手順 5 で修正した iis_web_client.bat を管理者権限で実行します。

注： コマンドが「エラー」ではなく「成功」の表示を返すことを確認してください。

7. Magic.ini のプロパティを以下のように変更してください（c:Magic\\Server フォルダ）。

```
WebClientCachePath = c:¥Magic¥Server¥Web_Client_Cache
WebClientCacheAlias = /Magic48WebCCache
WebDocumentPath = c:¥Magic¥Server¥Scripts
InternetDispatcherPath = /Magic48Scripts/MGrqispi.dll
WebDocumentAlias = /Magic48Scripts
```

8. Angular アプリケーションの構築については、ドキュメント「Deploying a Web Client Application.Pdf」の説明を参照してください。
9. 生成されたアプリケーションの配布フォルダにある server-config.json のエイリアスを変更してください。

これで、マシン A に Web Client アプリケーションの配備環境が整いました。

リッチクライアントアプリケーションの配置環境の作成

リッチクライアントの導入環境を新規に作成するには、以下の手順で行います。

前提条件

- マシン A …… このマシンに新しい展開環境を作成します。新しい配置環境は、このマシンに他の作業用 Studio がインストールされているかどうかに関係なく動作します。そのようなインストールがあったとしても、その設定は影響を受けません。
- マシン B …… Magic xpa のインストーラを使用して、このマシンに配備するサーバ製品をインストールします。このインストールの内容をマシン A にコピーします。

マシン A で行うべき変更

1. マシン B の 配備用の Server フォルダをマシン A の任意のフォルダにコピーします。例えば、c:Magic\Server とします。
2. 以下のファイルで Server フォルダ内の MRB のポートを変更します。

- Magic.ini …… [MAGIC_SERVERS] セクションの [Default Broker]
- mgrb.ini …… [Broker Port] セクションのエントリ
- Scripts\mgreq.ini …… [MessagingServer] セクションのエントリ。セミコロンが入っている場合は削除してください。
- EngineDir 内の mgreq.ini …… [MessagingServer] セクションのエントリ。セミコロンが入っている場合は削除してください。

たとえば、デフォルトのポート "6115" を新しいポート "6200" に変更します。

3. mgrb.ini に [APPLICATIONS_LIST] セクションにアプリケーションエントリを追加します。.

例えば：

```
Web48 = MgxpaRuntime.exe /DeploymentMode=B /LicenseName=MGRIA14 /
StartApplication=d:¥mse¥xpa48¥Projects¥Web48¥Web48.ecf,c:¥Magic¥Server,,,1,1
```

4. Scripts フォルダ内の BrokerSettings.exe を使用して、手順 2 で指定した新しいポートで動作する Broker Service を作成します。BrokerSettings.exe の実行時のパラメータ構文と使用例は以下のとおりです。

```
BrokerSettings.exe <MRB の起動パス> <MRB.INI のパス> <サービス名>
```

例えば：

```
BrokerSettings.exe "c:¥Magic¥Server¥MgBroker.exe" "c:¥Magic¥Server¥mgrb.ini" "Deployment
Broker"
```

注： Broker サービスが作成されていることを確認してください。

5. Web サーバを設定するには、Scripts フォルダ内の iis_rich_client.bat の以下の引数の値を変更します。

```
SET ApplicationPoolName=MagicXpaAppPool 4.8
SET ScriptsDirectory=c:¥magic¥server¥Scripts
SET ScriptsAliasName=Magic48Scripts
SET publishedRIAAppsDirectory=c:¥magic¥server¥PublishedApplications
SET PublishedRIAAppsAliasName=Magic48RIAApplications
SET IsxpaServer32bit=false
```

6. 手順 5 で修正した iis_rich_client.bat を実行します。

注： コマンドが「エラー」ではなく「成功」の表示を返すことを確認してください。.

7. Magic.ini のプロパティを以下のように変更してください。

```
RIACacheFilesPath = c:¥Magic¥Server¥RIACache
RIAApplicationsAlias = Magic48RIAApplications
WebDocumentPath = c:¥Magic¥Server¥Scripts
InternetDispatcherPath = /Magic48Scripts/MGrqispi.dll
WebDocumentAlias = /Magic48Scripts
```

8. 開発サーバから配備サーバに PublishApplications フォルダをコピーしてエイリアスを変更します。

これで、マシン A にリッチクライアントアプリケーションのデプロイメント環境が整いました。

サポートバージョン：4.8

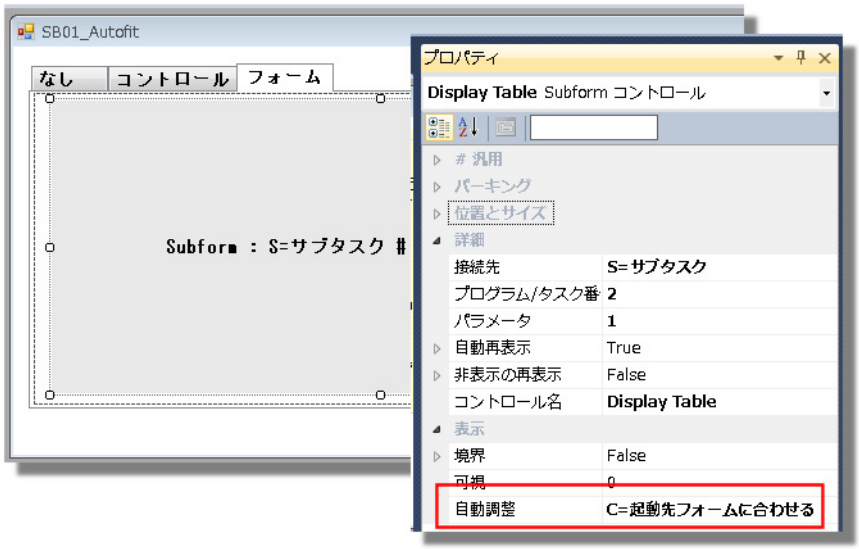
［このページは意図的に空白にしています。］

第 8 章： サブフォーム

サブフォームコントロールを起動プログラムフォームの寸法に合わせるには

サブフォームを定義する際に、起動されるプログラムのフォームサイズがわからない場合があります。推測して設定することもできますが、自動調整オプションを使用することで、コントロールのサイズを自動的にフォームサイズに合わせることができます。

自動調整を設定する



1. サブフォームコントロールを選択します。
2. ズームしてコントロール特性を開きます。
3. 自動調整特性で *C= 起動先フォームに合わせる* を選択します。

注： *起動先フォームに合わせる* に合わせるを使用すると、サブフォームのサイズは実行時の実際のサイズには影響しません。このため位置特性にも影響しません。

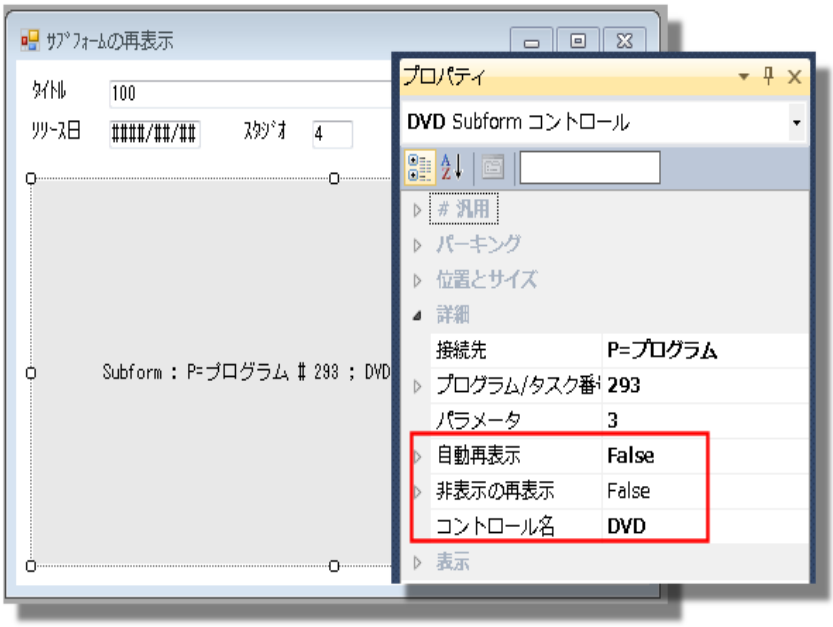
参照： 「自動調整オプションを使用するには」（195 ページ）

サブフォームを手動で再表示するには

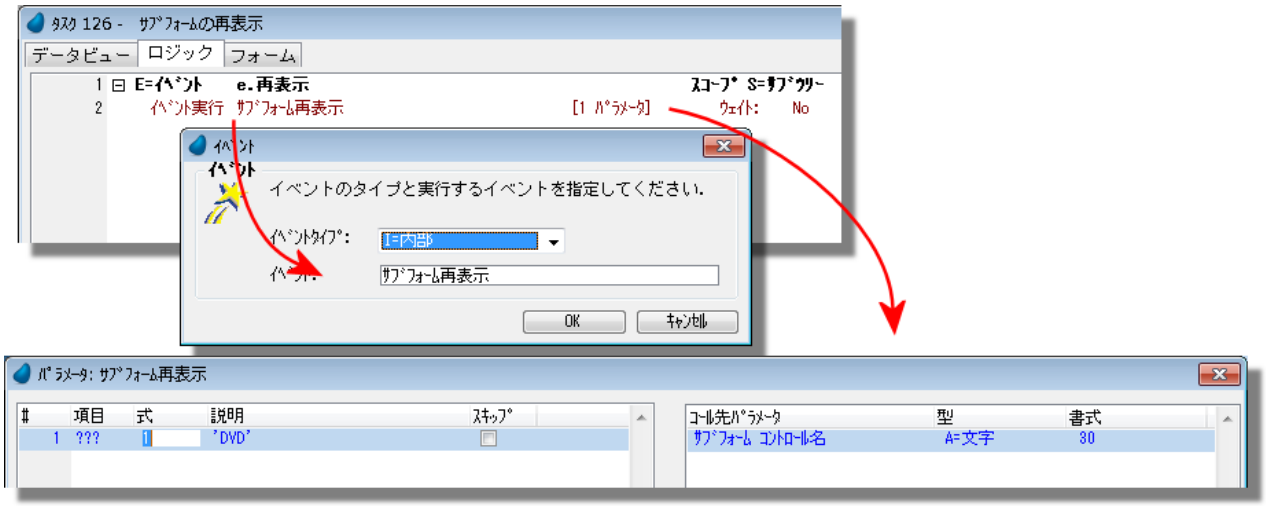
通常、子タスクに渡されるパラメータの値が変わった場合、サブフォームは自動的に再表示します。例えば、親タスクで DVD 一覧を表示し、サブフォームでその詳細を表示している場合、一覧表示をスクロールすると、詳細表示も自動的に変更します。

これは通常に要求される機能で、この処理のために開発者が何かを行う必要はありません。しかし、レコード長が大きかったり、検索が複雑で再表示に非常に時間がかかる場合があると、ユーザにとって面倒なことになるかもしれません。このような場合、自動的な再表示機能を無効にし、ユーザが検索（または再表示）のボタンをクリックした場合だけ実行させるようにした方が都合がいい場合もあります。ここでは、どのようにして無効にするかについて説明しています。

サブフォームの再表示を手動化する



1. サブフォームコントロールの**自動再表示**特性を *False* に設定します。
2. サブフォームコントロールの**コントロール名**特性を確認します。この例では、*DVD* となっています。

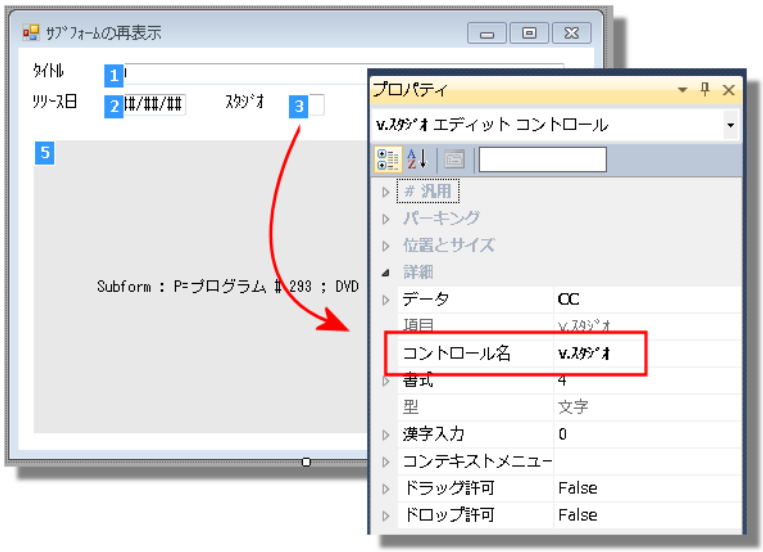


3. サブフォームを再表示したい場合に、実行されるユーザイベントを作成します。この例では、**e. 再表示**のユーザイベントを実行する**プッシュボタン**が定義されています。
4. **ロジックエディタ**で、イベントに対する**ロジックユニット**を作成します。
5. この**ロジックユニット**内で、**サブフォーム再表示**イベントを実行するように定義します。
6. **サブフォーム再表示**イベントへのパラメータとして、再表示したい**サブフォーム**のコントロール名を渡します。これで、ユーザがプッシュボタンをクリックすると、サブフォームは再表示されます。

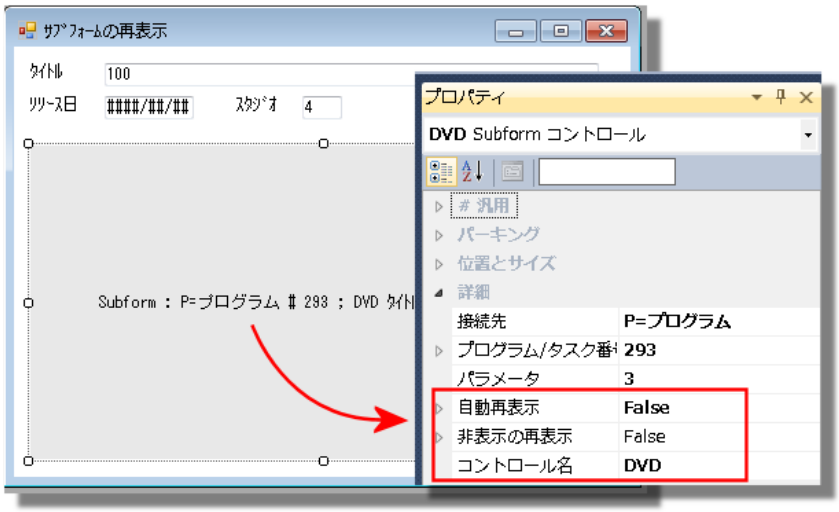
複数のパラメータをサブフォームに渡した場合、最後のパラメータを修正した時のみ再表示するには

通常、サブフォームに渡されるパラメータの値が変更されると再表示されます。しかし、リストに対する検索条件などのように複数のパラメータが指定されている場合、ユーザが条件値のどれかを変更するたびに、サブフォームは再表示されます。これは、ユーザの要求とは合っていないかもしれません。また、複雑な検索である場合、処理速度を落とす可能性もあります。

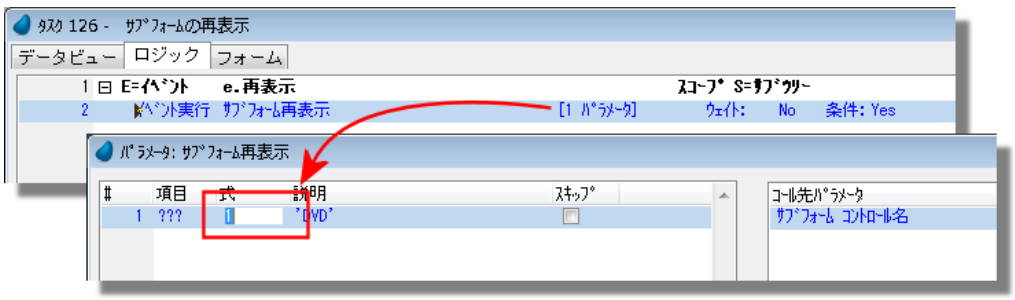
サブフォームの自動再表示特性を無効にする



1. 親タスク上で、ツールバーのタブオーダーアイコン  をクリックして、TAB 順序が正しく設定されていることを確認します。
2. 最後の項目にコントロール名が設定されていることを確認します（この例では v. スタジオになっています）。



3. サブフォームのコントロール特性で、**自動再表示**特性を *False* に設定します。
4. サブフォームコントロールの**コントロール名**を確認します。この例では、*DVD* となっています。



5. ロジックエディタで、最後の検索条件コントロールに対するコントロール後ロジックユニットを作成します。

6. イベント実行処理コマンドを作成し、サブフォーム再表示イベントを定義します。
7. このイベントへのパラメータとして、サブフォームのコントロール名（この例では **DVD** ）を渡します。
   これで、ユーザが最後のコントロールを通過すると、サブタスクは再表示されます。

*ヒント : サブフォームの **TAB で移動** 特性と組み合わせた場合、ユーザは条件に入った後で、サブフォームに正しく移るはずです。また、コントロール後が定義されたコントロールでは、**CtrlGoTo()** を持つアクション処理コマンドを追加することができます。これによってユーザがサブフォーム内のコントロールに正しくフォーカスを移すことができます。*

タブコントロールに配置したサブフォームの表示を制御するには

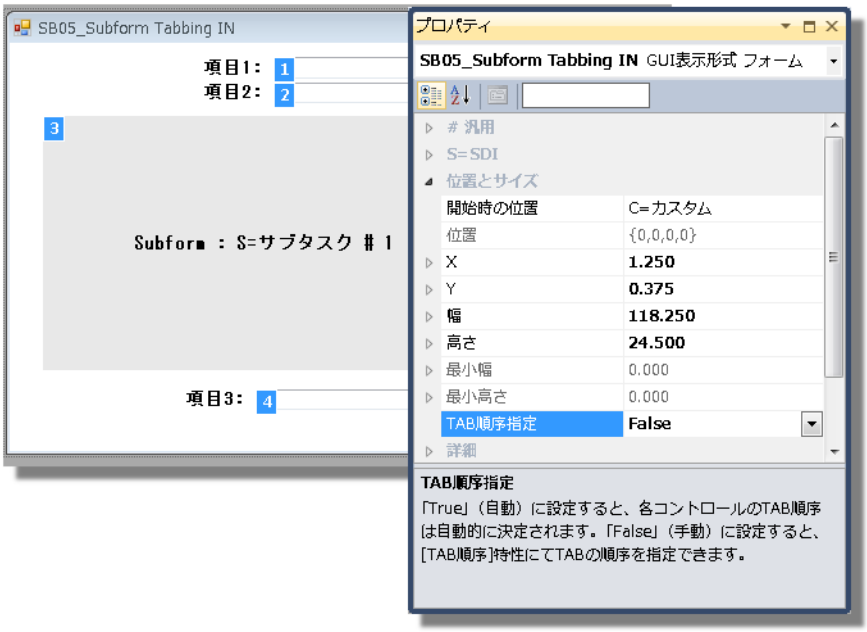
サブフォームを**タブ**コントロールに配置した場合、タブが選択された場合のみ表示されます。他にする必要は何もありません。他のコントロールと同じように配置するだけです。

参照 : 第 3 章 :「コントロールモデルを使用して、フォームにコントロールを自動配置するには」（46 ページ）
第 5 章 :「コントロールをリンクする」（122 ページ）

親フォームのコントロールからサブフォームに Tab 入力できるようにするには

通常、Magic xpa では、**TAB 順序**は左上から右下に順番に設定されます。しかし、手動で設定することもできます。

サブフォームコントロールの TAB 順序を設定する



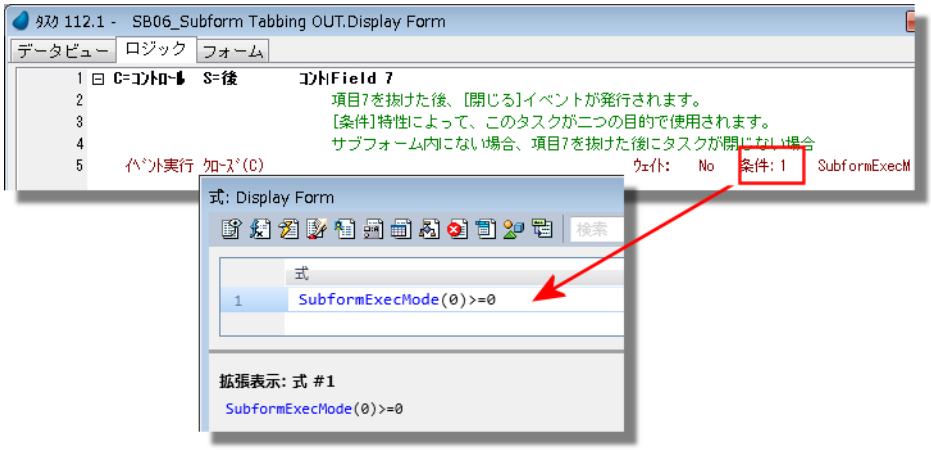
1. **フォーム特性**の **TAB 順序指定**特性を ***False*** に設定して、**自動 TAB 順序**モードを解除します。（解除されていない場合）
2. **サブフォーム**以外のコントロールの **TAB 順序**を設定します。
3. **サブフォーム**コントロールの **TAB 順序**を、他のコントロールより大きい値に設定します。
4. **サブフォーム**コントロールの**パーキング可**特性と **TAB で移動**特性がデフォルト値（***True***）に設定されていることを確認します。

注： 各コントロールに割り当てられた **TAB 順序**をを確認するには、表示メニューの**タブオーダー**を選択するか、ツールバーから  をクリックします。

サブフォーム上のコントロールから親のコントロールに自動的に戻るには

ユーザがサブフォームに移動した場合に、そこから抜けられるようにする必要があります。コントロールイベントを使用することでこのようなことを簡単に実現することができます。しかし、サブフォームプログラムは *Tab* 移動で終了することのない１つのプログラムとして受け取られます。このため、終了する必要がある場合 **SubformExecMode()** 関数を使用して制御します。

サブフォームを終了するイベントを設定する



1. どのコントロールが最後にパークされるのかを決定し、そのコントロールにコントロール名が定義されていることを確認します。この例では、住所という名前が設定されています。
2. コントロール後ロジックユニットを作成し、そのコントロール名を指定します。
3. イベント実行処理コマンドを作成し、**条件**特性に以下の式を設定します。

   ```
   SubformExecMode(0)>=0
   ```

   タスクがサブフォームとして呼ばれていない場合、**SubformExecMode()** 関数は -1 を返します。戻り値がこれ以外の場合、サブフォームがどのように起動されたかの情報が取得できます。『リファレンスヘルプ』を参照してください。

最初にサブフォームタスクが起動されたときのみタスク前 / 後を実行させるには

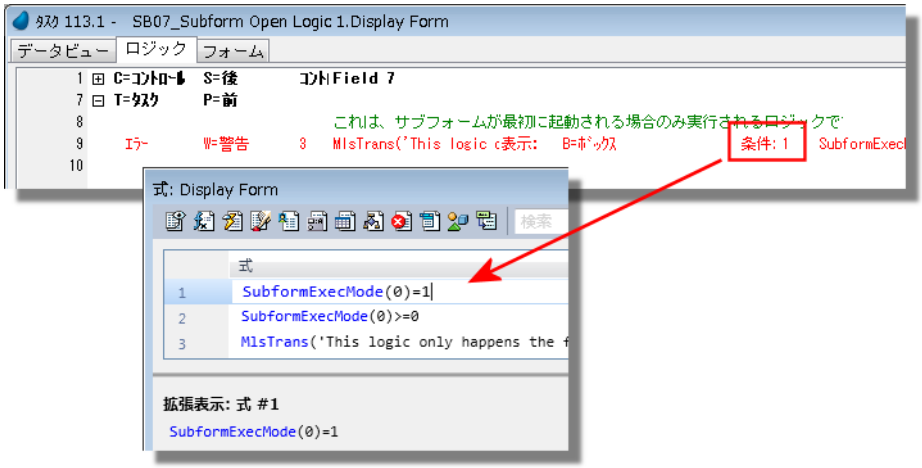
タスク開始時は常にタスク前が実行され、タスク終了時は常にタスク後が実行されます。しかしサブフォームを使用する場合、親タスクから最初に起動された場合に、内容を表示するために、タスク前／後を一度を実行します。しかし、親タスクが再表示した場合、再びユーザがサブタスクに入った場合も実行されます。このため、サブフォームタスクが最初に起動されたときのみ実行するように定義する場合は、**SubformExecMode()** 関数を使用する必要があります。

サブフォームタスクが最初に起動された場合、**SubformExecMode(0)** は 1 を返します。このため以下の条件式を使用します。

```
SubformExecMode(0) = 1
```

ブロック処理コマンドの条件にこの式を設定すると、サブフォームが最初に起動された場合のみ、ブロック内の処理が実行されます。

サブフォームが最初に起動された場合のみ実行するブロックを定義する



1. 修正中のロジックユニットで詳細行を追加します（*F4*、または編集→行作成）。
2. B を入力するかプルダウンメニューを使用して、ブロック処理コマンドを選択します。
3. ***ブロック If*** と ***ブロック End*** の２行が追加されます。*If* の後のカラムに移動します。
4. ズーム（*F5*、またはダブルクリック）して式エディタを開き、以下の式を入力します。

   ```
   SubformExecMode(0)=1
   ```

5. *Enter* を押下して式を確定しロジックエディタに戻ります。

これで、サブフォームが最初に起動されたときのみブロックは実行されます。

ユーザがサブフォームに入ると常にタスク前 / 後が実行されるようにするには

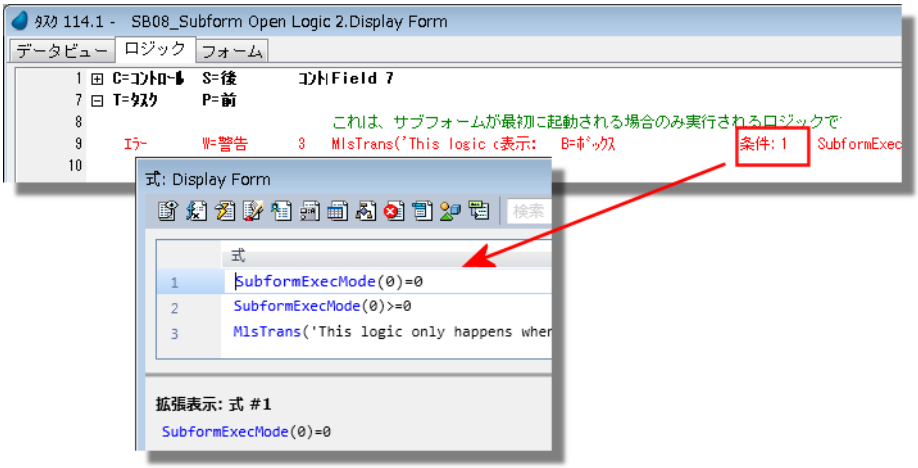
タスク開始時は常にタスク前が実行され、タスク終了時は常にタスク後が実行されます。しかし、サブフォームでは、どのように起動されたかによって処理内容を制御するためにブロック処理コマンドを利用することができます。この場合、**SubformExecMode()** 関数を使用します。

ユーザの操作（*Tab* 移動やマウスクリックなど）でサブフォームタスクに入った場合、**SubformExecMode(*0*)** 関数は「0」を返します。このため以下の条件式で判別できます。

```
SubformExecMode(0) = 0
```

ブロック処理コマンドの条件にこの式を設定すると、ユーザの操作によってサブフォームが起動された場合のみ、ブロック内の処理が実行されます。

ユーザがサブフォームに入るときにのみ実行するブロックを定義する



1. 修正中のロジックユニットで詳細行を追加します（*F4*、または編集→行作成）。
2. *B* を入力するかプルダウンメニューを使用して、ブロック処理コマンドを選択します。
3. ***ブロック If*** と ***ブロック End*** の２行が追加されます。*If* の後のカラムに移動します。
4. ズーム（*F5*、またはダブルクリック）して式エディタを開き、以下の式を入力します。
   ```
   SubformExecMode(0)=0
   ```
5. *Enter* を押下して式を確定しロジックエディタに戻ります。

これで、ユーザがサブフォームに入った場合のみブロックは実行されます。

サブフォームが再表示されたときにサブタスクのタスク前 / 後を実行させるには

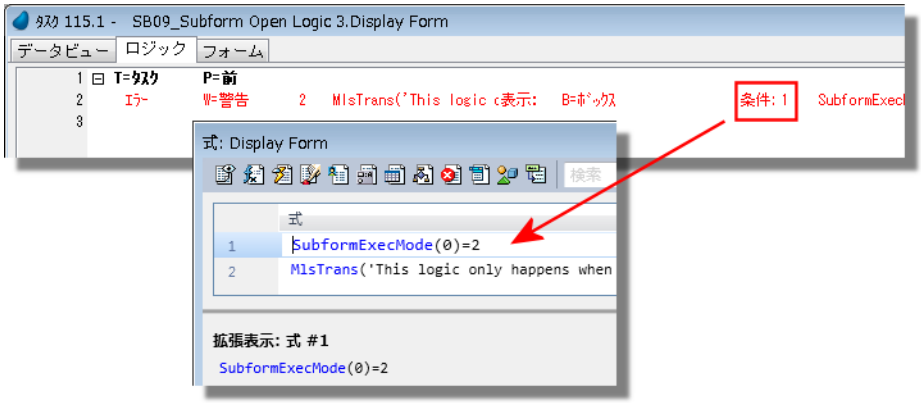
タスク開始時は常にタスク前が実行され、タスク終了時は常にタスク後が実行されます。しかし、サブフォームでは、どのように起動されたかによって処理内容を制御するためにブロック処理コマンドを利用することができます。この場合、**SubformExecMode()** 関数を使用します。

自動再表示や手動のサブフォームを再表示イベントなどで再表示処理を行うことによりサブフォームタスクを実行させた場合、**SubformExecMode(0)** は 2 を返します。このため以下の条件式で判別できます。

```
SubformExecMode(0) = 2
```

ブロック処理コマンドの条件にこの式を設定すると、サブフォームが再表示した場合のみ、ブロック内の処理が実行されます。

サブフォームが再表示されたときにのみ実行するブロックを定義する



1. 修正中のロジックユニットで詳細行を追加します（*F4*、または編集→行作成）。
2. *B* を入力するかプルダウンメニューを使用して、ブロック処理コマンドを選択します。
3. *ブロック If* と *ブロック End* の 2 行が追加されます。*If* の後のカラムに移動します。
4. ズーム（*F5*、またはダブルクリック）して式エディタを開き、以下の式を入力します。

   ```
   SubformExecMode(0)=2
   ```

5. *Enter* を押下して式を確定しロジックエディタに戻ります。

これで、サブフォームが再表示された場合のみブロックは実行されます。

自動調整オプションを使用するには

サブフォームコントロールの**自動調整**特性は設定内容によって動作が異なります。これにより、どれを選択すればいいか決めかねる場合があります。以下の例は、３つのオプションを選択した場合どのように動作するかを説明したものです例です。３つの場合のサブフォームコントロールはまったく同じサイズになっています。

自動調整オプションの結果

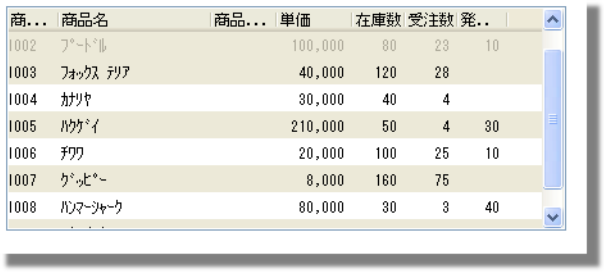
N= なし

商品番号	商品名	商品タイプ	単価	在庫数	受注数	発注
1002	ﾌﾟｰﾄﾞﾙ	D	100,000	80	23	
1003	ﾌｫｯｸｽ ﾃﾘｱ	D	40,000	120	28	
1004	ｶﾅﾘﾔ	B	30,000	40	4	
1005	ﾊｳｹﾞｲ	F	210,000	50	4	
1006	ﾁﾜﾜ	D	20,000	100	25	
1007	ｸﾞｯﾋﾟｰ	F	8,000	160	75	
1008	ﾊﾝﾏｰｼｬｰｸ	F	80,000	30	3	
1009	ﾌﾞﾝﾁｮｳ	B	8,000	80	4	
1010	ﾊﾑｽﾀｰ	R	1,000	140	69	

なし：サブフォームタスクのフォームは、**サブフォーム**コントロールのサイズ内に収まる内容のみ表示されます。呼び出されたフォームが大きすぎる場合、切り取られ、スクロールバーが表示されます。例えば、**サブフォーム**コントロールの幅が 80 ユニットに設定され、呼び出されたフォームの幅が 88 ユニットに設定されている場合、実行時は 80 ユニットのサイズでのみ表示されます。

サブフォームコントロールに**位置**特性が設定され、フォームのサイズが変更される場合、サブフォームタスクはより広い範囲を表示することができますが、サブフォームタスク内のコントロールは変更されません。

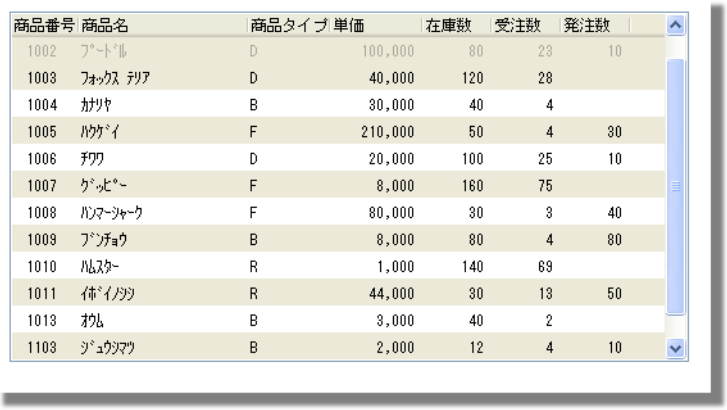
A= コントロールに合わせる



商...	商品名	商品...	単価	在庫数	受注数	発...	
1002	ﾌﾟｰﾄﾞﾙ		100,000	80	23	10	
1003	ﾌｫｯｸｽ ﾃﾘｱ		40,000	120	28		
1004	ｶﾅﾘﾔ		30,000	40	4		
1005	ﾊｳｹﾞｲ		210,000	50	4	30	
1006	ﾁﾜﾜ		20,000	100	25	10	
1007	ｸﾞｯﾋﾟｰ		8,000	160	75		
1008	ﾊﾝﾏｰｼｬｰｸ		80,000	30	3	40	

コントロールに合わせる：サブフォームタスクはコントロールのサイズに適合されます。ユーザがサブフォームタスクのサイズを変更したように表示されます。従って、サブフォームタスクで**位置**特性が設定されている場合、コントロールのサイズはそれに応じて伸びたり、縮んだりします。

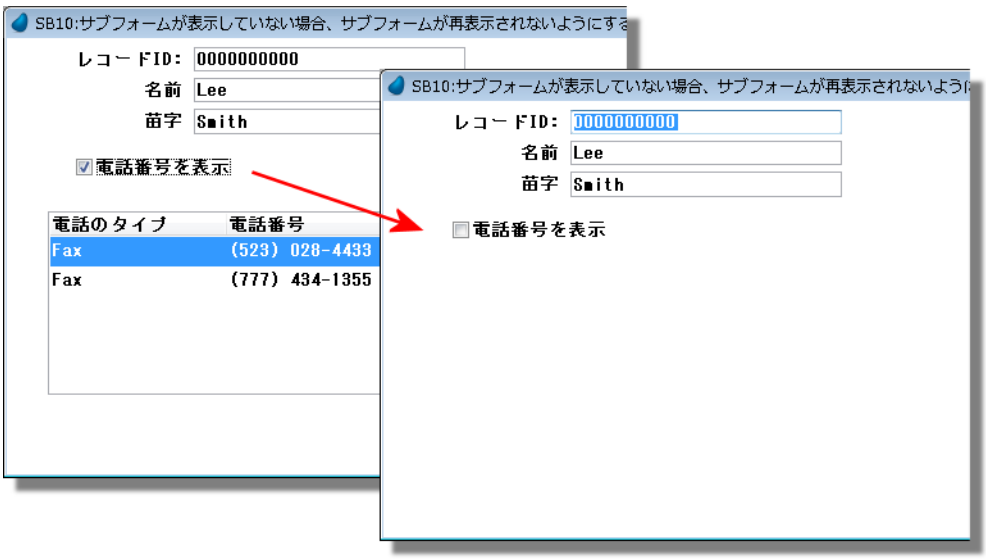
C= 起動先フォームに合わせる



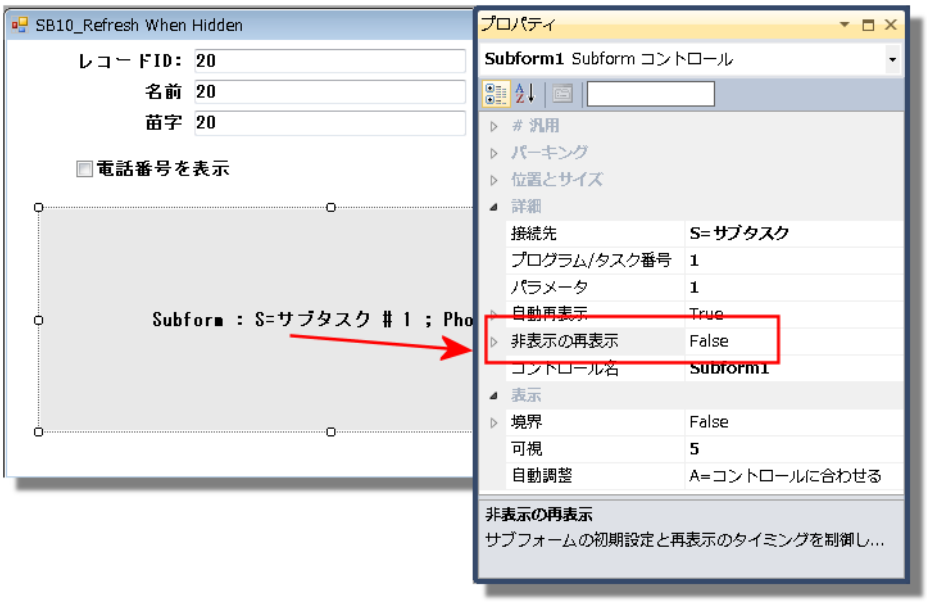
商品番号	商品名	商品タイプ	単価	在庫数	受注数	発注数
1002	ﾌﾟｰﾄﾞﾙ	D	100,000	80	23	10
1003	ﾌｫｯｸｽ ﾃﾘｱ	D	40,000	120	28	
1004	ｶﾅﾘﾔ	B	30,000	40	4	
1005	ﾊｳｹﾞｲ	F	210,000	50	4	30
1006	ﾁﾜﾜ	D	20,000	100	25	10
1007	ｸﾞｯﾋﾟｰ	F	8,000	160	75	
1008	ﾊﾝﾏｰｼｬｰｸ	F	80,000	30	3	40
1009	ﾌﾞﾝﾁｮｳ	B	8,000	80	4	80
1010	ﾊﾑｽﾀｰ	R	1,000	140	69	
1011	ｲﾎﾞｲﾉｼｼ	R	44,000	30	13	50
1013	ｵｳﾑ	B	3,000	40	2	
1103	ｼﾞｭｳｼﾏﾂ	B	2,000	12	4	10

起動先フォーム合わせる：**サブフォーム**コントロールは、サブフォームタスクのフォームサイズに合わせて大きさが変更されます。この例では、**サブフォーム**コントロールが上記の２つの例と同じサイズですが、**サブフォーム**コントロールは、サブタスクフォーム全体に収まるように拡大されて表示しています。この例では、**サブフォーム**コントロールの**位置**特性は全く影響しません。

サブフォームが表示していない場合、サブフォームが再表示されないようにするには

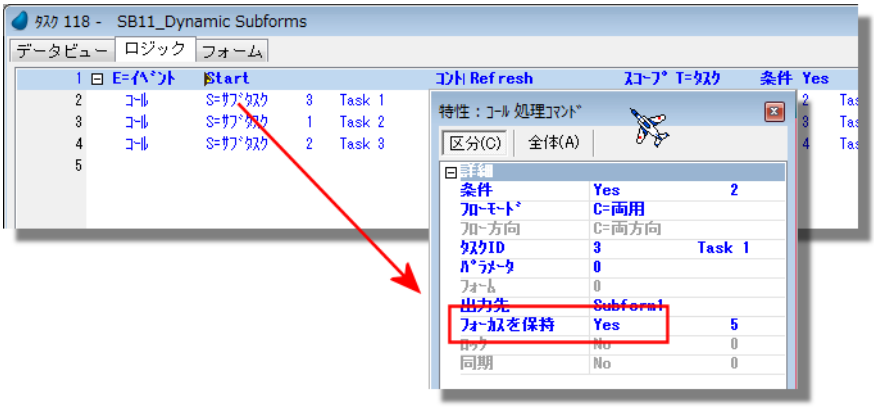


表示されないサブフォームを使用することは、有益な場合もあります。これはタブコントロール上に配置したり、一度に一つだけ表示されるように、複数のサブフォームを重ねて表示させる場合に利用できます。この例では、出荷情報を必要に応じて表示させたり非表示にしたりしています。そして、他のコントロールは、Y 特性を使用して移動するようにしています。しかし、サブフォームが表示されない場合に情報を再表示するのは処理時間を浪費して無駄になります。



サブフォームが再表示されることを防止するには、**サブフォーム**コントロールの**非表示の再表示**特性を ***False*** に設定します。これで、サブフォームが表示されているときだけ再表示されます。

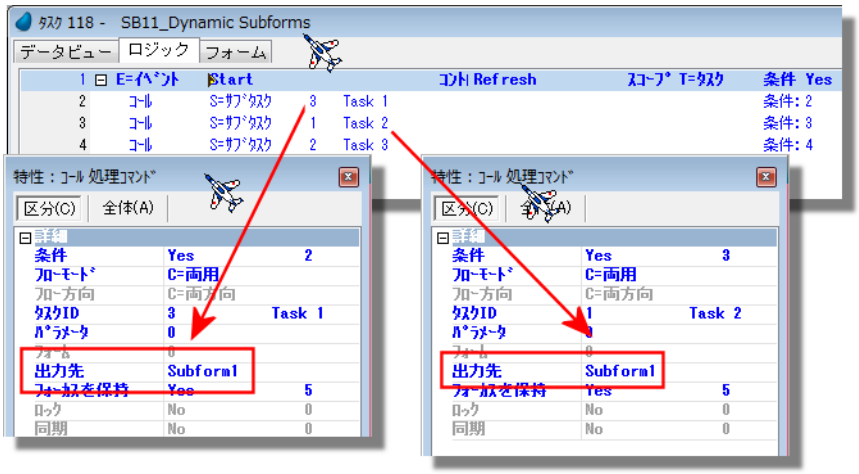
新しいタスクを呼び出すときにフォーカスを現在のタスクに保持するには



通常、新しいタスクを呼び出すと、フォーカスは新しいタスクへ移ります。しかし、タスクを表示するだけでカーソルの位置を変更したくない場合もあります。この例では、ユーザはプログラムがどのサブフォームを表示するかを選択することができます。しかし、最初のレコード上に位置付けられたままでフォーカスは変わりません。

このようにするには、**コール処理**コマンドの**フォーカスを保持**特性の *Yes* に設定します。

サブフォームの表示内容を動的に置き換えるには



通常、各サブフォームは、1 つのプログラムまたはタスクを表示します。しかし、サブフォームの内容を指定した内容に動的に変更することもできます。この例では、ユーザはメニューから顧客データか受注データの表示を選択することができます。そして、選択内容に応じて異なるプログラムを呼び出しています。

1. **サブフォーム特性**に、サブフォームのコントロール名を設定します。
2. 表示させたいプログラムまたはサブタスクを呼び出すために、**コール**処理コマンドを定義します。
3. **出力先**特性にサブフォームの**コントロール名**を設定します。この例では、***Subform1*** に呼び出されるサブフォームは受注データか顧客データが表示されるために使用されます。

これでプログラムを実行させると、指定されたタスクがサブフォーム上に表示されます。

注： 同様に、サブフォームは、タブコントロール上に使用することもできます。この場合、サブフォームを適切なタブに配置してみてください。コール処理コマンドは必要ありません。フレームに、この方法を使用することもできます。

サブフォームで使用される子タスクで、[空のデータビュー] イベントを処理するには

製品仕様では、タスクにフォーカスがある時のみ、サブフォームタスクの［空のデータビュー］イベントが実行されます。

しかし、サブフォームタスクのデータビューがない場合に、フォーカスをサブフォームタスク上に設定する必要が無い状態で実行しなければならないようなサブフォームのロジックを実装する必要があるかもしれません。

この場合、以下のようにすることができます。

1. サブフォームタスクに、［親による再表示後］イベントに対する［イベント］ロジックユニットを追加し、以下の式によって条件づけられるブロックを作成してください：
   ```
   EmptyDataview (0)
   ```
2. このブロック内で、サブフォームタスクが、フォーカスをタスクのフォーム上に設定することなく、データビューが無い状態にあるとき、実行させたいロジックを定義します。

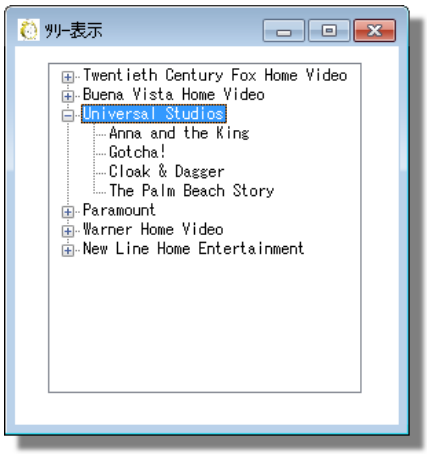
注： 現在、この内部のイベント（親による再表示後）は、リッチクライアントタスクでのみサポートされます。

［このページは意図的に空白にしています。］

第 9 章： ツリーコントロール

データをツリー形式で表示するには

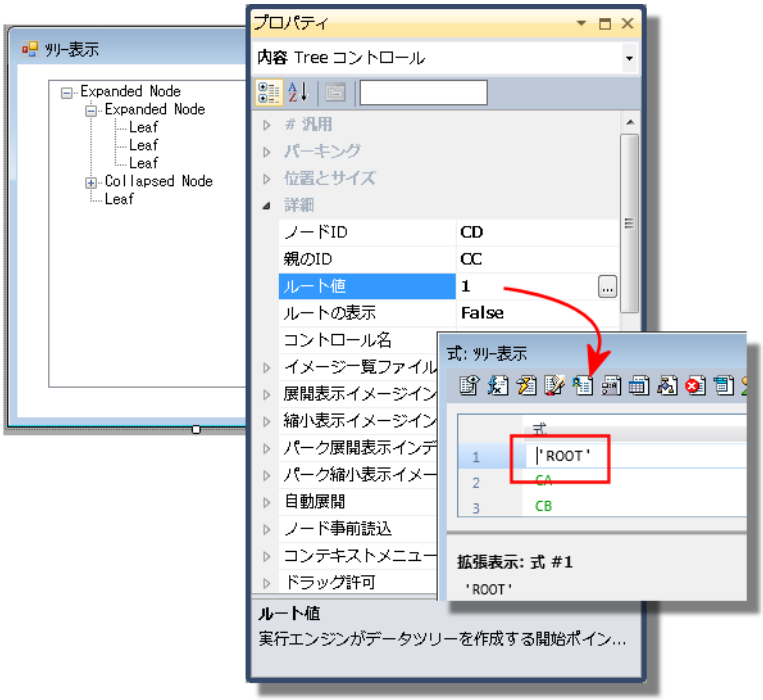
ツリー形式は、データを一覧表示する場合に便利です。この形式は Windows 環境では一般的で、ユーザにも見慣れている表示です。



Magic xpa には、必要に応じてこのような表示を行うためのツリーコントロールがあります。開発者が行わなければいけないことは、階層構造に定義されたデータソース（メモリテーブルで十分です）にデータを格納し、必要なオプションを設定することです。

ここでは、基本的な作業手順を説明しています。

ツリーコントロールを定義する



1. ツリーに対応した書式でデータを保存します。「ツリー表示させるための階層的なデータソースを定義するには」（203 ページ）を参照してください。
2. データビューエディタでメインソースとしてツリー用のデータソースを定義し、タスクに必要なカラム（ノード ID、親 ID など）を定義します。
3. コントロールパレットからツリーコントロールを選択し、フォームにドロップします。サイズに必要に応じて変更します。
4. ツリーのコントロール特性を開き、以下の特性値を設定します。
   - **ノード ID**：レコードのキーとなるカラムを指定します。
   - **親 ID**：このレコードの親のレコードを示す ID が格納されたカラムを設定します。

- **ルート値**：ツリーの最上位のレコードを示す値を指定します（この例では、ROOT です）。
- **表示項目**：ツリーの上で値が表示される項目を指定します。

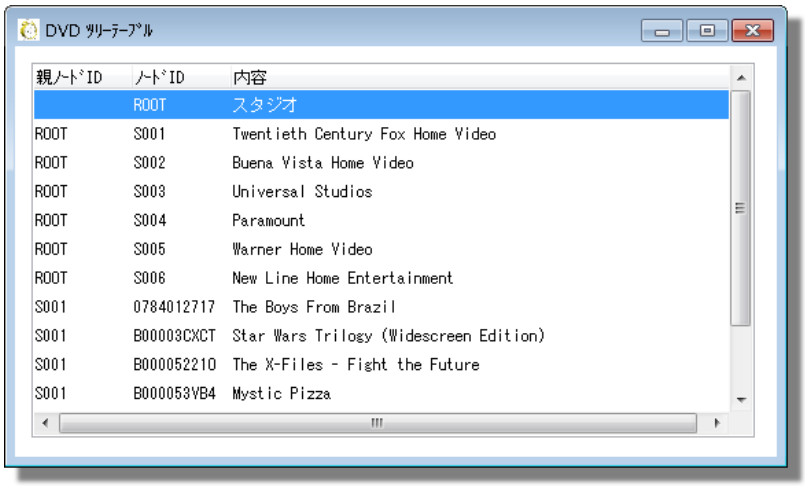
データの書式が正しく設定されていれば、テーブルとして表示されます。実行時に、テーブルの外観やどのように動作するかを制御するためのオプションがあります。これらの内容については以下の説明を参照してください。

参照： 「ツリー表示させるための階層的なデータソースを定義するには」（203 ページ）
「ツリーのノードにアイコンを設定するには」（204 ページ）
「ツリーを表示する際にノードを自動的に展開させるには」（209 ページ）

ツリー表示させるための階層的なデータソースを定義するには

ツリーを表示させるためには、正しく構造化されたデータを定義する必要があります。このためには、一時的にデータを保持するためのメモリテーブルを作成する必要があります。

構造を実現する上で２つの項目（*ノード ID* と *親 ID*）が必要です。**ノード ID** は特定のレコードを識別するもので、レコード毎にのユニークな ID が定義されている必要があります。**親 ID** はこのレコードの親を識別するためのものです。

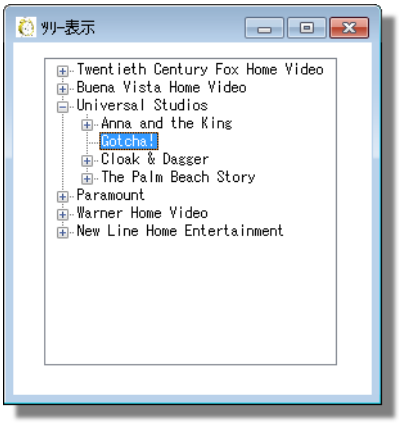


再帰

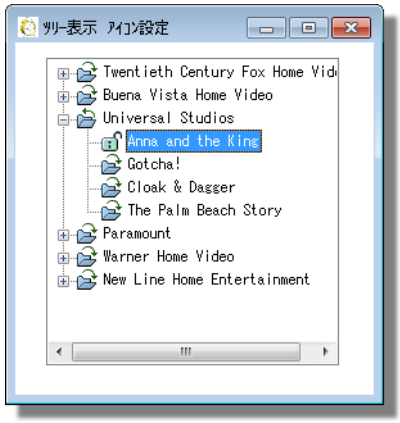
各レコードの**ノード ID** と**親 ID** には異なる値が設定されていなければなりません。異なっていない場合、1 つの枝から何回でも開くことのできるができる再帰的なツリーになってしまいます。例えば、**ノード ID**：S002 を持つレコードは、S002 の **親 ID** を持つことができないということです。また、**親 ID** と**ノード ID** の両方を空白にできるのは、ルートノードのみになります。

ヒント：*ツリーを複数の異なるデータソースから構成することは可能です。この例では、**スタジオ**テーブルと **DVD** テーブルのデータを使用しています。これらのデータソースはそれぞれが非常に異なるデータが定義されているため、１つのツリーとして表示させるため、最低限のカラムのみ使用するようにしています。より多くの情報を表示したい場合は、一時テーブルにコピーして使用しないで、オリジナルのデータソースとリンクして使用してください。*

ツリーのノードにアイコンを設定するには



アイコン未設定



アイコン設定済み

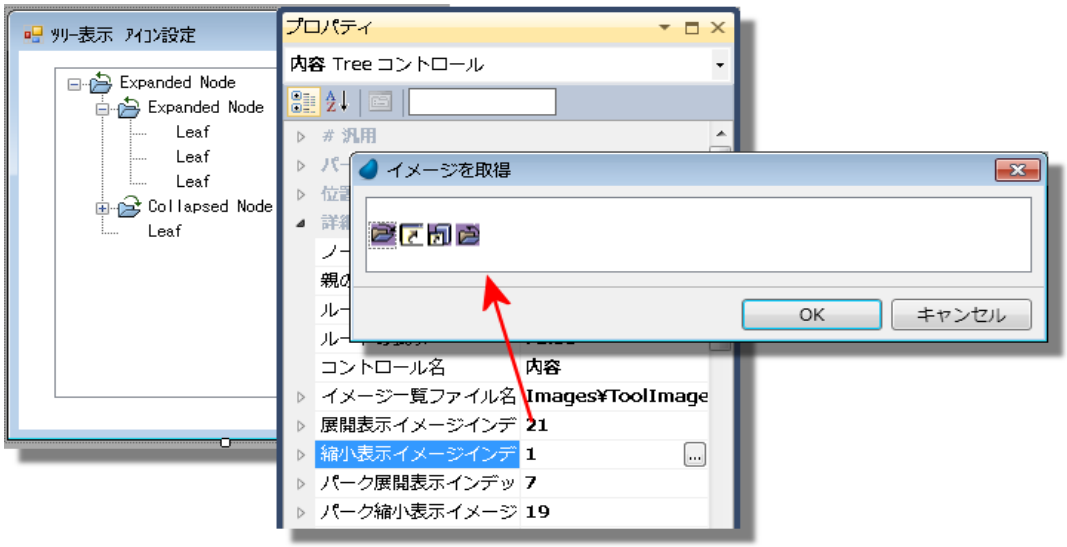
ツリーコントロールにアイコンを表示させることができます。

ツリーには以下の４つの状態があります。

- **展開表示イメージ**：ノードが完全に展開されているか、子ノードが存在しない場合
- **縮小表示イメージ**：ノードが縮小されている場合
- **パーク展開表示イメージ**：カーソルが展開されたノード上でパークされている場合
- **パーク縮小表示イメージ**：カーソルが縮小したノード上でパークされている場合

これらのアイコンは各々、番号を指定することによって設定されます（これはアイコンファイルにおけるインデックスです）。インデックスが 0 に設定されている場合、イメージは表示されません。

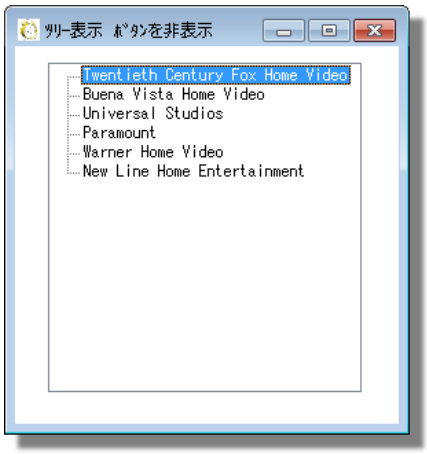
アイコンを使用する



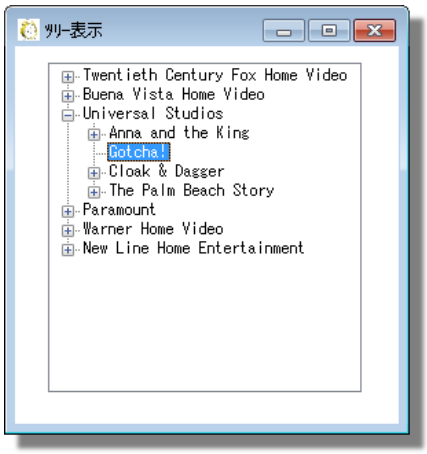
必要条件：　アイコンファイルを事前に用意しておく必要があります。アイコンファイルは、必要なアイコンが同じサイズで水平方向に並んだ状態で定義されているビットマップファイルです。どのアイコンを使用するかは、番号で指定されます。これらは、汎用的なイメージファイルを購入したり、独自に作成することで入手してください。

1. ツリーコントロールを選択します。
2. コントロール特性の**イメージ一覧ファイル名**特性を選択します。
3. ズームしてイメージファイルを選択するか、直接ファイル名を入力します。
4. ４つの状態の**イメージインデックス**特性でズームしてアイコンを選択します。

展開 / 縮小ボタンを表示 / 非表示するには



ボタンの表示 = *No*



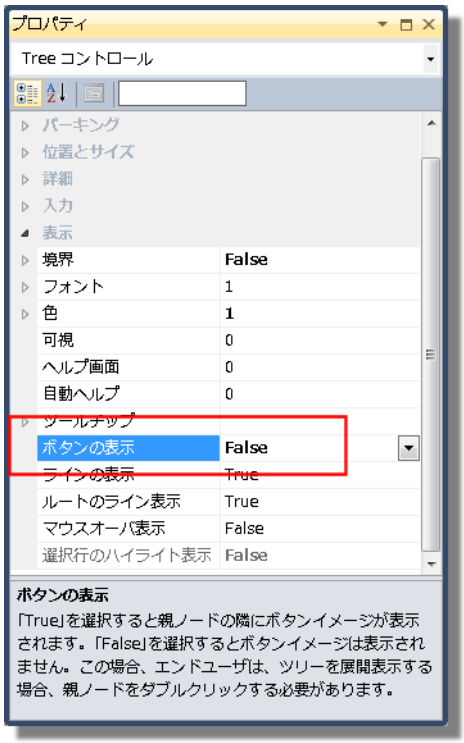
ボタンの表示 = *Yes*

ツリーの各ノードは、デフォルトで隣に小さなプラス（+）またはマイナス（−）のボタンが表示されています。+をクリックするとノードが展開表示され、−をクリックすると縮小表示されます。

ボタンの表示特性を使用することで、これらのボタンの表示／非表示を切り替えることができます。

展開 / 縮小ボタンの表示 / 非表示を切り替える

1. ツリーコントロールを選択します。
2. コントロール特性の**ボタンの表示**特性を選択します。
3. ボタンを表示する場合は *True*、そうでない場合は *False* を設定します。

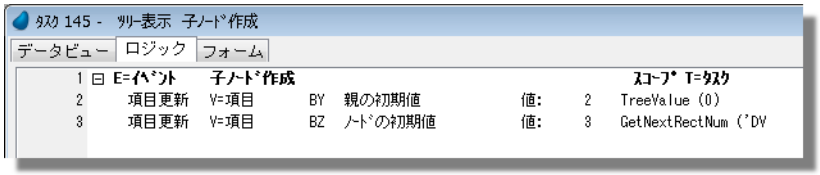


注： **ボタンの表示**特性を *True* に設定すると、ツリーが最初に表示された場合、*ツリーのリーフ*（サブノードがないノード）は、+がデフォルトで表示されます。このような状態を回避するには、**ノードの事前読み込み**特性を *True* に設定してください。「子ノードを持つノードのみ展開ボタンを表示させるには」（211 ページ）も参照してください。

実行時にノードを追加するには

ツリーが表示されている場合、その機能の多くは**テーブル**コントロールの動作と同じです。しかし、ツリーではデータは平面的に表示されません。従って、ユーザがレコードを追加した場合、そのレコードは兄弟関係のノードか、子ノードになります。どちらの場合も、新しいノードを作成する際の親ノードと現在のノードの初期設定が正しく行われなければなりません。

子ノードを作成する



1. **子ノード作成**イベントを実行する**プッシュボタン**を定義します。**プッシュボタン**のラベルは、子ノード作成（アプリケーションに合わせて任意に設定可）に設定します。子ノード作成はプルダウンメニューにはデフォルトで存在しないので、実行時のオプションメニューに表示させたい場合は、追加する必要があります。

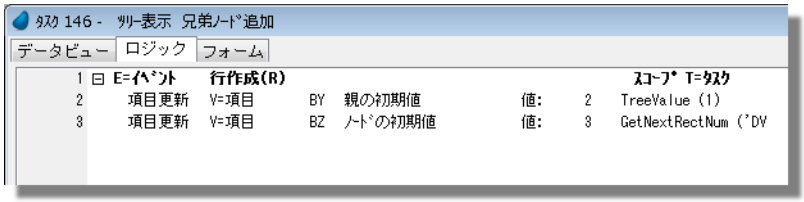
2. **データビューエディタ**内で、レコードが作成された場合に**親 ID** と**ノード ID** の初期値を設定するための変数項目を作成します。
3. **子ノード作成**イベントに対する**ロジックユニット**を作成し、**伝播**特性を ***Yes*** に設定します。
4. **ロジックユニット**内で、新規レコードが子ノードになるように、適切に項目を更新します。これは、**親 ID** の項目には現在の項目の**ノード ID** の値が設定される必要があることを意味しています。ID は、**TreeValue(*I*)** 関数を使用することで自動的に取得することができます。
   また、ユニークな値を**ノード ID** として初期設定する必要があります。この例では、自動的にユニークなキーを取得するために作成した**ユーザ定義関数**を使用しています。

   これで、新規ノードは適切に初期設定され、ユーザによって必要なデータを入力できるようになります。

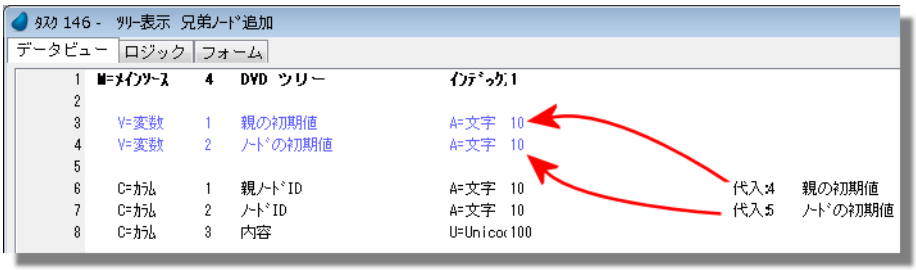
実行時に兄弟関係のノードを追加するには

ツリーが表示されている場合、その機能の多くはテーブルコントロールの動作と同じです。しかし、ツリーではデータは平面的に表示されません。従って、ユーザがレコードを追加した場合、そのレコードは兄弟関係のノードか、子ノードになります。どちらの場合も、新しいノードを作成する際の親ノードと現在のノードの初期設定が正しく行われなければなりません。

兄弟関係のノードを作成する



1. 行作成イベントを実行するプッシュボタンを定義します。プッシュボタンのラベルは、兄弟ノード作成（アプリケーションに合わせて任意に設定可）に設定します。*F4* の押下（または編集→行作成）でイベントを発生させることもできますが、エンドユーザは気がつかない可能性があります。



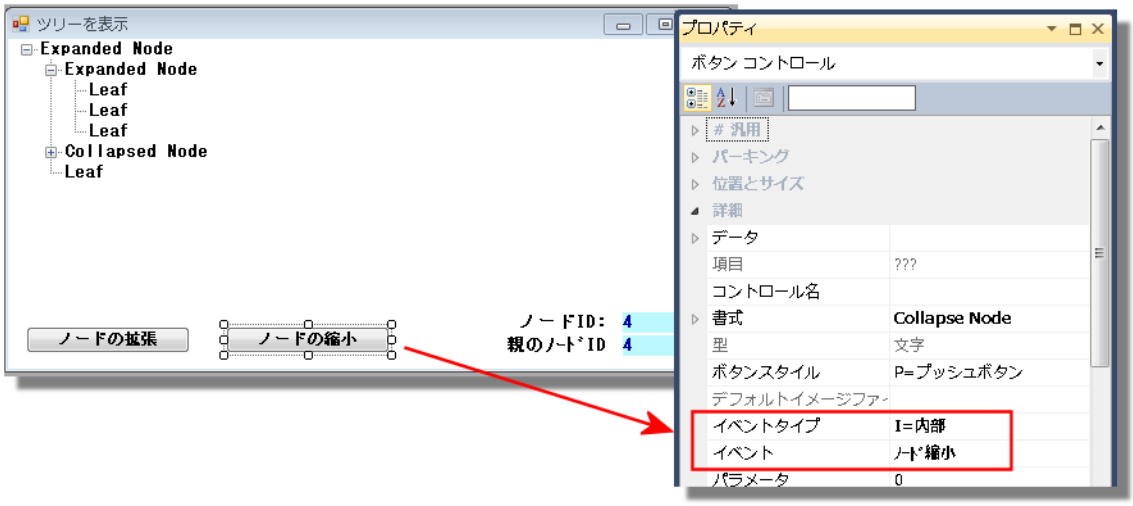
2. データビューエディタ内で、レコードが作成された場合に**親 ID** と**ノード ID** の初期値を設定するための変数項目を作成します。
3. 行作成イベントに対するロジックユニットを作成し、**伝播**特性を *Yes* に設定します。
4. ロジックユニット内で、新規レコードが兄弟ノードになるように、適切に項目を更新します。これは、**親 ID** の項目には現在の項目の**ノード ID** の値が設定される必要があることを意味しています。ID は、**TreeValue(*1*)** 関数を使用することで自動的に取得することができます。
   また、ユニークな値を**ノード ID** として初期設定する必要があります。この例では、自動的にユニークなキーを取得するために作成した**ユーザ定義関数**を使用しています。

   これで、新規ノードは適切に初期設定され、ユーザによって必要なデータを入力できるようになります。

実行時にツリーノードを明示的に展開 / 縮小させるには

ユーザが、ツリーのノードを展開したり縮小する場合、＋表示や－表示をクリックします。しかし、イベントを実行することで明示的にノードを展開したり縮小する場合は、ノード展開 / ノード縮小イベントを使用します。

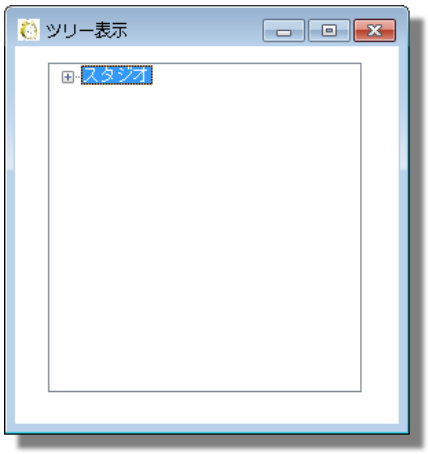
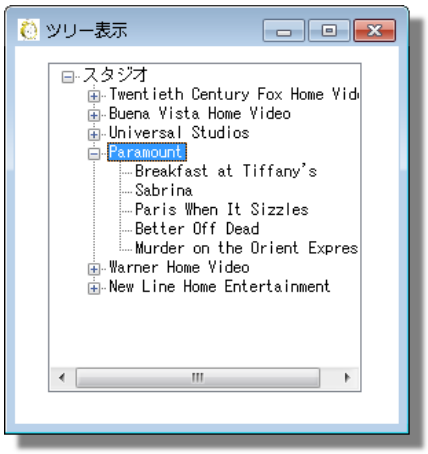
ノード展開 / ノード縮小イベントを実行する



ロジックユニットにイベント実行処理コマンドを定義したり、プッシュボタンにイベントを割り当てることでこれらのイベントを他のイベントと同じように実行させることができます。

どちらのイベントも、ユーザが現在パークしているノードで実行されます。

ツリーを表示する際にノードを自動的に展開させるには

	
自動展開 = *False*	**自動展開** には式が定義され、ルートノードと１つのノードが展開されるように設定されています。

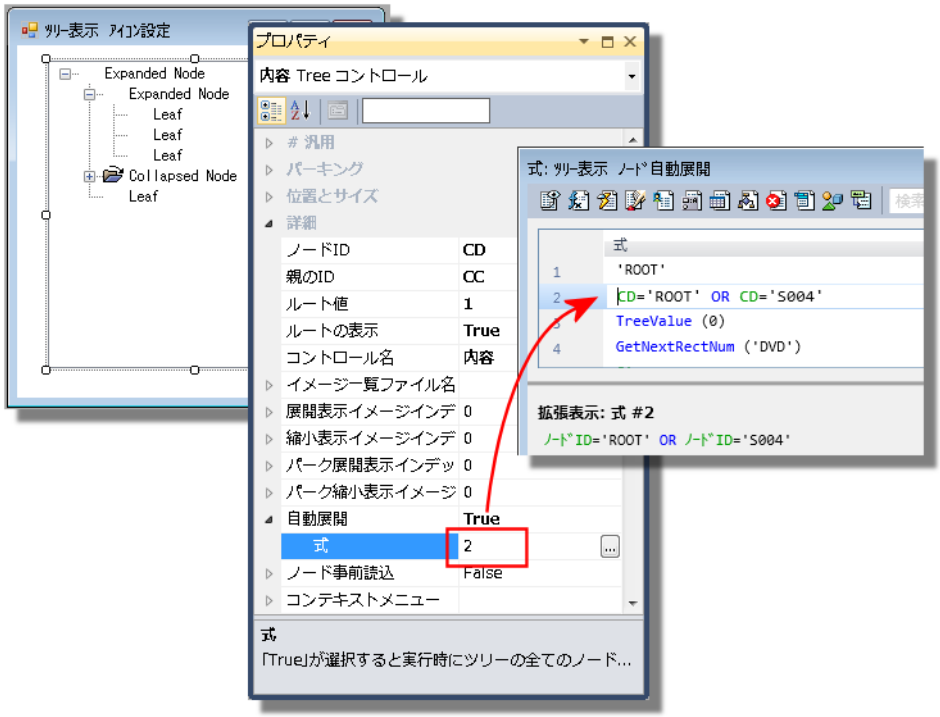
デフォルトでは、ツリーが開く際、上図の左の例のようにノードは展開されないようになっています。ユーザが、ノードを**ダブルクリック**したり、＋表示を**クリック**することで展開することができます。

しかし、ツリーが表示される時点でノードを展開するか否かは、開発時に指定することができます。これは、**自動展開**特性を使用することで実現できます。

自動展開特性が *True* に設定されている場合、すべてのノードは展開表示されます。しかし、式で指定することで、実行時に動的に指定することもできます。右の上の例では２つのノード（ROOT と S004）を展開しています。

この機能は、**TreeNodeGoto()** 関数と一緒に使用することで、ツリーを開き、カーソルを特定のノードに位置付けることができます。

特定のノードに対して自動展開を設定する



1. ツリーコントロールを選択します。
2. コントロール特性の**自動展開**特性を選択し、*True* を設定します。この設定では、ツリーが開くとすべてのノードが展開されます。

指定されたノードのみを展開させるには、式を指定します。この例では、ノード ID が ROOT か S004 の場合に展開するように設定されています。これによって上図の例のようにルートノードと１つのノードのみ展開されます。

自動展開をツリーレベルで設定する

TreeLevel() 関数を使用することで、特定のツリーレベルだけを展開させるようにすることができます。

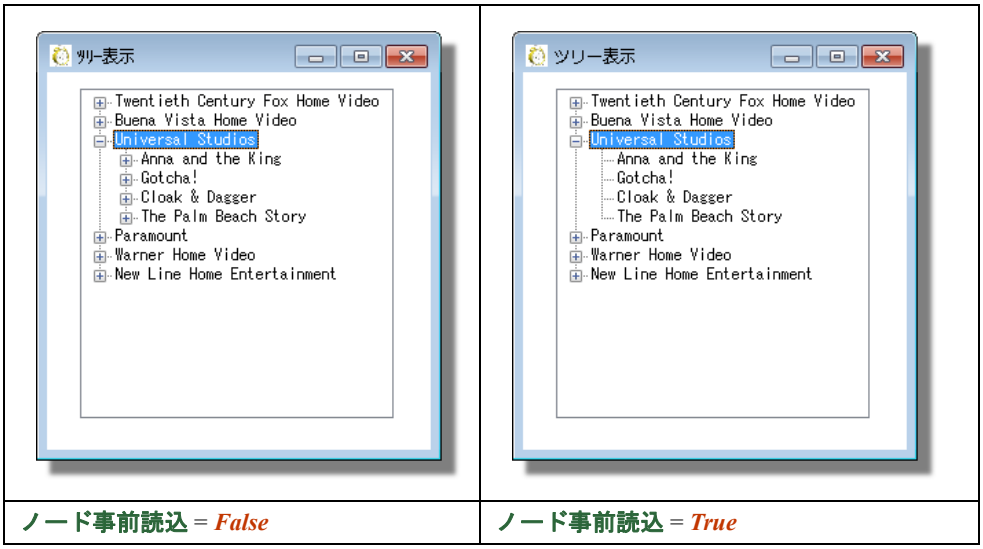
この例では、**自動展開**特性に以下の式を設定することでツリーの一番上のレベルだけを展開しています。

```
TreeLevel()=0
```

指定する式を以下のように変更することで、上位の２つのレベルのみを展開させることができます。

```
TreeLevel()=0 または TreeLevel()=1
```

子ノードを持つノードのみ展開ボタンを表示させるには



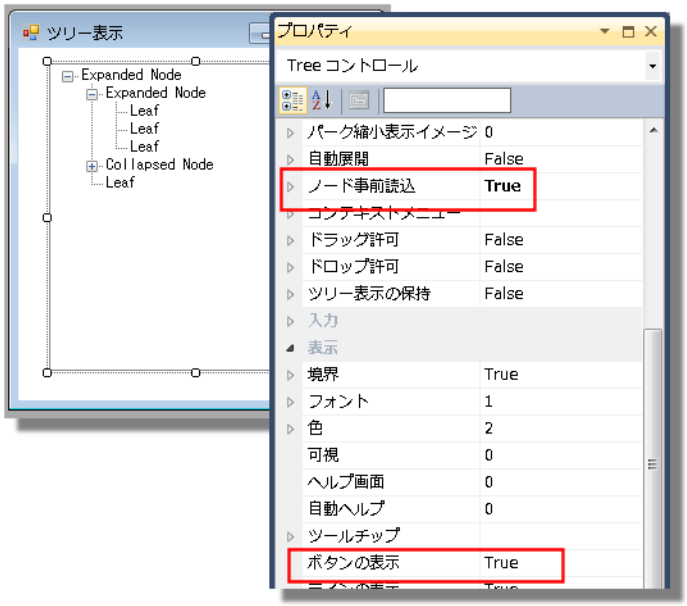
ノード事前読込 = *False*	ノード事前読込 = *True*

デフォルトでは、ツリーのノードには＋や－のボタンが表示されます。＋をクリックするとノードが展開され、－をクリックすると縮小されます。最初にツリーが表示された時点では、すべてのノードに＋が表示されます。これは、Magic エンジンが子ノードに該当するレコードをまだ読み込んでいないため、子ノードが存在するかどうかを認識できないためです。ノードをクリックして、子ノードが存在しないことを確認すると＋は表示されなくなります。

Magic エンジンに対して、事前にツリー内のすべてのレコードを事前に読み込むように指定することで、このような動作を変更することができます。これによって、リーフノードは展開ボタンが表示されなくなります。

事前読込を有効にする

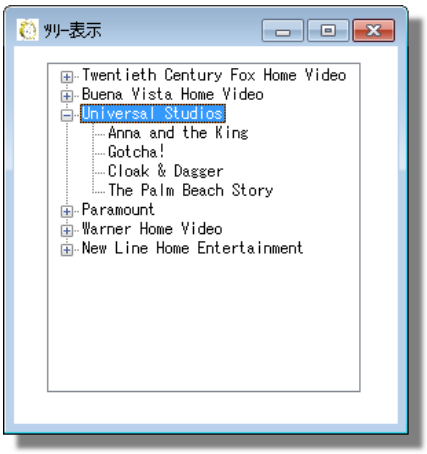
1. ツリーコントロールを選択します。
2. コントロール特性の**ノード事前読込**特性を選択し、*True* を選定します。
   これによって、子ノードの存在に応じて展開 / 縮小のボタンが適切に表示されます。



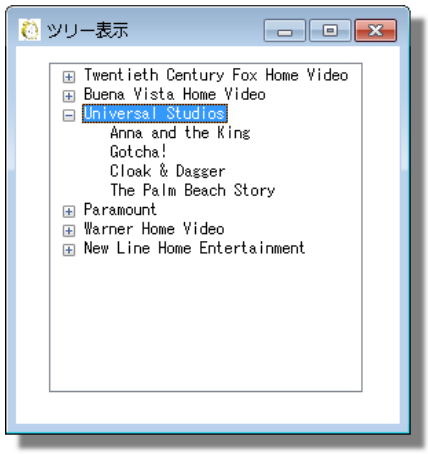
注： レコード数が非常に多い場合、**ノード事前読込**特性を *True* に設定すると、ツリーの読込に時間がかかる場合があります。

参照： 「展開 / 縮小ボタンを表示 / 非表示するには」（205 ページ）

ツリーコントロールの接続線を表示 / 非表示させるには



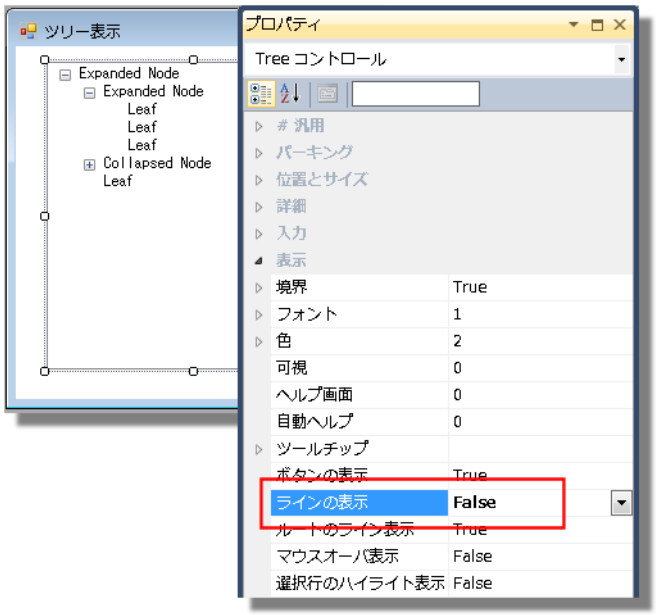
ラインの表示 = *True*



ラインの表示 = *False*

デフォルトでは、ツリー内のノードは点線で接続されて表示しています。**ラインの表示**特性を使用することで、この点線の表示 / 非表示を切り替えることができます。

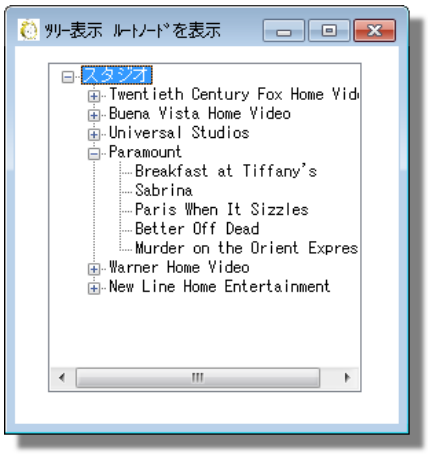
接続線の表示を切り替える



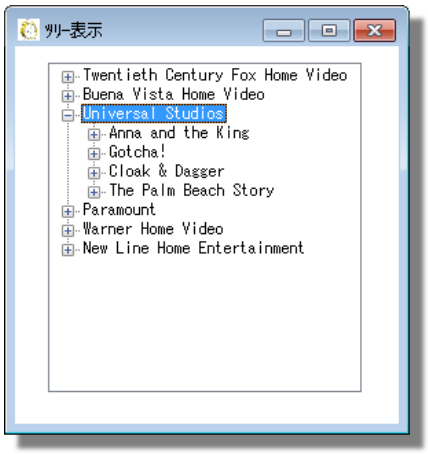
1. ツリーコントロールを選択します。
2. コントロール特性の**ライン表示**特性を選択します。接続線を表示させたい場合は *True* を選定し、表示しない場合は *False* を設定します。

   特性値を変更するとツリー表示に即反映されます。

ルートノードの表示 / 非表示を切り替えるには

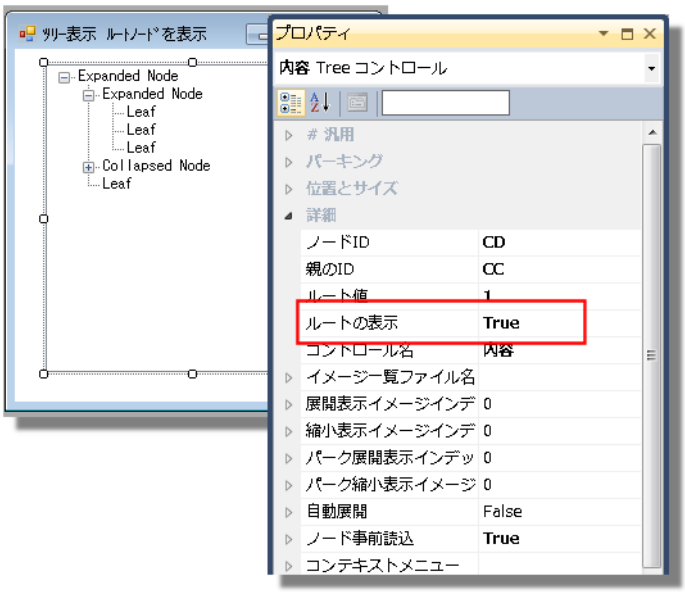


ルート表示 = *True*



ルート表示 = *False*

ルートノードの表示 / 非表示を切り替える

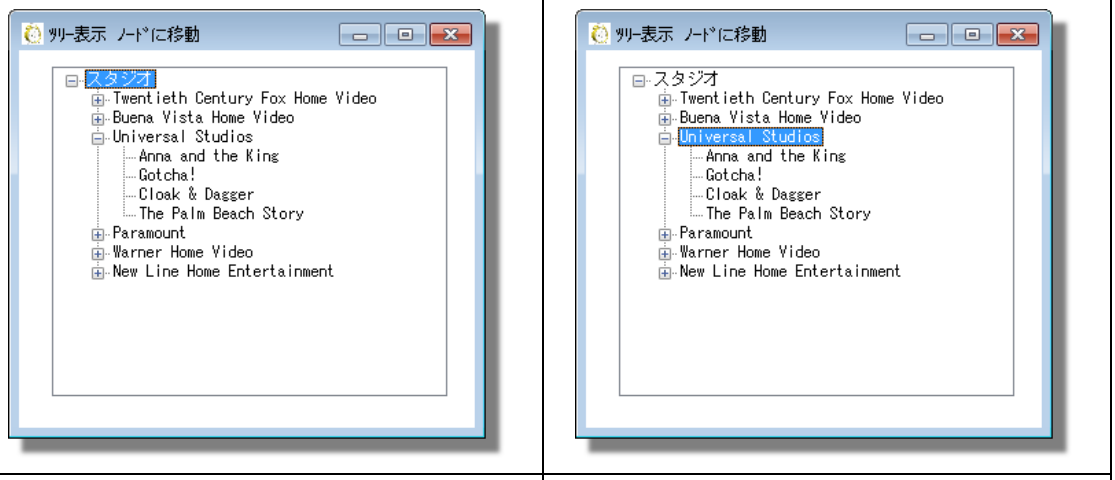


1. ツリーコントロールを選択します。
2. コントロール特性の**ルートの表示**特性を選択します。ルートを表示させたい場合は *True* を選定し、表示しない場合は *False* を設定します。

   変更内容は、実行時に反映されます。

注： 表示の有無にかかわらずルートノード用のレコードは必要です。ルートが存在しない場合は、エラーメッセージが表示されます。

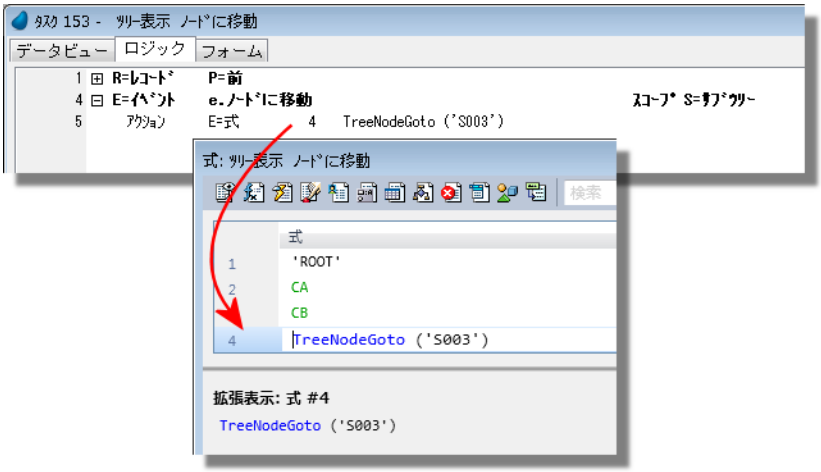
指定ノードに移動するには



カーソルのデフォルトの位置	カーソルが指定されたノードに移動

デフォルトでは、左図に示されているように、ツリー開くとカーソルを一番先頭のノード上にパークします。しかし、指定されたノードにカーソルを移動させることもできます。これは、**TreeNodeGoto()** 関数を使用することで実現できます。

TreeNodeGoto() 関数を使用する



1. ロジックエディタを開き、ツリーの展開処理を行うヘッダ行を作成し、必要なイベントを定義します。
2. アクション処理コマンドを定義し、以下の式を設定します。

TreeNodeGoto (*Node*)

パラメータ：

Node：移動先のノード ID を表す値です。この例では、Universal Studio は S003 の ID が設定されており、このノードに移動するように設定されています。指定されたノードがどのレベルに存在しているかは、知る必要がありません。

ヒント :タスク前で TreeNodeGoto() 関数を使用することで、指定したノード上にパークさせることができます。レコード前でも使用できますが、IsFirstRecordCycle (0) またはウェイト =No の条件を設定する必要があります。条件を指定しないとループする可能性があります。

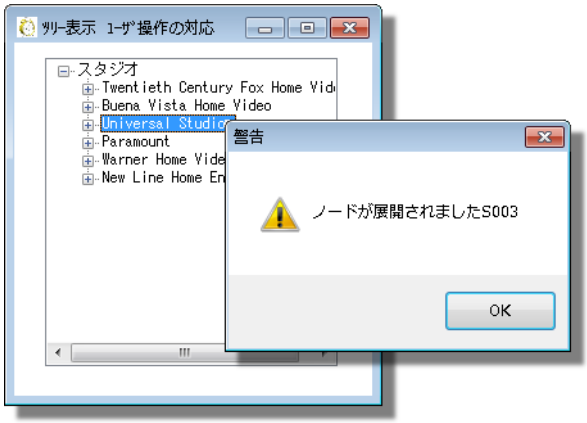
参照： 「実行時にツリーノードを明示的に展開 / 縮小させるには」（208 ページ）

エンドユーザが行ったノード展開 / 縮小の操作に対応する処理を実行させるには

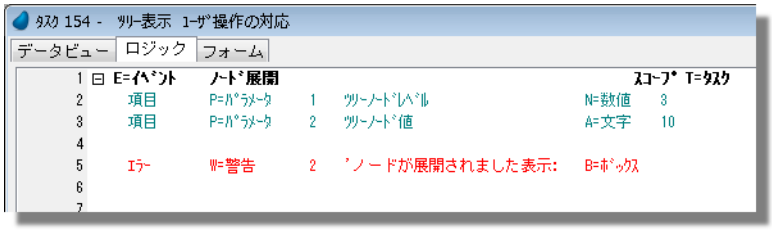
ノードが展開されたり縮小された場合に、何らかの処理を実行させる必要が発生します。例えば、画面に表示されている小計を更新する処理などが考えられます。また、実行中に子ノードを作成する必要があるかもしれません。

ツリーを定義するためにメモリテーブルを使用することで、各ノード上にカスタマイズされたデータを表示することができます。

これらの処理を実現するために、ノード展開 / ノード縮小イベントを使用してロジックユニットを作成します。

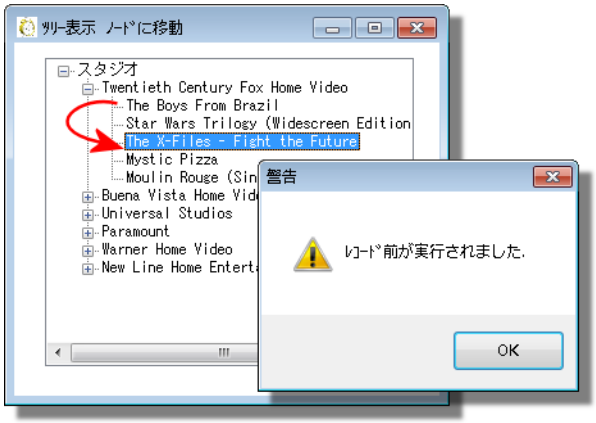


ノード展開 / 縮小イベントを取り込む



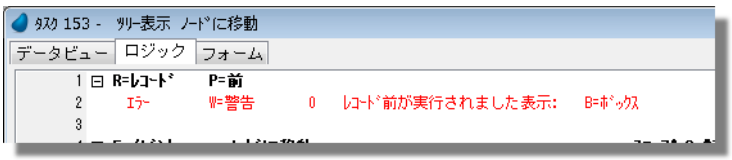
1. *Ctrl+H* を押下してロジックユニットのヘッダ行を作成します。
2. *E* を入力して**イベント**を選択し、内部イベントのノード展開またはノード縮小を選択します。
3. ダイアログボックスで「イベントに対応したパラメータを作成しますか？」というメッセージが表示されます。**Tree Node Level** と **Tree Node Value** の２つのパラメータ項目を作成する場合は、はいをクリックします。
4. これで、ユーザがノードを展開したり縮小した場合、実行するロジックユニットが定義できました。ロジックユニット内に定義されたパラメータ項目を使用してノードレベルと値を確認することができます。
5. イベント特性で、**伝播**特性を *Yes* に設定します。設定しないと、このロジックユニット内でイベントは止められ、ノードの展開 / 縮小が行われません。

ユーザのノードから別のノードに移動する操作に対応するには



ユーザが、あるノードから別のノードにフォーカスを移した場合に、何らかの処理を実行させたい場合があります。ツリー内の各ノードが同じデータソースのレコードであれば、容易に実現できます。通常のレコードイベント（**レコード前**や**レコード後**）が、レコードの移動の際に実行されるからです。レコードが異なるフォーマットで表示されているにすぎません。

ノードに入る動作を取り込む

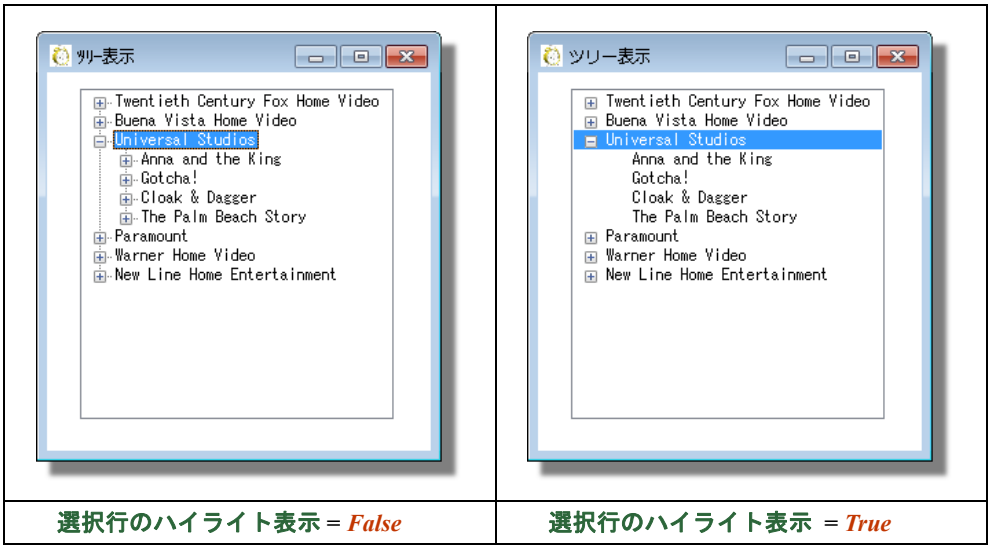


カーソルがノードに入る前にその動作を取り込むには、**レコード前**に処理コマンドを定義する必要があります。

ノードから抜け出る動作を取り込む

データビューが変更された場合のみ、**レコード後**が実行されます。レコードが変更されなくても、ノードから抜け出る動作を取り込む必要がある場合は、**タスク特性**の**強制レコード後**特性（**動作**タブ）を *Yes* に設定します。

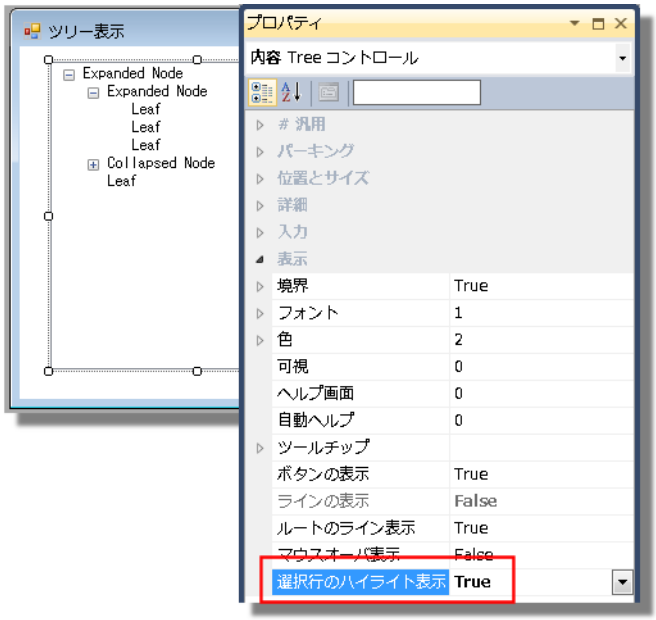
現在のツリーノードの行を強調表示させるには



選択行のハイライト表示 = *False*	選択行のハイライト表示 = *True*

ツリー内でノードを選択すると、そのノードは強調表示されます。デフォルトでは、ノードテキストのみが強調表示されます。しかし、ツリー全体を横切る強調行として表示させることもできます。これは、**選択行のハイライト表示**特性によって制御することができます。

行ハイライトの指定を切り替える



1. ツリーコントロールを選択します。
2. コントロール特性の**選択行のハイライト表示**特性を選択し、強調行を表示させる場合は *True* を設定します。表示させない場合は、*False* を設定します。

   特性値を変更すると動作に即反映されます。

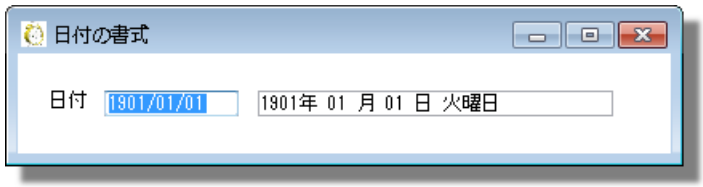
注： **ラインの表示**特性が *False* の場合のみこの特性は有効になります。

［このページは意図的に空白にしています。］

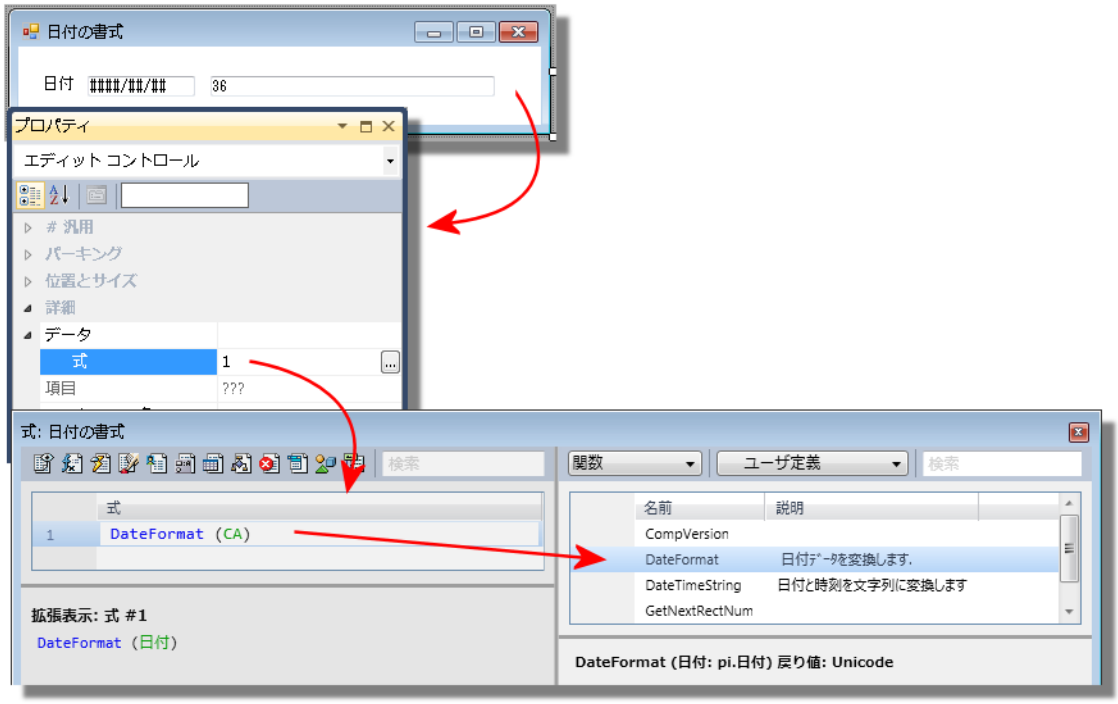
第 10 章 : さらなるロジックの拡張

ここでは、比較的最近追加された Magic xpa の機能を利用したロジック例を紹介しています。

ユーザ定義関数を作成するには



ユーザ定義関数は特殊なタイプの**ロジックユニット**として扱われます。**ユーザ定義関数**には、他の**ロジックユニット**で利用されているものと同じパラメータ項目や変数項目、および処理コマンドを使用します。主な違いの 1 つに、戻り値があるということです。関数が式やフォームで使用される場合、関数から返る値を（いったんデータ項目に格納することなく）直接使用することができます。



上図の例では、帳票用として複雑な書式で定義された文字列として日付データを変換する関数について示されています。変換された文字列は、式の中でこの関数を使用することで直接フォームに定義することとができます。

ユーザ定義関数は、Magic xpa に組み込まれている関数と同じように**関数一覧**に表示されます。これにより、通常の内部関数と全く同じように利用することができます。これはまた、内部関数と同じ名前の**ユーザ定義関数**を作成した場合、内部関数をオーバライドすることになります。このような関数の再帰的な利用はできないため、関数名の定義方法については注意する必要があります。

以下、**ユーザ定義関数**の作成方法について説明しています。

関数の有効範囲を定義する

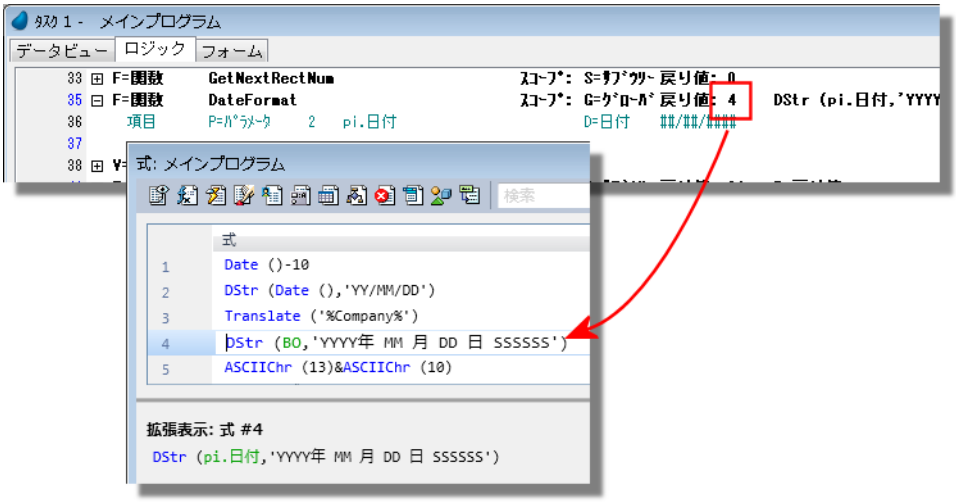
ユーザ定義関数の有効範囲は、それが定義されていル場所と関数の**スコープ**特性の値に依存します。

- 関数があるタスク内に定義されており、**スコープ**特性が ***T= タスク***と設定された場合、その関数はそのタスク内でのみ参照できます。

- 関数が親タスク内で定義されており、**スコープ**特性が ***S= サブタスク***と設定された場合、その関数はすべてのサブタスク内で参照できます。従って、**メインプログラム**内で作成された関数は、アプリケーション全体で参照することができます。
- 関数がコンポーネントの一部の場合、複数のプロジェクト間で共有することができます。

参照： 「現在のタスクでのみ有効な関数を作成するには」（223 ページ）
「プロジェクト全体で有効な関数を作成するには」（224 ページ）

ユーザ定義関数を作成する



1. タスクの**ロジックエディタ**を開きます。
2. ***Ctrl+H*** を押下してヘッダ行を作成します。
3. ***F*** を入力して***関数***を選択します。カーソルが右側に移動し、関数名の入力カラムに入ります。
4. この関数の名前を入力します。どのような名前も使用できますが、Magic xpa の内部関数と同じ名前を指定しないようにしてください。
5. 必要であれば、入出力用のパラメータ項目を定義します（詳細は、「ユーザ定義関数のパラメータを設定するには」（221 ページ）を参照してください）。
6. **戻り値**を定義します（詳細は、「ユーザ定義関数の戻り値を設定するには」（222 ページ）を参照してください）。

これで、**ユーザ定義関数**が作成されました。この関数は、内部関数と一緒に**関数一覧**に表示されるようになります。

ユーザ定義関数のパラメータを設定するには

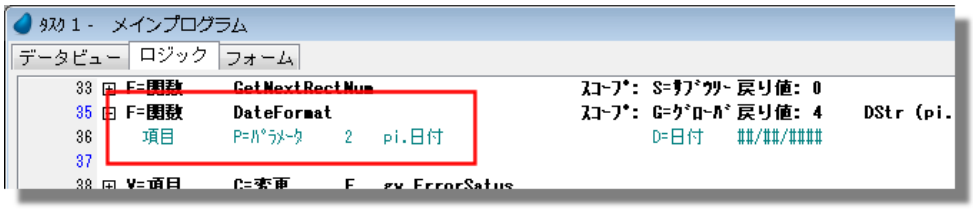
ユーザ定義関数のパラメータはタスクやイベントロジックユニットに定義されたパラメータと同じように動作します。これらは、パラメータとして定義された項目です。

関数用にパラメータを作成するには、必要なパラメータ項目を適切な順番で作成します。パラメータを作成すると、関数一覧にはデータ型に対応する文字でパラメータが表示されます。

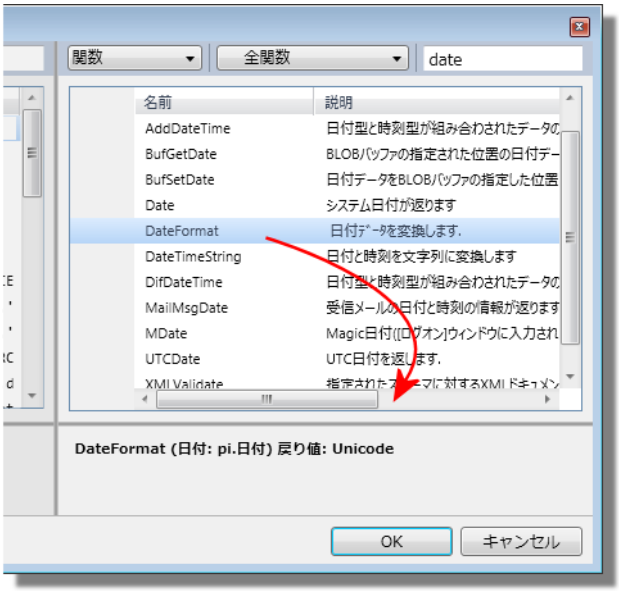
関数で使用されるパラメータは、入力パラメータです。関数が式で使用されるため、データは常に参照型で渡され、関数によって変更することはできません。データを返したい場合は、**戻り値**特性を使用する必要があります（「ユーザ定義関数の戻り値を設定するには」（222 ページ）を参照してください）。

ヒント : *パラメータ項目として定義しない（変数項目のまま）場合も同じように動作しますが、関数一覧にパラメータは表示されません。*

関数のパラメータを定義する



1. 関数ロジックユニットに移動します。
2. *F4*（編集→行作成）を押下します。空白行がカーソル位置の下に作成され、カーソルは追加された行の左側の入力カラムに移動します。
3. *V* を入力します。***項目***という文字が表示されカーソルは右側に移動します。
4. *P* を入力します。***パラメータ***という文字が表示され、カーソルが右側に移動します。
5. パラメータの名前を入力し、カーソルを右側に移動します。
6. この項目の**モデル**を選択するか、**型**や**書式**などの特性を設定します。
7. 複数のパラメータがある場合は、上記と同じような操作を繰り返して作成します。



これで、作成した関数はパラメータを受け取ることができるようになります。関数一覧の作成した関数名にパークすると、定義されたパラメータのデータ型やパラメータ名が下に表示されます。

注 : Magic xpa Ver3 より、関数一覧は、式ウィンドウの右側に表示されるようになりました。

ユーザ定義関数の戻り値を設定するには

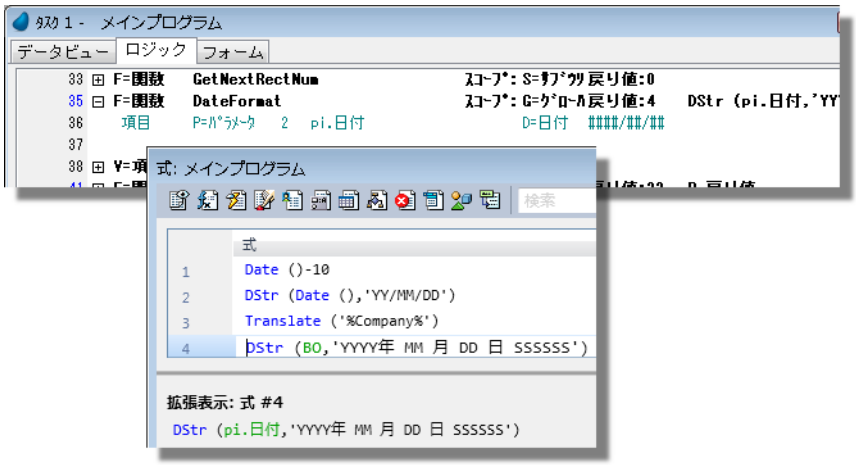
ユーザ定義関数の主な利点の一つとして、戻り値を定義することができることです。戻り値は、変数項目に一時的に格納することなく、フォーム上やパラメータで直接使用することができます。これにより、プログラミングの効率化が高まります。

また、関数は常に式の中で使用されるため、入力パラメータは入力のみに使用されます。関数内でこのパラメータの値を変更しても、起動元には返りません。データを返したい場合は、戻り値を使用します。

戻り値の使用は任意です。必ずしも値を返す必要があるわけではありません。

ユーザ定義関数では、どのようなデータ型の戻り値も返すことができます。正しく関数を使用するか否かは開発者に委ねられています。しかし、例えば、数値項目に文字項目を戻すように設定した場合、**構文チェック**ユーティリティを実行するとエラーを返します。

ユーザ定義関数に戻り値を設定する



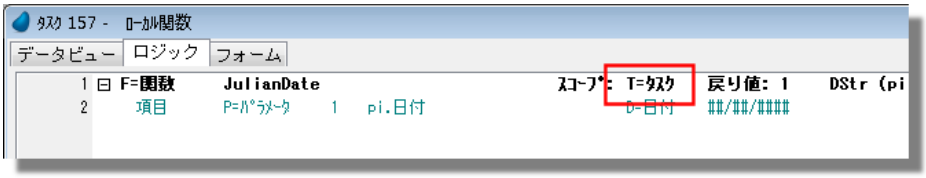
1. **関数**ロジックユニットのヘッダ行に移動します。
2. **戻り値**特性に移動します。
3. ズーム（*F5*、またはダブルクリック）して**式エディタ**を開きます。
4. 戻り値として返したい値として評価される式を入力します。この例では、日付を文字列に変換するために **Trim()** と **Dstr()** の２つの関数を使用しています。

これで、関数を実行すると、式で指定された値が返ります。

現在のタスクでのみ有効な関数を作成するには

作成された関数の有効範囲は**スコープ**特性で決まります。**スコープ**特性が ***T= タスク***と設定されている場合、関数はこのタスクでのみ参照できます。

ローカル関数を作成する



1. 関数のヘッダ行に移動します。
2. **スコープ**特性に移動します。
3. ***T= タスク***を選択します。

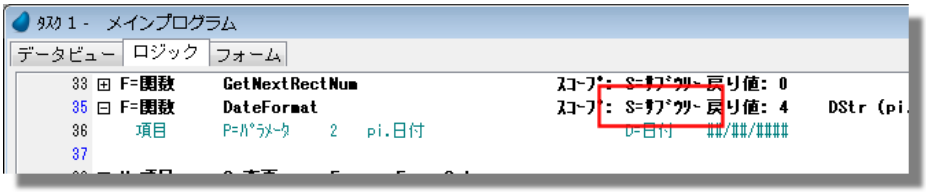
これでこの関数は、このタスク内でのみ有効になります。

参照： 関数を作成する基本的な作業方法については、「ユーザ定義関数を作成するには」（219 ページ）を参照してください。

プロジェクト全体で有効な関数を作成するには

作成された関数の有効範囲は**スコープ**特性で決まります。**スコープ**特性が ***S= サブタスク***と設定されている場合、関数はこのタスクとサブタスクで参照できます。これは、**スコープ**特性を ***S= サブタスク***と設定された**ユーザ定義関数**が**メインプログラム**に定義された場合、この関数はプロジェクト全体で使用することができることを意味しています。

グローバル関数を作成する



1. **メインプログラム**でユーザ定義関数を作成します。
2. **スコープ**特性に移動します。
3. ***S= サブツリー***を選択します。

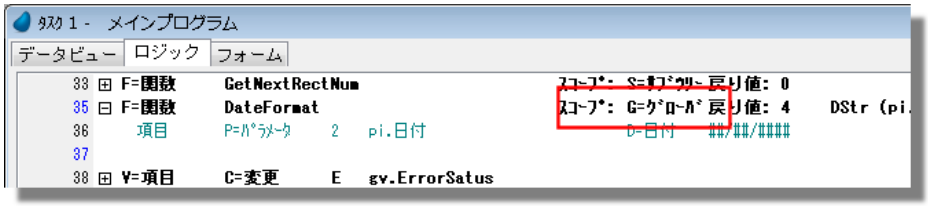
これでこの関数は、プロジェクト全体で有効になります。

参照： 関数を作成する基本的な作業方法については、「ユーザ定義関数を作成するには」（219 ページ）を参照してください。

複数のプロジェクト間で関数を共有するには

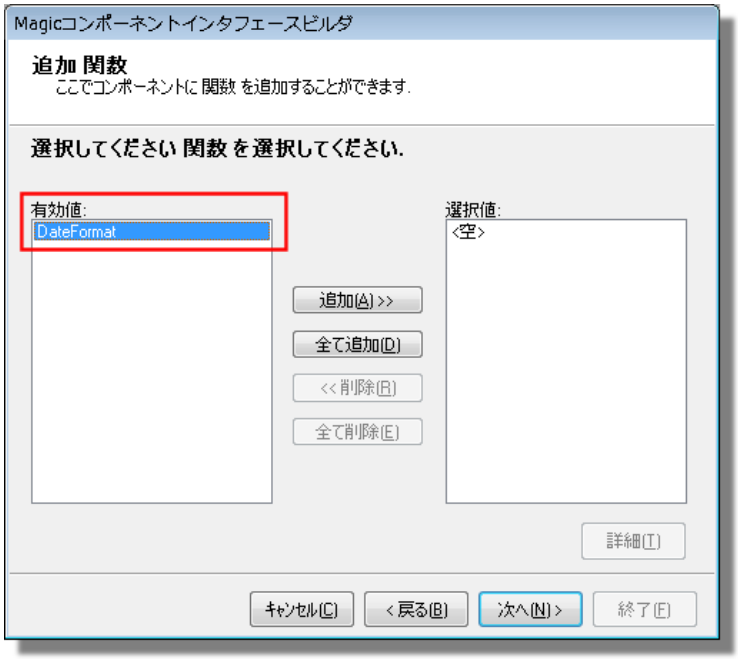
作成された関数の有効範囲は**スコープ**特性で決まります。**スコープ**特性が ***G= グローバル***と設定されている場合、関数はコンポーネントとして公開することができます。これは、**スコープ**特性を ***G= グルーバル***と設定された**ユーザ定義関数**が**メインプログラム**に定義された場合、この関数は他のプロジェクトでも使用することができることを意味しています。

グローバル関数を作成する



1. **メインプログラム**でユーザ定義関数を作成します。
2. **スコープ**特性に移動します。
3. ***G= グローバル***を選択します。

これで、以下に示すように、コンポーネントの作成時に、Magic コンポーネントインタフェースビルダで選択できるようになります。

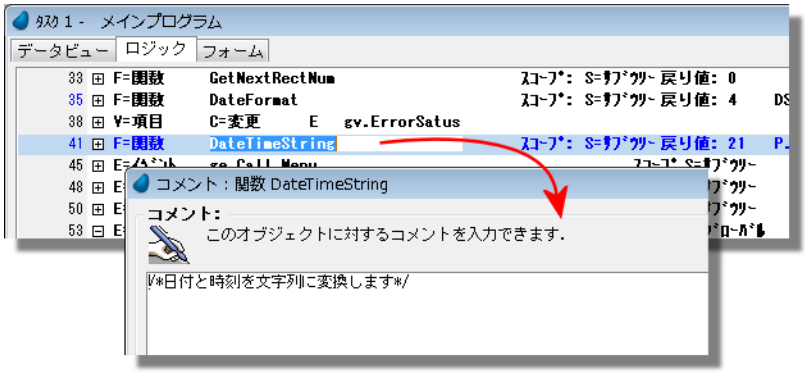


参照： 第 15 章：「プロジェクト間で Magic xpa のオブジェクトを再利用するには」（357 ページ）

ユーザ定義関数のヘルプを定義するには

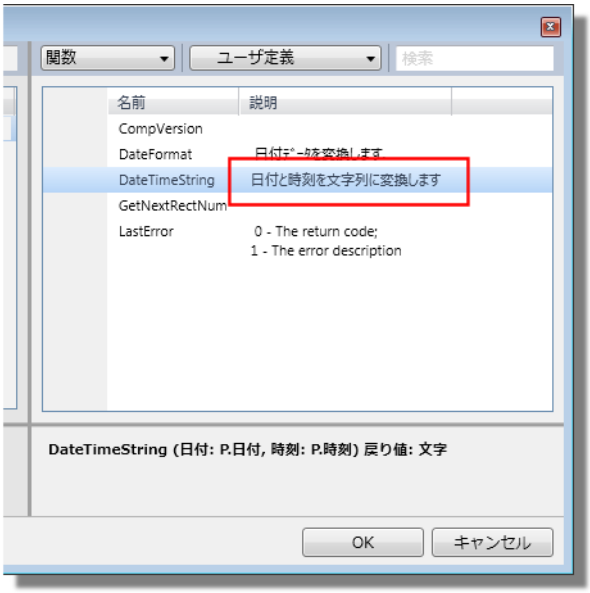
作成された関数のヘルプテキストを**関数一覧**で表示させることができます。**関数**ロジックユニットに**コメント**を追加することで表示されます。このようにすることで、この関数の意味や利用方法などを利用者に正確に伝えることができます。

ユーザ定義関数にコメントを定義する



1. 任意のタスクで**ユーザ定義関数**を作成します（**メインプログラム**で作成することで、プロジェクト内で有効な関数になります）。
2. ***F12***（**オプション→コメント入力**）を押下して**コメント**ダイアログを開きます。
3. この関数の内容に関するテキストを入力します。テキストは、以下のように「/*」と「*/」で囲みます。
   /* 日付と時刻を文字列に変換します。*/

これで、**関数一覧**でこの関数にカーソルが位置付けらると、**説明**カラムに入力したヘルプテキストが表示されます。



一連の処理を繰り返し実行させるには

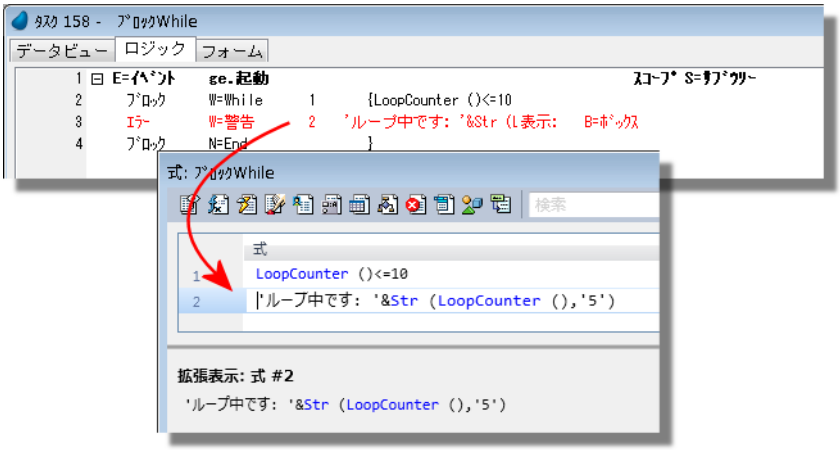
一連の処理コマンドを何度も繰り返して実行させたい場合、一般的なプログラミング言語ではある種類のループメカニズムを使用しています。Magic xpa におけるループメカニズムは、**ブロック While** 処理コマンドで実現できます。これは、どの**ロジックユニット**にも定義することができます。

ブロック While 処理コマンドは、指定された条件が True の場合、ブロック内の処理を実行します。この例では、処理を 10 回実行するように定義しています。

LoopCounter() 関数を使用する

ブロック While 処理コマンドでは、**LoopCounter()** と呼ばれる独自の関数を使用することができます。この関数は、処理が繰り返された回数を戻すため、ループ処理のための特別なカウンタを作成する必要がありません。

ブロック While 処理コマンドを定義する



1. 使用したい**ロジックユニット**に移動します。
2. ***F4***（**編集→行作成**）を押下します。空白行が追加され、カーソルは追加行の左側の入力カラムに移動します。
3. ***B*** を入力します。**ブロック**処理コマンドが選択され、***ブロック If*** と ***ブロック End*** の 2 つの行が表示されます。カーソルは、***If*** の位置に位置付けられます。
4. ***W*** を入力します。***ブロック While*** が選択されます。
5. 右に ***Tab*** 移動し、**ズーム**（***F5***、または**ダブルクリック**）します。**式エディタ**が開きます。
6. **式エディタ**には、ループを抜ける場合に False が返る条件式を設定します。この例では、以下の式を設定しています。

   ```
   Loopcounter () <= 10
   ```

   繰返し番号が 11 になると False になるため、ループは 10 回実行されることになります。

データ項目を更新するには

Magic xpa でデータ項目を更新するには、いくつかの方法があります。

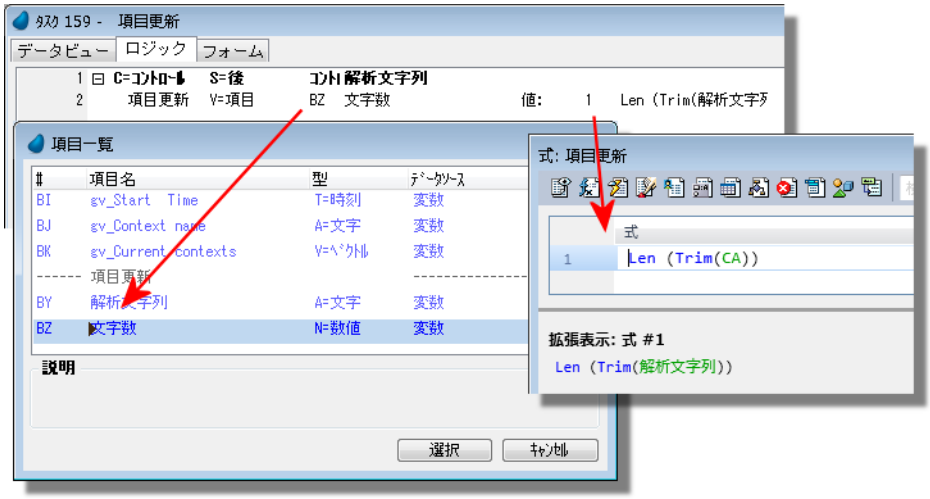
- デフォルトの値を、**項目モデル**やデータソースまたはタスク内での変数定義であらかじめ割り当てる。
- タスク内でデータビューを定義する際に、**代入**特性を使用する。
- フォーム上の項目の場合、ユーザによって入力することができます。
- **項目更新**処理コマンドの実行する。
- **VarSet()** 関数のような関数を使用する。
- **フォーム入力**処理コマンドによってデータ入力をする。

項目更新処理コマンドは、**ロジックユニット**で項目の値を更新する通常の方法です。**ロジックユニット**内で実行される範囲では、**項目更新**処理コマンドは手続型です。

オンラインプログラムでは、**代入**特性の方が**項目更新**処理コマンドよりよく使用されます。**代入**特性は非手続き型のため、代入式で使用されている項目の値が変更されると自動的に代入値も更新されます。このような動作によって、いつ更新すべきか気にする必要がなくなります。

この例では、文字型項目内の文字数を**文字数**項目に格納しています。**コントロール後**ロジックユニットに更新処理が定義されているため、ユーザが入力欄から抜けるとすぐに文字数項目が更新されます。

項目更新処理コマンドを使用する

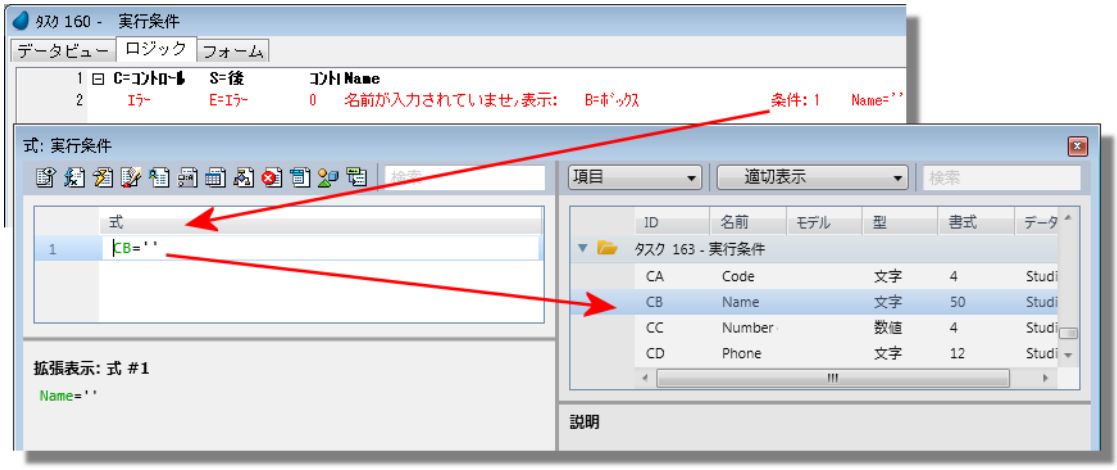


1. 使用したい**ロジックユニット**に移動します。
2. *F4*（**編集→行作成**）を押下します。空白行が追加され、カーソルは追加行の左側の入力カラムに移動します。
3. *U* を入力します。**項目更新**処理コマンドが選択され、カーソルは右に移動します。
4. **ズーム**（*F5*、または**ダブルクリック**）して**項目一覧**を開きます。更新したい項目を選択し *Enter* を押下するか選択ボタンを**クリック**します。一覧から選択しないで項目番号を直接入力することもできます。
5. 右側に *Tab* 移動して**値**カラムにカーソルを置きます。
6. **値**カラムで**ズーム**（*F5*、または**ダブルクリック**）して**式エディタ**を開きます。
7. 更新する値として評価される式を入力します。他の項目の値で更新するだけであれば、その項目の項目番号を指定します。カウンタのリセット処理を行うのであれば、0 という数値を入力することもあります。また、上記の例のような文字列の文字数を返す式なども指定できます。
8. *Enter* を押下するか、OK を**クリック**することで**値**カラムに戻ります。

参照： 第 19 章：「タスクのデータ項目の値を設定するには」（445 ページ）

項目の値を処理コマンドの実行条件に設定するには

Magic xpa には処理コマンドの実行を制御するための様々な方法があります。基本的には、論理値を条件式として指定することで可能です。この例では、ユーザが入力項目を抜けた時に、入力項目が空白の場合にエラーメッセージを表示させるように式を指定しています。



処理コマンドの条件式を入力する

1. 条件の設定が必要な処理コマンドの行に移動します。この例では、**エラー**処理コマンドになります。
2. **条件**カラムに移動します。
3. **ズーム**（*F5*、または**ダブルクリック**）して、**式エディタ**を開きます。
4. *F4* を押下して、１行追加します。
5. 再び**ズーム**（*F5*、または**ダブルクリック**）して右側に表示されている**項目一覧**にカーソルを移動します。
   以下のようにして項目を選択します。
   - 選択したい項目にカーソルを移動して、*Enter* を押下するか、選択ボタンをクリックします。
   - 項目番号を直接入力します。（この例では、*CB*）
6. 必要に応じて、式を追加します。

処理コマンドを実行させる必要がある場合は、True になるように式を定義します。**条件**カラムは、If 文と同じと考えてください。条件が True の場合、処理コマンドは実行されます。この例では、*CB=''* と設定されています。これは、項目 B が空白の場合 True と判断され、エラーメッセージを表示するようになります。

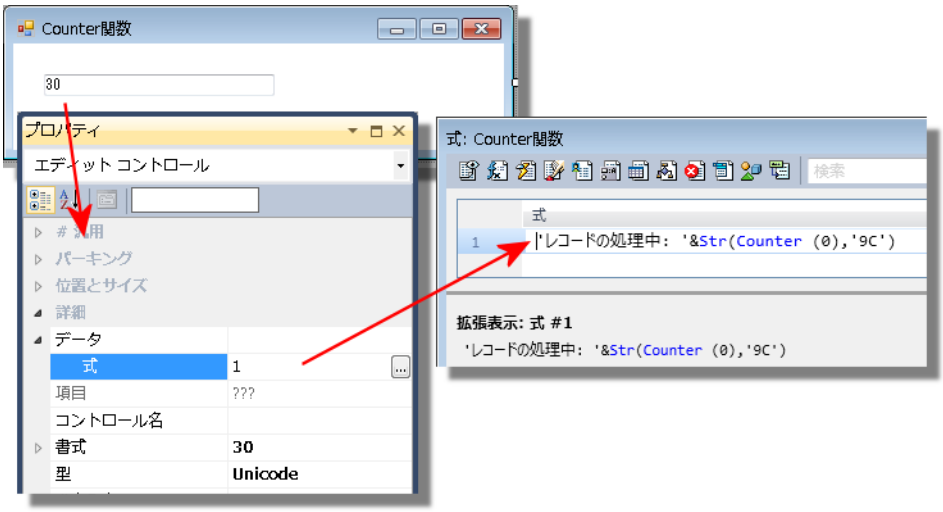
参照： 第 20 章：「式」（461 ページ）

バッチプログラムで処理されたレコードの連番を取得するには

バッチプログラムでは、処理中のレコード番号が必要な場合があります。ユーザに情報用メッセージを表示したり、連番を作成したり、またはデバッグ中に処理をキャンセルするためにこの番号を使用することがあります。

このような処理を実現するために、Magic xpa では **Counter()** 関数を使用します。この関数は、タスクの世代情報という 1 つのパラメータ指定します。**Counter(*0*)** は、現在のバッチタスクが処理しているレコード数を返し、**Counter(*1*)** は親タスクが処理しているレコード数を返します。

Counter() 関数を使用する



処理中のレコード数を取得するには、**式エディタ**で以下のように入力します。

```
Counter (0)
```

上の式は現在のタスクのレコード数を取得します。**Counter(*1*)** は親タスクのレコード数を返し、**Counter(*2*)** はその上位のタスクのレコード数が返ります。

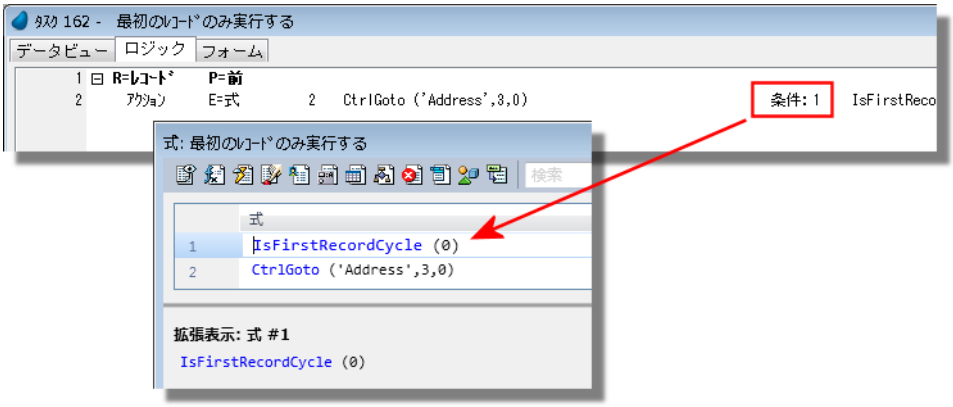
この例では、ユーザにメッセージを表示するために **Counter(*0*)** で返された値を使用しています。戻り値は数値のため、**Str()** 関数を使用して文字列に変換しています。

オンラインタスクで最初のレコードの場合のみ実行する条件を設定するには

オンラインタスクでは、ユーザが最初にタスクを起動した時点で、処理を実行させたい場合があります。タスク起動時は、データソースをオープンし、データの初期設定を行うため、このような処理を実行させる適切な**ロジックユニット**は**レコード前**になります。しかし、ユーザが表形式でデータを参照している場合、ユーザがレコード間を移動させるたびに処理が実行される必要はないかもしれません。

このような場合、**IsFirstRecordCycle()** 関数を使用して対応できます。この関数は、**レコード前**が最初に実行されたときのみ True を返します。

IsFirstRecordCycle() 関数を使用する



1. **IsFirstRecordCycle()** 関数を使用したい場所で式を定義します。
2. 以下のように入力します。
   ```
   IsFirstRecordCycle(0)
   ```

   上の式は、現在のタスクの最初のレコードサイクルを確認するものです。親タスクを確認は、**IsFirstRecordCycle(*1*)** を使用します。

この例では、タスクが起動された時点でのみ **CtrlGoTo** 関数が実行されるようになっています。

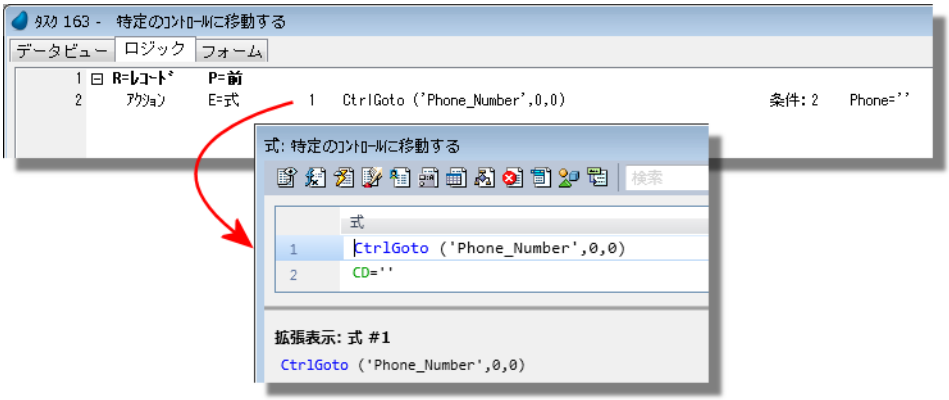
指定コントロールにカーソルを移動させるには

カーソルを特定のコントロールに移動させたい場合があります。例えば、エラーメッセージが表示された時に、エラー対応する表示項目に移動するようにしたい場合があるかもしれません。

CtrlGoto() 関数を使用することでカーソルを指定したコントロールに移動させることができます。

CtrlGoto('Control name', row, generation)	
'Control name'	移動先のコントロールのコントロール名。
row	テーブル上のデータの場合、行数を指定します。
generation	タスクの世代番号。現在のタスクが 0、親タスクが 1 になります。

CtrlGoto() 関数を使用する



1. カーソルの移動処理を実行させたいロジックユニットに移動します。この場合、ユーザが最初にレコードに入った時点で、カーソルを電話番号に移動させたいため、レコード前に定義しています。
2. *F4* を押下して 1 行追加します。
3. *A* を入力してアクション処理コマンドを指定します。カーソルは式カラムに移動します。
4. ズーム（*F5*、またはダブルクリック）して式エディタを開きます。
5. *F4* を押下して 1 行追加します。以下の関数名を入力します。

   ```
   CtrlGoto(
   ```

   Ct と入力することでオートコンプリート機能が動作します。
6. コンテキストメニューを開きコントロールを選択することでコントロール一覧が表示されます。ここから移動したいコントロール名を選択できます。直接コントロール名を入力することもできます。
7. ほとんどは、現在のタスクのコントロールへの移動で、テーブル内のコントロールでもないので、以降の 2 つのパラメータは 0,0 と入力するだけになります。

   しかし、テーブルを使用している場合、2 番目のパラメータは移動先の行番号となります。例えば、**CtrlGoto(*'Phone_Number',3,0*)** では、カーソルをテーブル内の 3 行目の Phone_Number という名前のコントロールに移動します。

   親タスクのコントロールに移動するのであれば、3 番目のパラメータを 1 に、その上のタスクに移動するのであれば 2 を指定します。**CtrlGoto(*'Phone_Number',0,1*)** は、カーソルを親タスクの Phone_Number という名前のコントロールに移動します。

コントロールに入ったり抜けたりした場合に実行するロジックを作成するには

ユーザが項目に入ったり、抜けた時に、処理を実行させたい場合がります。例えば、ユーザが項目に入った時点で何らかの有益な情報を表示させたり、自動的に選択一覧を表示させたりすることなどが考えられます。ユーザが項目を抜けた時点で、何らかの計算処理を実行したり、ユーザの入力内容を検証するような処理も必要になります。

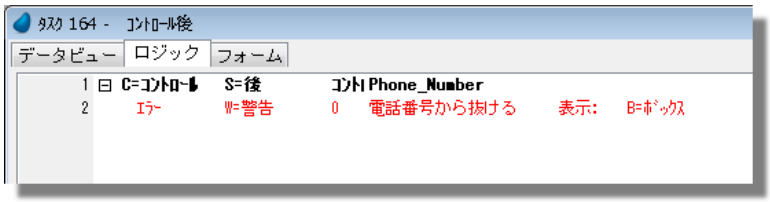
この種のロジックは、コントロール前とコントロール後のロジックユニットに定義されます。

コントロール前を使用する

1. *Ctrl+H* を押下してロジックユニットのヘッダ行を作成します。
2. C を入力して***コントロール***を選択します。次のカラムに移動します。
3. ドロップダウンリストから *P* を入力して、***前***を選択します。次のカラムに移動します。
4. ズーム（*F5*、またはダブルクリック）してコントロール一覧から対象となるコントロールを選択します。コントロールがまだ定義されていなかったり、（コンポーネントを使用するなどで）参照できない状態の場合、コントロール名を入力することもできます。
5. 作成されたロジックユニット内で実行したい処理コマンドを入力します。

これで、ユーザが指定されたコントロールに入った場合、定義された処理が実行されます。

コントロール後を使用する



1. *Ctrl+H* を押下してロジックユニットのヘッダ行を作成します。
2. *C* を入力して***コントロール***を選択します。次のカラムに移動します。
3. ドロップダウンリスから S を入力して、***後***を選択します。次のカラムに移動します。
4. ズーム（*F5*、またはダブルクリック）してコントロール一覧から対象となるコントロールを選択します。コントロールがまだ定義されていなかったり、（コンポーネントを使用するなどで）参照できない状態の場合、コントロール名を入力することもできます。
5. 作成されたロジックユニット内で実行したい処理コマンドを入力します。

これで、ユーザが指定されたコントロールを抜けた場合、定義された処理が実行されます。

注：　コントロール後とコントロール検証は同じような動作に見えますが、実は全く異なります。ユーザがコントロールを抜けた場合、コントロール後は必ず実行されますが、コントロール検証はコントロールにパークしたか否かにかかわらず、ユーザが項目を通過した時点で実行されます。例えば、レコードが保存される前に、項目が空白のままになっているかどうかを確認する場合は、コントロール検証を使用します。

カーソルを一定の方向に移動させた場合のみ処理を実行させるには

ユーザが Windows 環境でどこでもクリックする傾向がある場合、画面上でカーソルを上下に移動してもそれに関係なく、項目に対応する処理を実行させる必要があります。しかし、キーボード操作が中心のアプリケーションとして、カーソルの移動方向にもとづいて、処理が実行されるかどうかを指定することができます。**Flow()** 関数を使用することでこのような処理が可能です。

Flow() 関数のパラメータ	方向	モード
N	順方向	ステップモード
F	順方向	高速モード
P	逆方向	ステップモード
R	逆方向	高速モード
S	選択終了	
C	取消終了	

ユーザがエディタ上の前方（上）から項目に入った場合、**Flow(*'N'*)** は True を返します。後方（下）から入った場合、**Flow(*'P'*)** は True を返します。

順番にカーソルを移動したり、特定のコントロールをスキップした場合のみ処理を実行させるには

カーソルを動かしたり、クリックすることによって、特定のコントロールにフォーカスを置くとすぐに実行する必要のある処理を定義することができます。これは、そのコントロールのコントロール前に処理コマンドを定義することで可能となります。

コントロール前を使用する

1. *Ctrl+H* を押下してロジックユニットのヘッダ行を作成します。
2. *C* を入力して***コントロール***を選択します。次のカラムに移動します。
3. ドロップダウンリスから *P* を入力して、***前***を選択します。次のカラムに移動します。
4. ズーム（*F5*、またはダブルクリック）してコントロール一覧から対象となるコントロールを選択します。コントロールがまだ定義されていなかったり、（コンポーネントを使用するなどで）参照できない状態の場合、コントロール名を入力することもできます。
5. 作成されたロジックユニット内で実行したい処理コマンドを入力します。

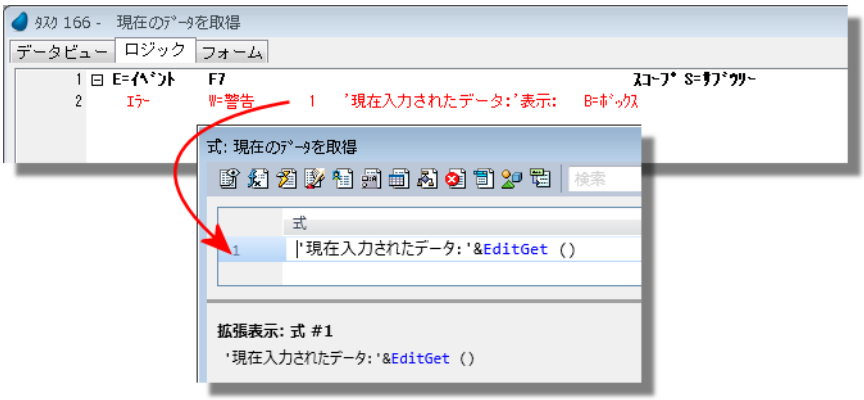
これで、ユーザが指定されたコントロールに最初にフォーカスを移した場合のみ、定義された処理が実行されます。

コントロール上（エディット / リッチエディット / 複数選択リストボックス）にフォーカスが残っている状態で新規入力データを検索するには

ユーザがデータをフィールドに入力した場合、ユーザがフィールドを抜けるまで、変更されたフィールドはデータ項目に保存されず、ハンドラでも処理できません。ユーザがフィールドを抜けた後、コントロール後で入力内容の検証が通常必要になります。

しかし、ユーザがフィールドを抜ける前に現在入力されたデータ内容を検索したい場合もあります。例えば、フィールド上にフォーカスがある状態で、計算処理など該当するフィールドと関連する何らかの処理を実行するために、ショートカットキーを押下するような場合がこれにあたります。

EditGet() 関数を使用する



フィールドの現在の値を取り出す場合は、以下の式を使用します。

```
EditGet()
```

ed を入力すると、オートコンプリートのポップアップメニューが表示されるのでここから関数を選択できます。

EditGet() 関数は、イベントロジックユニット内で起動されると、入力されたデータを返します。

EditGet() 関数はどのような編集フィールドに対しても動作します。また、式エディタではどのようなデータ型が返るのかが判別できません。上図の例において、数値フィールドで押下された F7 に対してロジックユニットが実行された場合、結果は無効になります。

このような問題を回避するため、コントロール名や **HandledCtrl()** 関数と組み合わせて使用するようにしてください。

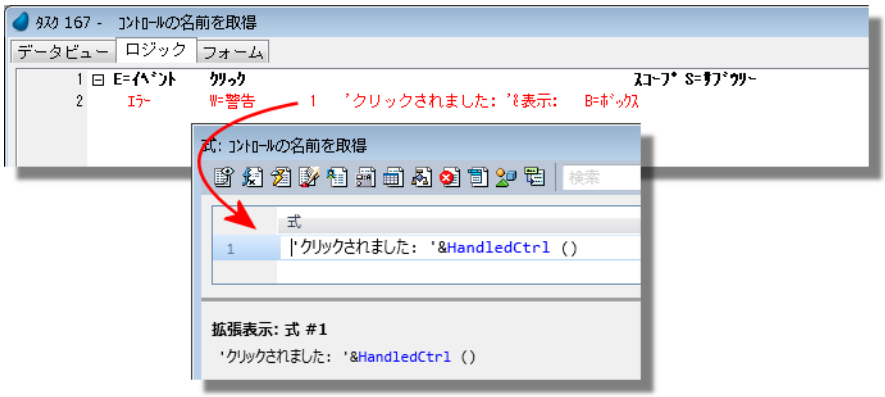
コントロール修正イベントをトリガとした場合、フィールドの修正を行った時点でロジックユニットが実行されるため、ここで **EditGet()** 関数を使用すると入力内容をチェックすることができます。

参照： 「イベントが発行されたコントロールを識別するには」（237 ページ）
「特定のコントロールにパークしている時にのみ実行されるイベントロジックユニットを定義するには」（238 ページ）

イベントが発行されたコントロールを識別するには

イベントロジックユニットは、それを処理するタスクより高いレベルで存在することができます。このよい例として、メインプログラム内でのイベントロジックユニットが挙げられます。これはプロジェクト内のどのタスクでも利用することができます。ユーザがどのコントロールでパークしているかを知るためには、**HandledCtrl()** 関数を使用する必要があります。

HandledCtrl() 関数を使用する



1. 式エディタを開きます。
2. 式を入力します。

```
HandledCtrl()
```

Ha を入力すると、オートコンプリートのポップアップメニューが表示されるのでここから関数を選択できます。

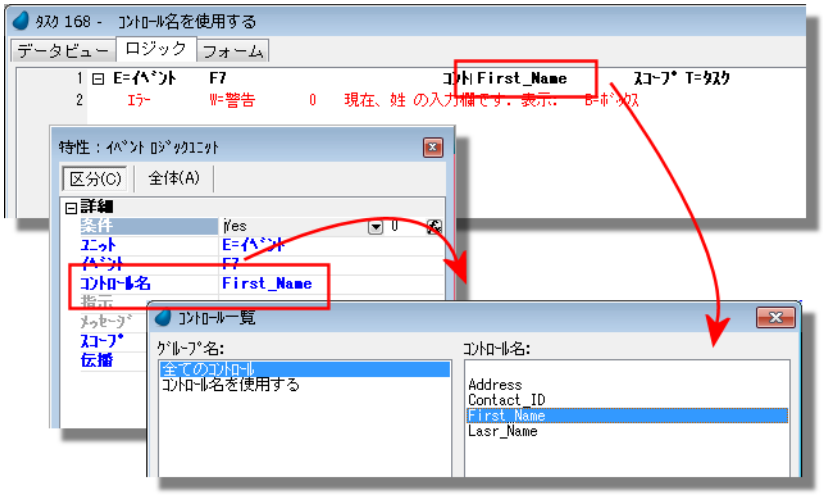
これで、イベントロジックユニット内で決定をするためにコントロール名を使用することができます。例えば、特定のヘルプファイルを呼び出すために、コントロール名の文字に対応した汎用的なヘルプシステムを持つことができます。

参照： 「特定のコントロールにパークしている時にのみ実行されるイベントロジックユニットを定義するには」（238 ページ）

特定のコントロールにパークしている時にのみ実行されるイベントロジックユニットを定義するには

イベントロジックユニットを使用することで、イベントを簡単に受け取ることができます。特定のコントロールにパークされている場合のみ、イベントが実行される必要があるのであれば、イベントの**コントロール名**特性を使用することができます。

イベントのコントロール名特性を使用する

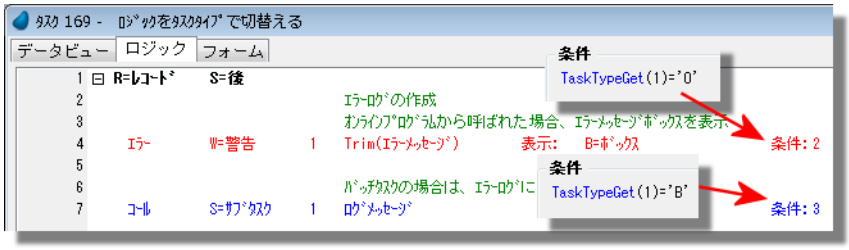


イベントロジックユニットの**コントロール名**特性から、ズーム（*F5* またはダブルクリック）してイベントが実行されるコントロール名を選択します。

この例では、システムイベントの *F7* に対応して First_Name という名前のコントロールを選択しています。従って、ユーザが *F7* を押下すると、ユーザが First_Name という名前のコントロールにパークしている場合のみロジックユニットが実行されます。

ヒント : *複数のコントロールに対して 1 つのイベントハンドラハンドルを定義する必要がある場合は、イベントハンドラで* **HandledCtrl()** *関数を使用します。詳細は、「イベントが発行されたコントロールを識別するには」 (237 ページ) を参照してください。*

同じロジックユニット内で実行されるロジックをタスクタイプによって切り替えるには



プログラミングにおいて工数を削減する方法の 1 つは、異なる様々なプログラムから呼び出されるイベントハンドラを使用することです。しかし、ハンドラは呼び出されたタスクタイプに基づいて異なる動作をするように定義する必要があるかもしれません。たとえば、この例では、呼び出しているプログラムがオンラインプログラムの場合、ユーザに対してメッセージが表示されるようになっています。これ以外の場合は、ログにエラーメッセージが書き込まれます。

この種のコンテクストに応じたプログラミングを作成するために、どのようなタスクがこのロジックユニットを呼び出したかを判断するために、**TaskTypeGet()** 関数を使用することができます。

TaskTypeGet() 関数の構文は、以下の通りです。

TaskTypeGet(*Generation***)**

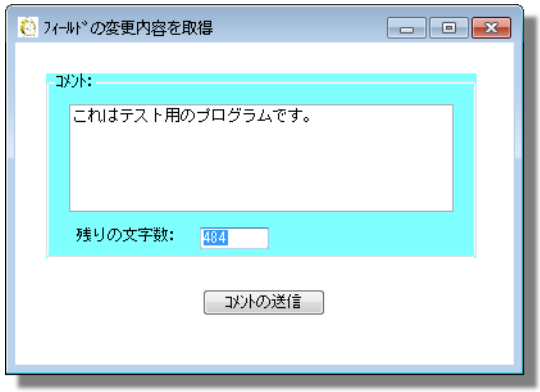
パラメータ：

- *Generation*：タスクツリー上のタスクの位置を表す数値。0 は現在のタスク、1 は親タスクを表します。

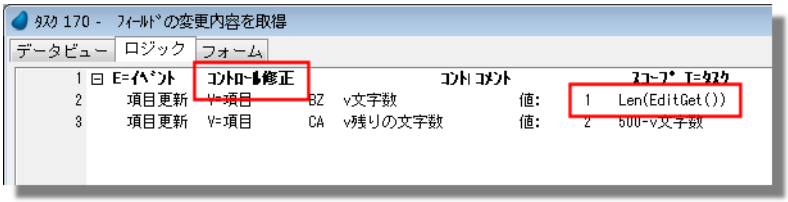
TaskTypeGet(*1***)** は、呼び出したタスクのタイプを返します。戻り値として、以下の値が返ります。§

- **O**= オンラインタスク
- **B**= バッチタスク
- **C**= リッチクライアントタスク
- **R**= ブラウザクライアントタスク
- **M**= メインプログラム
- **MC**= リッチクライアントプログラム内からのメインプログラム

エンドユーザがコントロールにデータを入力する時に、実行されるロジックを定義するには



通常、Magic xpa では、カーソルがフィールドを抜けた後、編集フィールドのデータはデータ項目に格納されます。しかし、ユーザがまだ入力している途中に、データを格納する必要があるかもしれません。一般的な例として、コメントボックスが挙げられます。コメントのスペースは制限されています、そして、データを入力すると、ユーザは「残りの文字」の数が少なくなることが確認できるはずです。



この種のインタラクティブな捕捉機能は簡単に実現することができます。

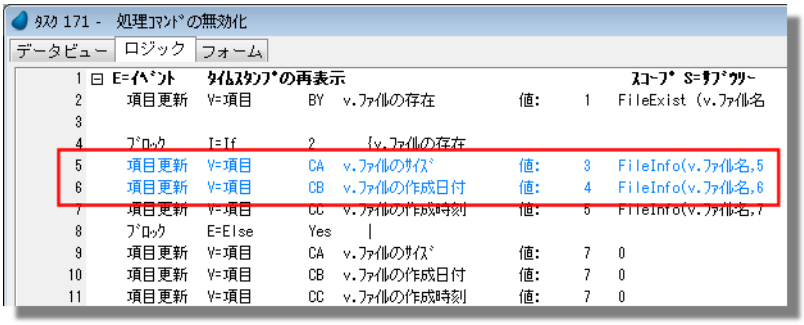
コントロール修正イベントを使用する

1. イベントロジックユニットを作成します。ここでは、**コントロール修正**の内部イベントによって実行されるように設定し、実行させたいコントロールを割り当てます。
2. ロジックユニット内で、**EditGet()** 関数を使用してユーザが入力したデータを受け取るようにします。この例では、**Len(EditGet())** を実行することで、入力データの長さ（バイト長）も取得しています。

様々な目的（インタラクティブな最適データリストや、エラーチェックなど）に応じて、ロジックを修正することができます。

注： 項目は、コントロールを抜ける場合のみ新しい値で更新されます。このため、項目自体を直接参照せず、**EditGet()** を使用する必要があります。

処理コマンドを無効にするには



プログラムのロジックをテストする間、ロジックの一部を働かないようにすることができれば便利な場合があります。このような場合、コマンド行を無効にすることで簡単に実現できます。

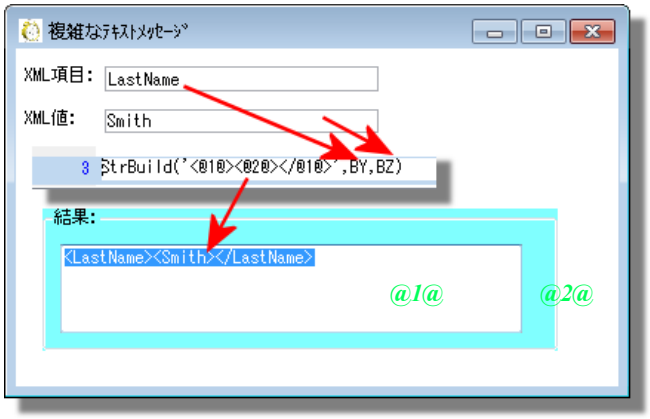
1. **ロジック**タブを開き、実行させたくない行に移動します。
2. **編集→行を無効化**（*Ctrl+Shift+D*) を選択します。これは、コンテキストメニューから選択することもできます。

無効にされた行は灰色になり、実行されないことを示します。

再度コマンド行を実行できるようにするには、**編集→行を有効化**（*Ctrl+Shift+D*) を選択します。

注： Windows の配色指定によっては、判りづらい場合があります。そのような場合は、開発用基本色の #77 の色を変更してください。

複雑なテキストメッセージを動的に作成するには



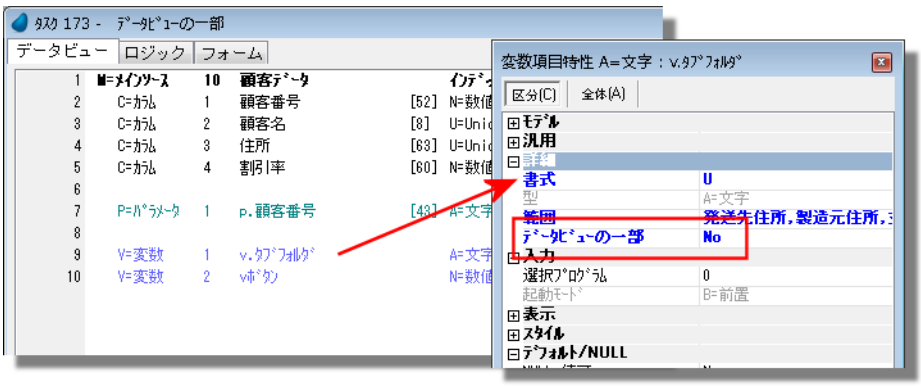
Magic xpa は、テキスト・メッセージを作成するために使用できる文字列関数が豊富にそろっています。しかし、StrBuild () 関数と呼ばれる、簡単な方法を使用することもできます。**StrBuild()** 関数は、様々な文字列を簡単にフォーマット化することができます。テンプレート部は、実行時に編集／保存することが可能なテキスト文字列であるため、ユーザによって変更される文字列のフォーマットを指定することができます。

StrBuild() は、2 つのセクションを持っています。最初の部分は、プレーンテキストを含み、@というプレイスホールダ（例えば @1@、@2@、@3@.）が先頭と最後に付加された番号タグを使用します。2 番目の部分は一連のパラメータです。これはプレイスホールダに割り当てられます。

この例では、**StrBuild()** は XML フォーマットで出力する簡単な方法として使用されています。余分な空白が自動的にトリミングされることに注意してください。しかし、数字または日付 / 時間データを文字列にフォーマット化することは任意です。

変数を更新した後にレコード後が実行されることを防止するには

通常、オンラインまたはリッチクライアントタスクにおいて、**レコード後**は、データビュー内のデータ項目が変更された場合に実行されます。しかし、変数項目の値はデータに関しては重要でないこともあります。これらの変更によってデータビューの更新処理を実行させたくない場合があります。たとえば、タブフォルダがダイアログ表示の変更でのみ使用されていたり、値がデータとして保存されない場合です。



フィールドをデータビューから分離するには：

1. **データビュー**タブを開き、項目定義の行に移動します。
2. **特性シート**の**詳細**セクションの**データビューの一部**特性を *No* に設定します。

これで、項目の値が変更されても、**レコード後**は実行されません。

［このページは意図的に空白にしています。］

第 11 章 : 日付と時刻

現在の日付を取得するには

Magic xpa で現在の日付を取得するには、**Date()** 関数を使用します。**Date()** 関数は、Magic xpa が現在実行中の PC のシステム日付を返します。

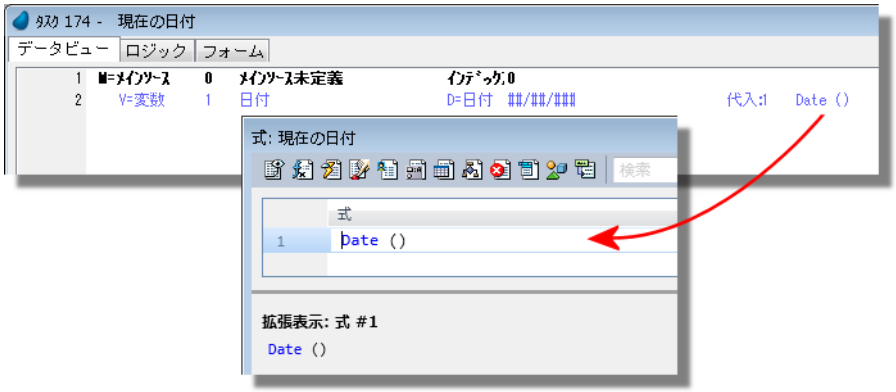
注 : この関数は、**MDate()** 関数とよく似ています。この関数は、システム日付ではなくユーザが設定可能なログイン日付を返します。この関数が使用される場合もあります。例えば、経理システムのようにユーザに対して登録日付の変更を可能にさせる必要がある場合に使用します。

日付の格納形式

Magic xpa での日付は、0001/01/01 以降からの日数を表す数値で格納されています。変換はすべて自動的に行われるため、開発者が意識する必要はありませんが、内部処理上は数値として扱われるということです。従って、5 日を追加する場合、もとの日付から 5 日を進めることになります。月末の日付やうるう年のような問題に関して考慮する必要はありません。

日付がデータソース内に格納される場合、記憶形式はデータソース定義で設定された特性によって決定されます。データがレポートライタなどのような他のツールと共有している場合、日付フィールドの定義が、他のツールで処理が可能であることが重要になります。

Date() 関数を使用する



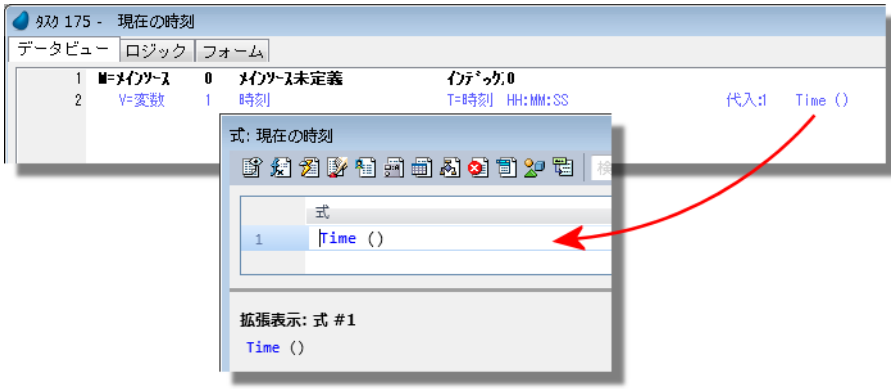
ここで示されるように、**Date()** 関数は**代入**カラムでよく使用されます。これは、タスクが起動された時に現在の日付を表示させたり、レコードの新規作成時にタイムスタンプを設定する場合に便利です。

この場合、式には、**Date()** と入力するだけです。実行時、関数は現在の日付を返します。

現在の時刻を取得するには

Magic xpa で現在の時間を取得するには、**Time()** 関数を使用します。**Time()** 関数は、Magic xpa が実行している PC のシステム時間を返します。

Time() 関数を使用する



ここで示されるように、**Time()** 関数は**代入**カラムでよく使用されます。これは、タスクが起動された時に現在の時刻を表示させたり、レコードの新規作成時にタイムスタンプを設定する場合に便利です。

この場合、式には、**Time()** と入力するだけです。実行時、関数は現在の時刻を返します。

現在の時間をミリ秒単位で取得するには

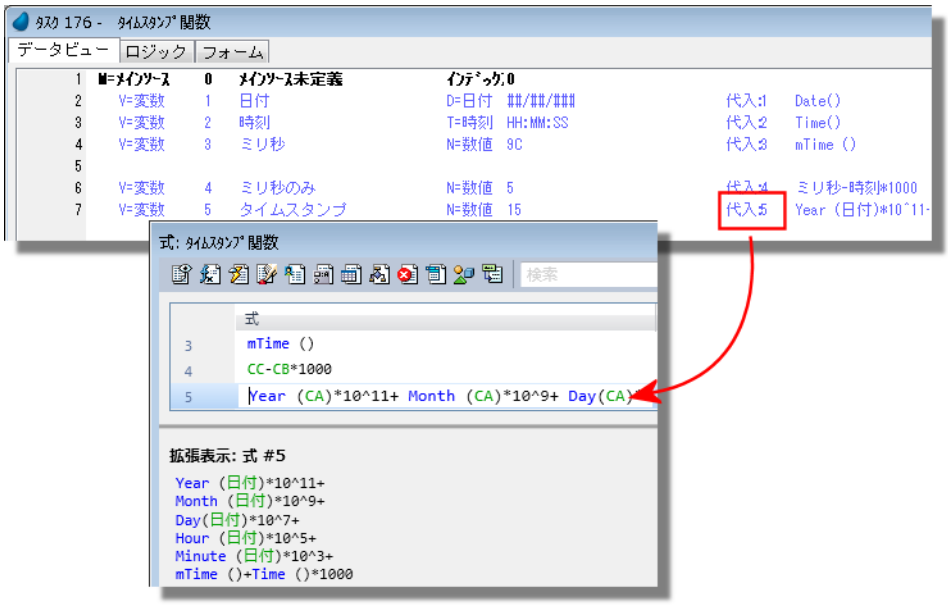
Time() 関数は、0 時以降の秒数を返します。秒数で格納されてはいますが、通常は **HH : MM : SS** の書式で表示されます。

しかし、タイムスタンプを作成する上で細かい単位が必要な場合があります。**mTime()** 関数は、このような場合に使用します。この関数は、0 時以降のミリ秒数を返します。

ミリ秒数を表示させるための自動化された機能はありません。ミリ秒を取得するには以下の式を使用します。

```
mTime() - Time() * 1000
```

mTime() 関数を使用してタイムスタンプを作成する



ここに示された式を使用して数値データとしての 日付 / 時刻のタイムスタンプを作成することができます。この場合、18 桁の数値コードを作成します。

```
YYYYMMDDHHMMSSmmm
```

文字列でタイムスタンプを作成することもできますが、数値で作成した方がバイト数が少なくすみます。

日付データの加算処理を行うには

前述のように、Magic xpa における日付データは 0 年以降の日付を表す数値が格納されています。従って、日付を加算するには、日数を追加するだけで可能になります。例えば：

```
Date() + 5
```

この式では、現在から 5 日後の日付が返ります。

しかし、年数や月数および日数をもとに加算処理を行う場合は、**AddDate()** 関数を使用します。この関数は、以下の４つのパラメータを使用します。

AddDate(Date, Years, Months, Days)	
Date	加算対象の日付データ
Years	加算する年数を表す数値
Months	加算する月数を表す数値
Days	加算する日数を表す数値

例えば、以下のような式を作成した場合：

```
AddDate(B,0,-1,0)
```

指定された日付から 1ヶ月が減算されます。従って、例えば 2008 年 7 月 6 日をもとに計算を行った場合、**AddDate()** 関数は 2008 年 6 月 6 日を返します。

時刻データの加算処理を行うには

Magic xpa における時刻データは 0 時以降の日付を表す数値が格納されています。従って、時刻を加算するには、秒数を追加するだけで可能になります。例えば：

```
Time() + 5
```

この式では、現在から 5 秒後の時刻が返ります。

しかし、時数や分数および秒数をもとに加算処理を行う場合は、**AddTime()** 関数を使用します。この関数は、以下の４つのパラメータを使用します。

AddTime(Time, Hours, Minutes, Seconds)	
Time	加算対象の時刻データ
Hours	加算する時数を表す数値
Minutes	加算する分数を表す数値
Seconds	加算する秒数を表す数値

例えば、以下のような式を作成した場合：

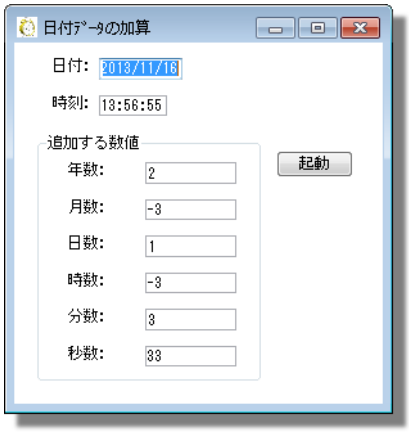
```
AddTime(B,0,-1,0)
```

指定された日付から 1 時間が減算されます。従って、例えば 14:08:03 をもとに計算を行った場合、**AddTime()** 関数は 13:08:03 を返します。

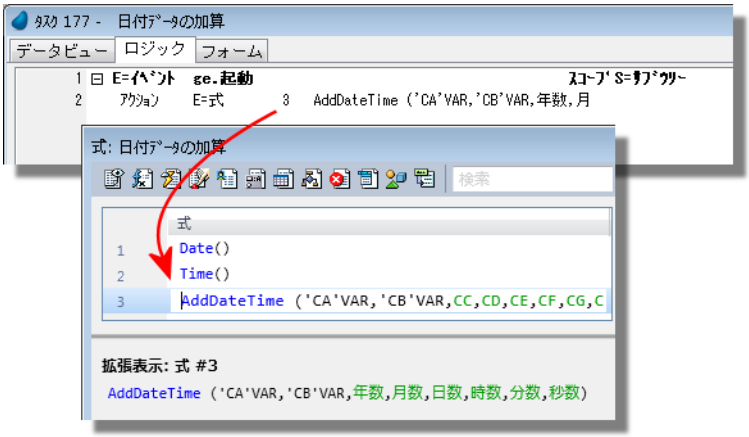
日付／時刻データを加算するには

日付と時刻の項目の値に対する加算処理は、さまざまな処理で使用されます。例えば、22:00:00 に業務を開始した作業者に対する作業時間を計算する場合を考えてみます。開始時間から 8 時間を追加した場合、それに応じて日付も加算する必要があります。

このような処理に対して、Magic xpa には 1 回で日付と時間を扱うことができる関数があります。**AddDateTime()** 関数は、日付と時間（年、月、日、時、分、秒）を指定することで、まとめて加算 / 減算の処理を行うことができます。



AddDateTime() 関数を使用する



AddDateTime() 関数は以下の構文で指定します。

AddDateTime(*DateVariable*, *TimeVariable*, *Years, Months*, *Days, Hours*, *Minutes*, *Seconds***)**

パラメータ：

- *DateVariable* ：更新対象の日付データ
- *TimeVariable*：更新対象の時刻データ
- その他のパラメータは、日付や時刻の各部分を表す数値です。
  正の数値を指定すると加算、負の数値を指定すると減算になります。

項目の指定方法について

上記の例では、最初の 2 つのパラメータの項目で **VAR** リテラルを使用しています。つまり、クォーテーションで囲まれた項目番号の後に VAR という文字が続いています（'CA'VAR と 'CB'VAR）。これは、Magic xpa がアドレス参照で項目を指定するための書式です。この指定は、式の中で項目を更新する場合に指定する必要があります。通常、Magic xpa は **項目更新**処理コマンドで項目を更新しますが、この場合、同時に 2 つの項目を更新しているため、このような参照指定が必要になります。

２つの日付／時刻データの差分を計算するには

２つの日付 / 時刻データの差分を計算する処理は、さまざまな処理で使用されます。

このような処理に対して、Magic xpa には 1 回で日付 / 時間の差分を計算することができる関数があります。**DifDateTime()** 関数は、以下の例のように２つの日付 / 時刻の組み合わせデータの差分を計算することができます。

DifDateTime() 関数は、差分の日数と秒数を返します。この秒数を時間データとして扱い、**Hour()**、**Minute()** そして **Second()** の各関数を使用して、時数 / 分数 / 秒数に変換することができます（「指定された時刻の時 / 分 / 秒の各部分を取得するには」（259 ページ）を参照してください）。

DifDateTime() 関数を使用する

DifDateTime() 関数は以下の構文で指定します。

DifDateTime(*Date1*, *Time1*, *Date2*, *Time2*, *DaysVariable*, *SecondsVariable***)**

パラメータ：

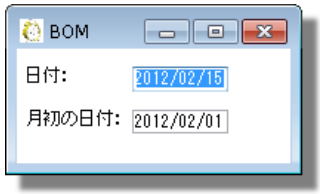
- *Date1* と *Time1*：最初の日付 / 時刻の組み合わせデータ
- *Date2* と *Time2*：２番目の日付 / 時刻の組み合わせデータ
- *DaysVariable*：差分の日数データを表す数値
- *SecondsVariable*：差分の秒数データを表す数値

項目の指定方法について

上記の例では、最初の２つのパラメータの項目で **VAR** リテラルを使用しています。つまり、クォーテーションで囲まれた項目番号の後に VAR という文字が続いています（'CC'VAR と 'CD'VAR）。これは、Magic xpa がアドレス参照で項目を指定するための書式です。この指定は、式の中で項目を更新する場合に指定する必要があります。通常、Magic xpa は **項目更新**処理コマンドで項目を更新しますが、この場合、同時に２つの項目を更新しているため、このような参照指定が必要になります。

指定された日付に対応する月の最初の日付を取得するには

帳票処理のための日付範囲を求めるために、月の最初の日付を取得する必要がある場合があります。**BOM()** 関数を使用することで、このようなことが実現できます。



BOM() 関数を使用する

BOM() 関数の構文は以下の通りです。

BOM(*date***)**

パラメータ：

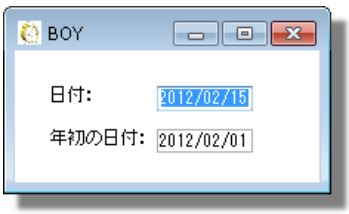
- *date* ：日付項目（直接日付を指定するか日付データを返す式を指定）

この関数は、月の最初の日付を返します。

参照： 「指定された日付に対応する月の最後の日付を取得するには」（254 ページ）

指定された日付に対応する年の最初の日付を取得するには

帳票処理のための日付範囲を求めるために、年の最初の日付を取得する必要がある場合があります。**BOY( )** 関数を使用することで、このようなことが実現できます。



BOY() 関数を使用する

BOY( ) 関数の構文は以下の通りです。

BOY(*date***)**

パラメータ：

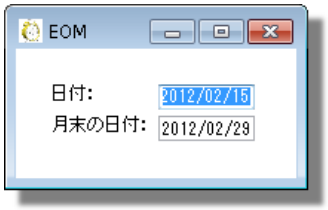
- *date* ：日付項目（直接日付を指定するか日付データを返す式を指定）

年の最初の日付を返します。

参照： 「指定された日付に対応する年の最後の日付を取得するには」（255 ページ）

指定された日付に対応する月の最後の日付を取得するには

帳票処理のための日付範囲を求めるために、月の最後の日付を取得する必要がある場合があります。**EOM()** 関数を使用することで、このようなことが実現できます。



EOM() 関数を使用する利点の一つは、うるう年を正しく処理することです。

EOM() 関数を使用する

EOM() 関数の構文は以下の通りです。

EOM(*date***)**

パラメータ：

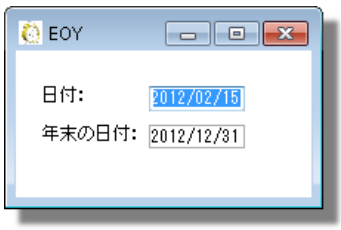
- *date* ：日付項目（直接日付を指定するか日付データを返す式を指定）

月の最初の日付を返します。

参照： 「指定された日付に対応する年の最初の日付を取得するには」（253 ページ）

指定された日付に対応する年の最後の日付を取得するには

帳票処理のための日付範囲を求めるために、年の最後の日付を取得する必要がある場合があります。**EOY()** 関数を使用することで、このようなことが実現できます。



EOY() 関数を使用する

EOY() 関数の構文は以下の通りです。

EOY(*date***)**

パラメータ：

- *date* ：日付項目（直接日付を指定するか日付データを返す式を指定）

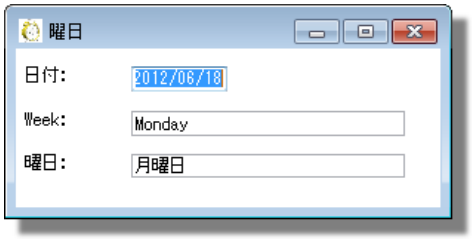
年の最後の日付を返します。

参照： 「指定された日付に対応する年の最初の日付を取得するには」（253 ページ）

指定された日付に対応する曜日名を取得するには

指定された日付が何曜日になるかを取得する場合は、**CDOW()** 関数を使用します。

日本語で曜日を表示させる場合は、**JCDOW()** 関数を使用します。



CDOW() 関数を使用する

CDOW() 関数の構文は以下の通りです。

CDOW(*date***)**

パラメータ：

- *date* ：日付項目（直接日付を指定するか日付データを返す式を指定）

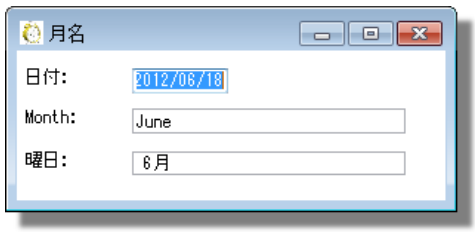
曜日を表す文字列を返します。

ヒント：*週単位の日付計算を行う必要がある場合は、***DOW()** *関数を使用することができます。この関数は週単位で日数（日曜日を 1 とした、日数）が返ります。プログラム内での演算用に使用するには、こちらの方が便利です。*

参照： 「日付書式」（261 ページ）

指定された日付に対応する月名を取得するには

指定された日付に対応する月名を取得する場合は、**CMonth()** 関数を使用します。



日本語で月名を表示させる場合は、**JMonth()** 関数と **Month()** 関数を組み合わせたり、**Month()** 関数で月数を求めて表示させるようにします。

CMonth() 関数を使用する

CMonth() 関数の構文は以下の通りです。

CMonth(*date***)**

パラメータ：

- *date* ：日付項目（直接日付を指定するか日付データを返す式を指定）

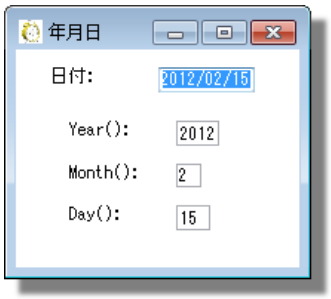
月名を表す文字列を返します。

ヒント : *月数の計算を行う必要がある場合は、**Month()** 関数を使用することができます。この関数は月数を数値で返します。プログラム内での演算用に使用するには、こちらの方が便利です。*

参照 : 「指定された日付の年 / 月 / 日の各部分を取得するには」（258 ページ）

指定された日付の年 / 月 / 日の各部分を取得するには

プログラム内で日付データを使用する場合、日付データを分離する必要があるかもしれません。例えば、その年の 1ヶ月間の全レコードを要約するような場合が考えれます。



日付データを分離するには、以下の３つの関数があります。

- **Day(*date*)**：その月の日数を表す数値を返します。
- **Month(*date*)**：月数を表す数値を返します。
- **Year(*date*)**：年数を表す数値を返します。

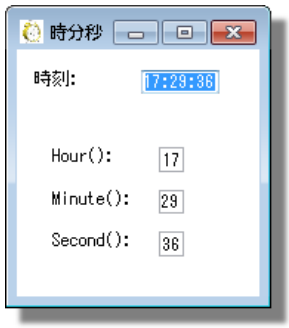
パラメータ：

- *date* ：日付項目（直接日付を指定するか日付データを返す式を指定）

ヒント：*月の名前や曜日の名前を表示させたい場合は、***CMonth()** *関数や***CDOW()** *関数を使用します。これらは、文字列で名前を返します。*

参照： 「指定された日付に対応する曜日名を取得するには」（256 ページ）

指定された時刻の時 / 分 / 秒の各部分を取得するには



時刻データを時数、分数、または秒数に分ける場合、計算処理を実行して値を取得することで実現できます。しかし、このような処理を簡単に行うことのできる関数があります。**Hour()**、**Minute()**、および **Second()** 関数を使用することで、分離された時刻データが返ります。

時刻データを分離するには、以下の３つの関数があります。

- **Hour(*time*)**：時数を表す数値を返します。
- **Minute(*time*)**：分数を表す数値を返します。
- **Second(*time*)**：秒数を表す数値を返します。

パラメータ：

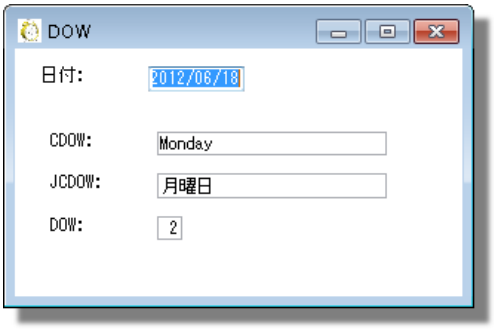
- *time* ：時刻項目（直接時刻を指定するか時刻データを返す式を指定）

ヒント：ミリ秒で取得したい場合は、**MTime()** を関数を使用します。

参照： 「現在の時間をミリ秒単位で取得するには」（247 ページ）

指定された日付の週単位の日数を取得するには

プログラムの内容によっては、週単位で処理を行う必要があるかもしれません。例えば、金曜日に給料が配達される場合、指定された日付が金曜日か否かを確認する必要があるはずです。



値が数値で返るようであれば、曜日を確実にチェックできます。特に、アプリケーションが他言語で実行されるような場合、このオプションはスペルや大文字／小文字の問題を低減させることになります。

指定された日付から週番号を取得するには、**DOW()** 関数を使用します。

DOW() 関数を使用する

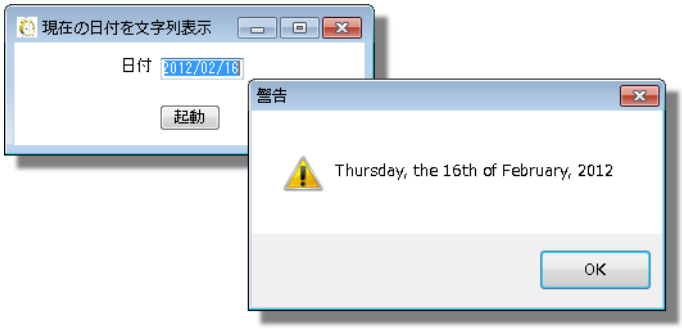
DOW(*date***)** は、週単位での日数を返します。返される値は、1（日曜日）から 7（土曜日）までです。

例えば、2007/08/19 は日曜日なので **DOW(*'2007/08/19'DATE*)** は、1 を返します。

ヒント : *指定された日付けの曜日名（英語）を取得する場合は、**CDOW()** 関数を使用します。日本語の曜日名を取得する場合は、**JCDOW()** 関数を使用します。*

参照 : 「指定された日付に対応する曜日名を取得するには」（256 ページ）

日付データを文字列に変換するには



Magic xpa における日付値は基本的には数値で、0001/01/01 以降の日数として扱われます。フォーム上の日付を表示する場合、コントロールの**書式**特性にもとづいて Magic xpa が自動的に処理するため、変換処理は必要ありません。

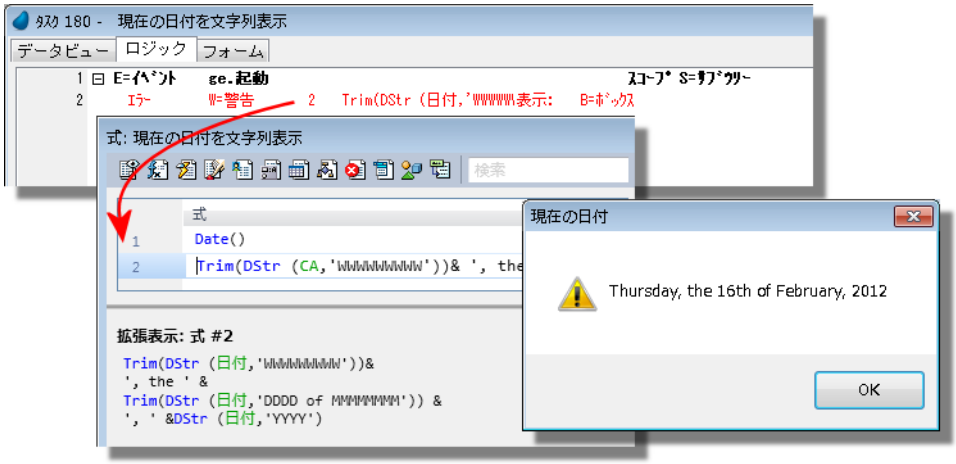
しかし、日付を文字列に含めたい場合（例えばエラー処理コマンドでのメッセージ）、日付を文字列に変換する必要があります。このような場合、**DStr()** 関数を使用して実現します。

日付書式

日付を表示する場合、Magic xpa には複数の書式を指定することができます。日付の書式は、どのように表示されるかを指定します。書式は、日付を解釈する一種のテンプレートです。文字で示されたプレイスホールダは、どのように日付を解釈するかを示すために使用されます。例えば、2007 年 6 月 30 日は以下のように扱われます。

書式	どのように表示されるか	コメント
YY/MM/DD	07/06/30	
YYYY/MM/DD	2007/06/30	
YYYYMMDD	20070630	
YYYY-MM-DD	2007-06-30	
YYYY	2007	
YY	07	
MMMM	June	
DDD	181	ユリウス歴での年内の日数
DDDD	30th	
WWW	Sat	
WWWWWWWWW	Saturday	
##/##/####	Depends	設定→動作環境→国別指定→日付モードの指定に依存します。

DStr() 関数を使用する



DStr() 関数の構文は以下の通りです。

DStr(*date, picture***)**

パラメータ：

date ：日付項目（直接日付を指定するか日付データを返す式を指定）

picure：文字列に変換する場合の書式

この関数は、***picure*** で指定された書式の文字列を返します。

参照： 「文字列で格納された日付を日付データに変換するには」（263 ページ）

文字列で格納された日付を日付データに変換するには

通常 Magic xpa は、日付変換処理を行います。すなわち、フォーム上に日付を表示したり、フォームの日付を読み込んだり、データソースから読み込む場合、変換処理を実行するプログラムを作成する必要がありません。しかし、文字列を日付として取得するためにその文字列を解析する必要がある場合、若干のプログラムを組む必要があります。このような場合、**DVal()** 関数を使用する必要があります。

DVal() 関数は文字列を解析して日付データに変換するために指定呈された書式を使用します。書式と文字列の整合性は、開発者に依存しています。例えば以下のようになります。

パラメータの例	結果
DVal('07/06/30'DATE, 'YY/MM/DD')	2007,June 30
DVal('06/30/07'DATE, 'DD/MM/YY')	不正な結果

参照 : 「日付データを文字列に変換するには」（261 ページ）

［このページは意図的に空白にしています。］

第 12 章 : GUI の処理

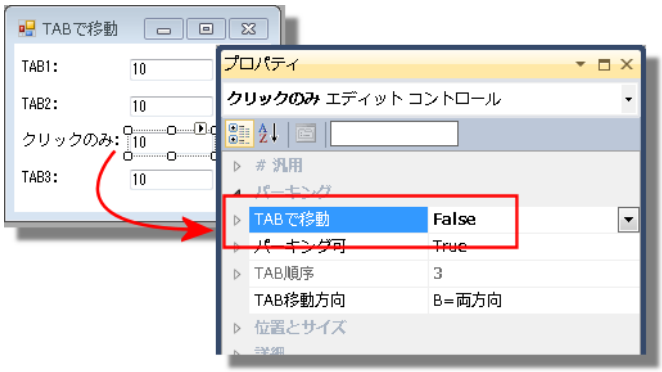
注 : Magic xpa Ver3 では、表示フォームはフォームデザイナで編集するようになりました。ここでは、特性値などの表示が旧バージョンと異なる場合があります。

マウスを使用した場合のみコントロールへのパークを可能にするには

デフォルトでは、オンラインフォーム上のフィールドは、*Tab* によってカーソルをパークさせることができます。しかし、マウスを使用した場合のみアクセス可能にしたい場合は、コントロールの **TAB で移動**特性を *False* に設定します。

式で指定することで、必要に応じて *Tab* 移動を可能にすることもできます。例えば、表示されている文字列をコピーする場合のみパークさせるというような制御が可能です。

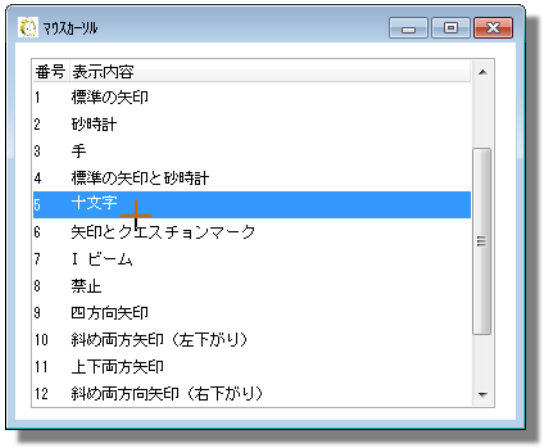
TAB で移動特性を設定する



1. 変更したいコントロールを選択します。必要であれば、*Ctrl+* クリックまたはラバーバンドを使用して、複数のコントロールを選択することで同時に変更することもできます。
2. コントロール特性（*Alt+Enter*、すでに開いている場合は特性シートをクリック）に移動します。
1. **TAB で移動**特性の値を *False* に設定します。式で指定する場合は、右側に移動し、ズーム（*F5*）して式エディタを開きます。パークさせたい場合に True が返る式を入力します。

   これで、ユーザは、フィールドに *Tab* 移動することはできなくなりますが、マウスでクリックすることは可能です。

マウスカーソルのアイコンを変更するには



通常の Windows アプリケーションと同じように Magic xpa もマウスカーソルを制御しています。しかし、必要に応じてデフォルトの表示内容を変更することができます。

Magic xpa では、14 個の異なるアイコンのセットから選択できます。これらのアイコンは、Windows 環境で標準的に持っているものです。例えば、標準の矢を大きくしたり、異なるシンボルを使用したり、ユーザ毎に独自の Windows 設定を変更することができます。

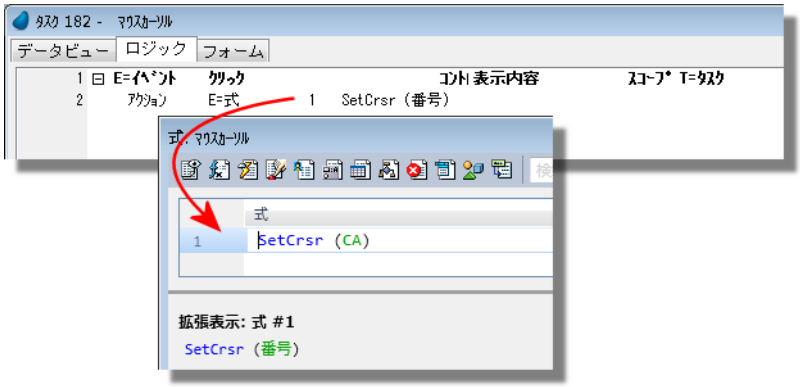
使用するアイコンを変更するには、**SetCrsr()** 関数を使用します。構文は以下のとおりです。

SetCrsr(*number*)

パラメータ：

- ***number***：1 ～ 14 の数値です。

SetCrsr() 関数を使用する



1. カーソル形状の変更処理を実行させたいロジックユニットに移動します。上記の例では、ユーザがカーソルタイプのリストをクリックすることで、カーソルの形状が変化するようになります。
2. アクション処理コマンドを作成します。
3. 以下の式を設定します。

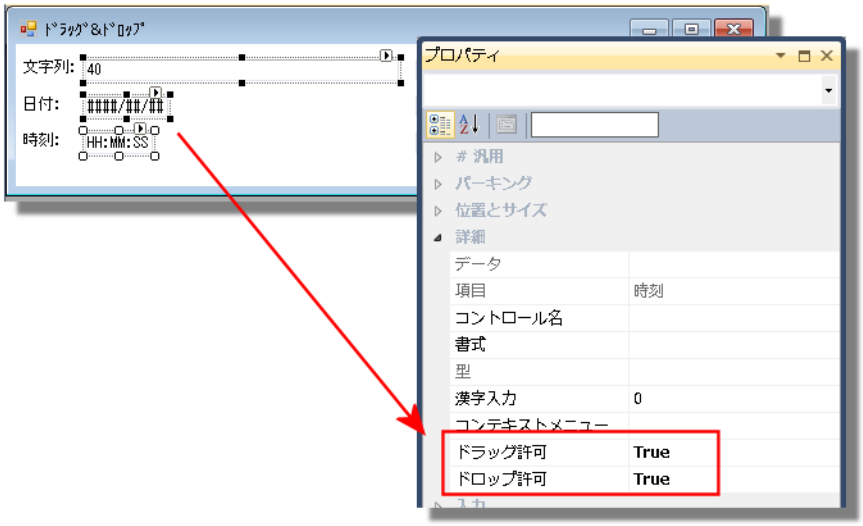
   SetCrsr (number)

   パラメータの ***number*** は変更したいカーソルの形状を現す数値です。1 ～ 14 の数値を直接指定すること可能ですが、この例では、数値型項目を使用しています。

   これで、ロジックユニットが実行されると、カーソルの形状が変化します。

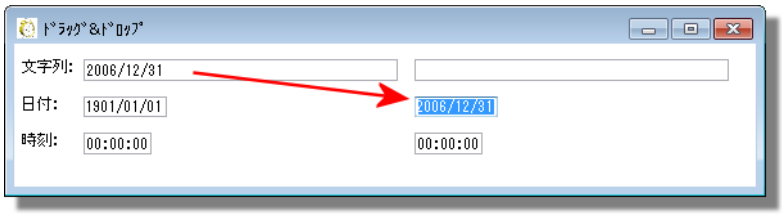
注： **アイコンを変更させた後は、デフォルトの形状に戻すようにリセット処理を組み込むようにしてください。**

ドラッグ & ドロップでデータをコピーするには



アプリケーションでドラッグ＆ドロップ処理を行うには、**ドラッグ許可**と**ドロップ許可**の各特性を ***True*** に設定する必要があります。

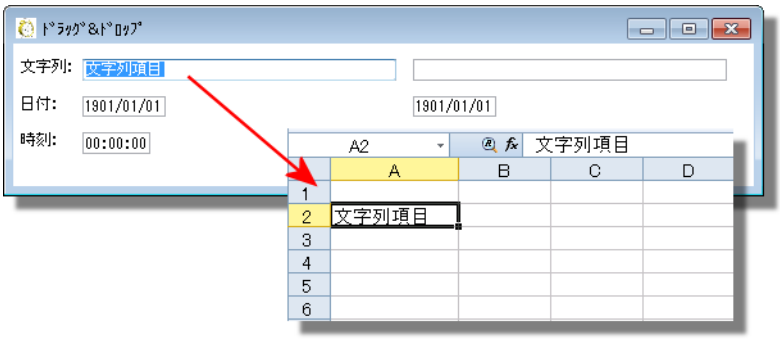
すべて設定されていれば、データをフィールドでドラッグすることができます。



例えば、この例では、ドラッグ＆ドロップ処理を使用して日付フィールドに文字列のデータをコピーしています。

Magic xpa でより高度なドラッグ＆ドロップの編集処理を行うには、ドラッグ＆ドロップ関数やハンドラを使用する必要があります。ドラッグまたはドロップ処理が発生した場合、**ドラッグ開始**と**ドロップ**に対する**イベント**ロジックユニットによってこれらを処理することができます。**DragSetData()** と **DropFormat()** 関数を使用することで、ドラッグデータの書式に関するより細かい制御が可能になります。

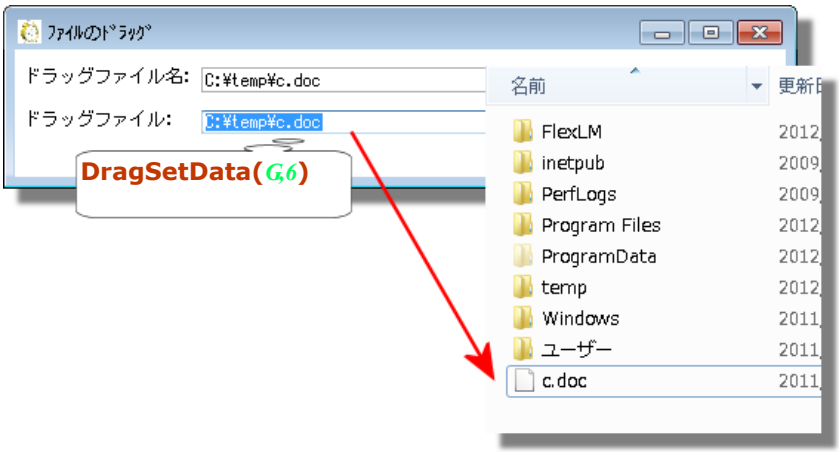
外部アプリケーションにデータをドラッグ & ドロップするには



外部アプリケーションへのドラッグ & ドロップを有効にするには、ドラッグしたいコントロールの**ドラッグ許可**特性を *True* に設定する必要があります。

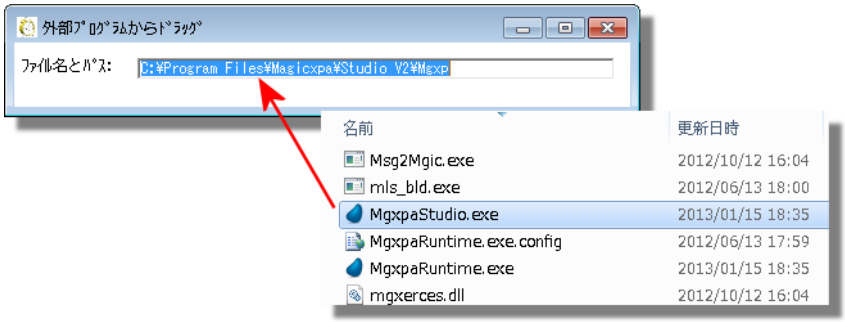
Magic xpa でより高度なドラッグ&ドロップの編集処理を行うには、ドラッグ&ドロップ関数やハンドラを使用する必要があります。ドラッグまたはドロップ処理が発生した場合、ドラッグ開始とドロップの各イベントロジックユニットがこれらを取得することができます。**DragSetData()** と **DropFormat()** 関数を使用することで、ドラッグデータの書式に関するより細かい制御が可能になります。

例えば、ファイルを参照ウィンドウ内にドラッグしたい場合、ファイルのテキスト名をドラッグすることはできません。参照プログラムに対して、ドラッグするものがファイル全体であることを指定する必要があります。このような処理を実行するには、ドラッグ開始のイベントロジックユニット内で **DropSetData()** 関数を使用します。

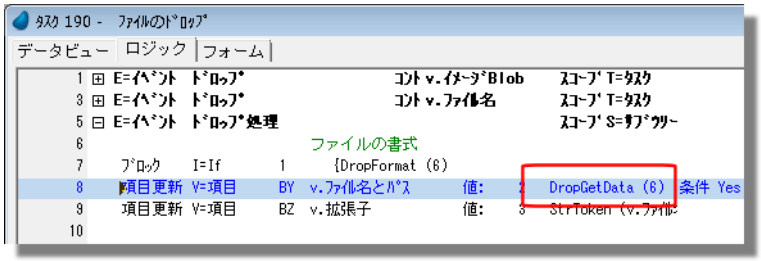


様々な状況において、外部データの書式を処理するために **DropGetData()** 関数を使用する必要があります。例えば、Magic xpa のフィールド内に JPG ファイルを Windows エクスプローラからドラッグすることを想定してください。**DropGetData(*6*)** を使用することでファイル名を取り出すことができます。ファイルの拡張子をチェックし、それが JPG ファイルであると識別された場合、画像を表示させるにはファイルを BLOB に変換する必要があるため、**File2Blb()** 関数を使用します。

ドラッグされたファイルからファイル名を取得するには



Windows からファイルを文字型フィールドにドラッグすると、フルパスのファイル名が自動的に取得されます。ファイル参照ウィンドウ内で２つのファイルを選択した場合、両方のファイル名をパイプ（|）区切りで取得することができます。

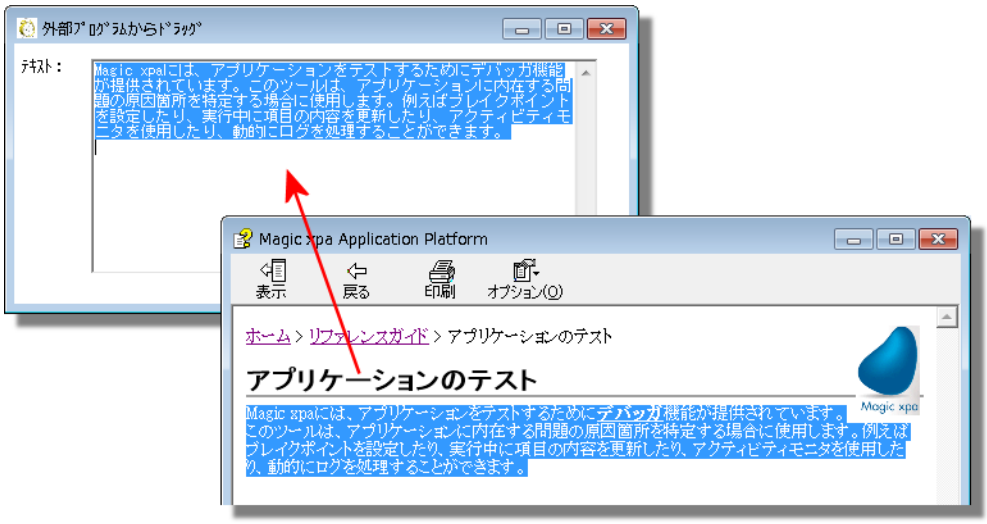


手動でこのファイル名を取り出すには、**DropGetData()** 関数を使用します。

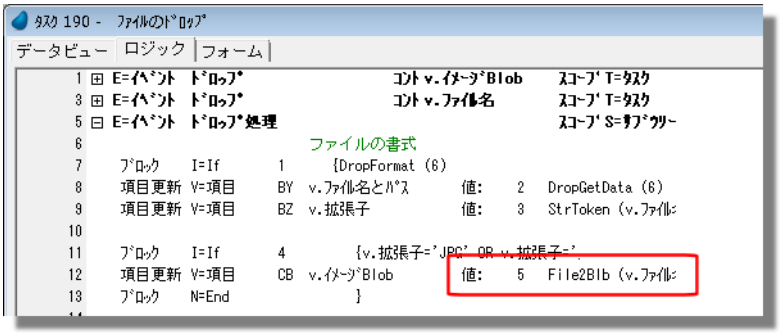
上記のプログラム例では、以下のように関数を使用しています。

- **DropFormat(*6*)**：ファイルがドラッグされていることをチェックしています。
- **DropGetData(*6*)**：項目を更新することでファイルの名前を取得しています。

外部アプリケーションから Magic xpa にデータをドラッグするには

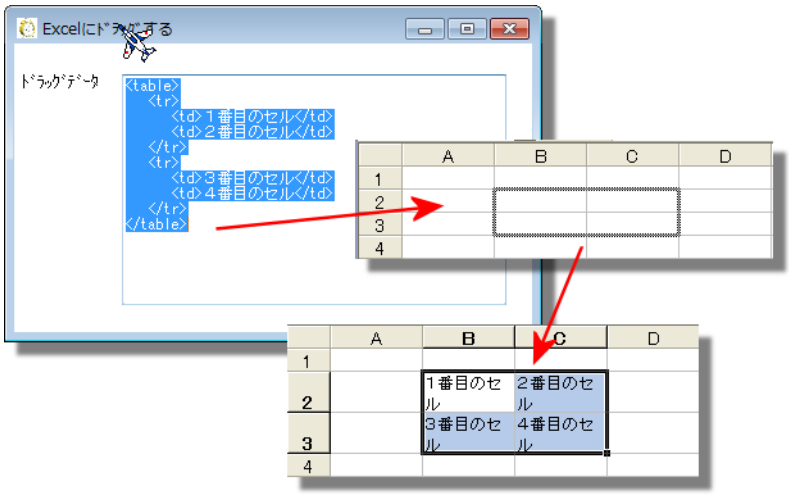


テキストコントロールの**ドロップ許可**特性が *Yes* に設定されている場合、他のアプリケーションから Magic xpa にテキストを**ドラッグ**することができます。この動作の内容の善し悪しは、データを**ドラッグ**するアプリケーションの内部処理に依存します。これは、Windows 内のドラッグ & ドロップ処理と同じように、２つのフィールド間の書式を合わせる必要があるからです。

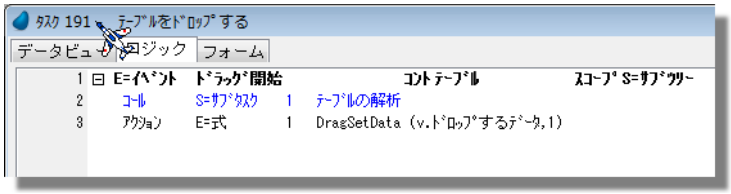


様々な状況において、外部データの書式を処理するために **DropGetData()** 関数を使用する必要があります。例えば、Magic xpa のフィールド内に JPG ファイルを Windows エクスプローラからドラッグすることを想定してください。**DropGetData(*6*)** を使用することでファイル名を取り出すことができます。ファイルの拡張子をチェックし、それが JPG ファイルであると識別された場合、画像を表示させるにはファイルを BLOB に変換する必要があるため、**File2Blb()** 関数を使用します。

テーブルのデータを Excel にドラッグするには



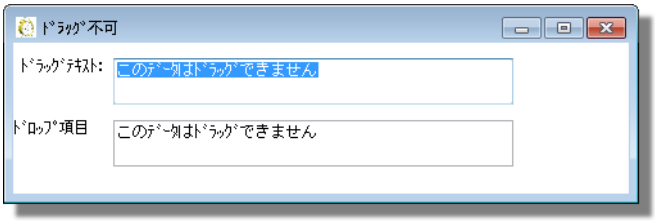
Magic xpa から Excel にデータを**ドラッグ**した場合、Excel はその書式を認識し、それに対応したデータとして処理します。この例では、複数のテーブルデータを HTML 形式に変更することで、テキストを Excel に**ドラッグ**しています。Excel はこのテキストを 4 つのセルを持つテーブルとして認識し、それに応じてデータが**ドロップ**されています。



ユーザに対し、Magic xpa のデータソースの内容を ドラッグ することを許可する場合、Magic xpa のデータを文字列フィールド内に HTML または XML 形式で格納する必要があります。そして、ドラッグするテキストデータを送るために **DragSetData()** 関数を使用します。

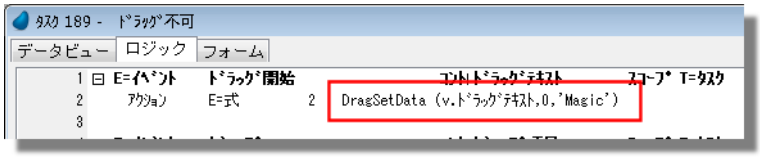
ドラッグ可特性を ***True*** に設定した**グループ**コントロール内にテーブルを配置して、この**グループ**コントロールに対する**ドラッグ開始**イベントでロジックが実行するようにします。

外部アプリケーションがデータを取得できないようにするには



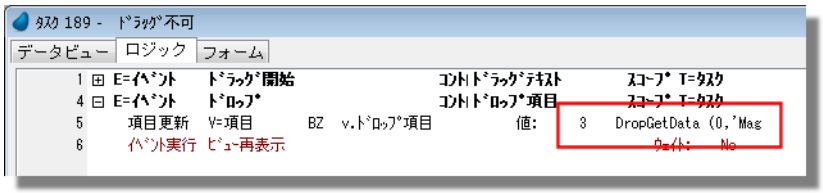
外部アプリケーションは、処理に必要なデータフォーマットと整合性がとれない限り**ドロップ**されたデータを読むことはできません。アプリケーションが指定されたデータフォーマットを認識していない場合、**ドロップ**は許可されません。

従って、例えば、**フォーマット 0** の「**ユーザ定義フォーマット**」を使用した場合、どの外部アプリケーションもデータフォーマットを認識することができず、**ドロップ**も許可されません。

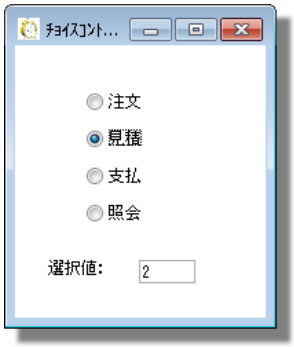


この例において、フォーマットを 0 に設定するために、**DragSetData()** 関数を使用しています。また**伝播**特性を *No* に設定しているため、組み込まれているドラッグ機能も働きません。

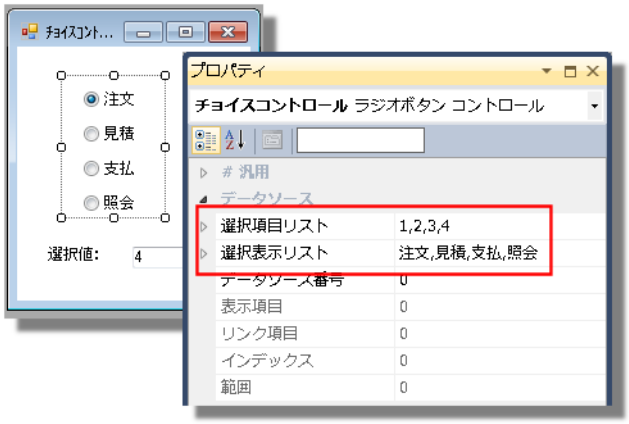
この設定を行った場合、**ドロップ**ハンドラ内で同じフォーマット指定が使用されるまで、データは Magic xpa の他のフィールドに**ドラッグ**することはできません。



チョイスコントロールで定義されている値と異なるテキストを表示するには



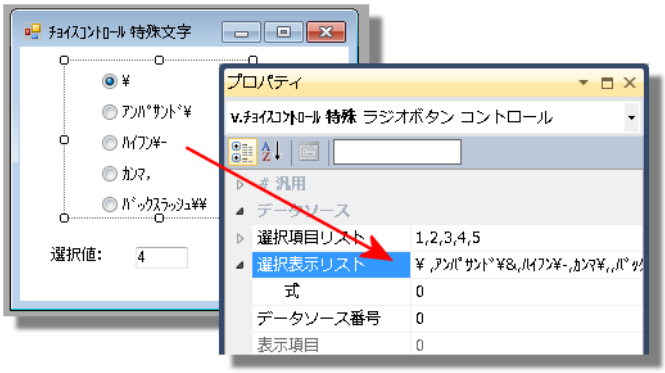
チョイスコントロールを使用する際、定義されているデータと異なる内容を表示させたい場合があります。異なる言語環境で実行するアプリケーションを開発する場合、特に重要な問題となります。この例では、4 つの指示が表示されています。注文、見積、返済、問い合わせです。しかし、プログラム内部では 1、2、3、4 のコードとして格納されています。



ここで示されているように、この設定内容はコントロールにてすべて処理されます。**選択項目リスト**特性にチョイスコントロールの値を順番に設定します。**選択表示リスト**特性に、同じ順番で表示項目を設定します。

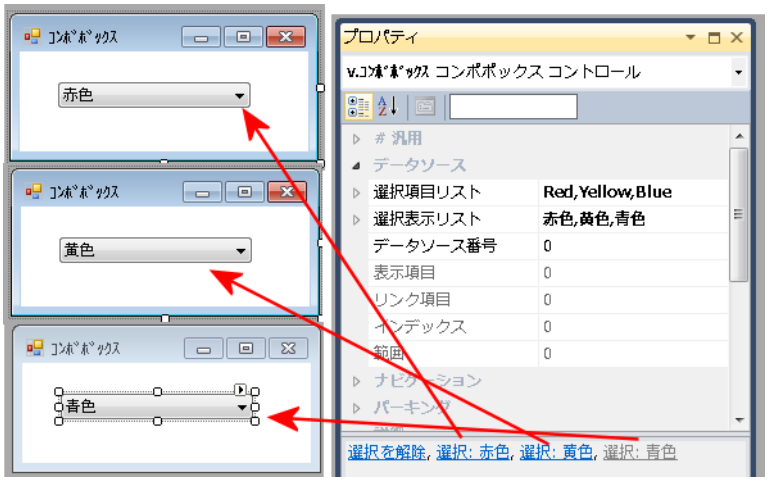
この例では、値が直接設定されていますが、式を使用したりデータソースを使用して設定することもできます。

チョイスコントロールに特殊文字を表示するには



いくつかの文字は、内部的にチョイスコントロールで使用されています。例えば、カンマは、選択項目を区切り文字として使用されています。リスト内でこれらの文字を使用するためには、エスケープ文字としてバックスラッシュ (¥) を前に付加する必要があります。ここで示されているように、空白、アンパサンド、およびカンマがこれに当てはまります。

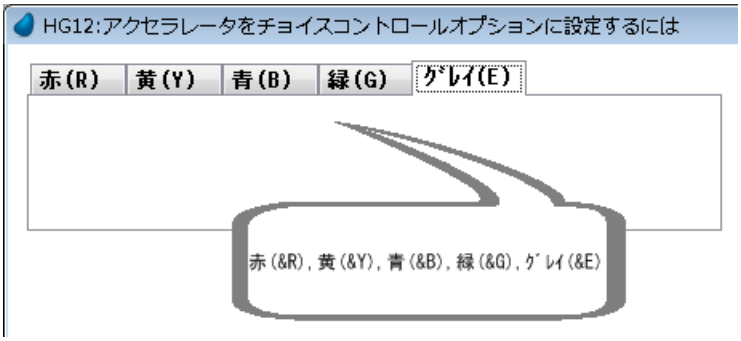
フォームデザイナ上でチョイスコントロールオプションを切り替えるには



コンボボックスコントロールとリストボックスコントロール、ラジオボタンコントロールの特性シートの下には選択：項目名が表示されます。これをクリックすることで選択項目の表示を切り替えることができます。

タブコントロールの場合は、タブの部分をクリックするだけで切り替えることができます。

アクセラレータをチョイスコントロールオプションに設定するには

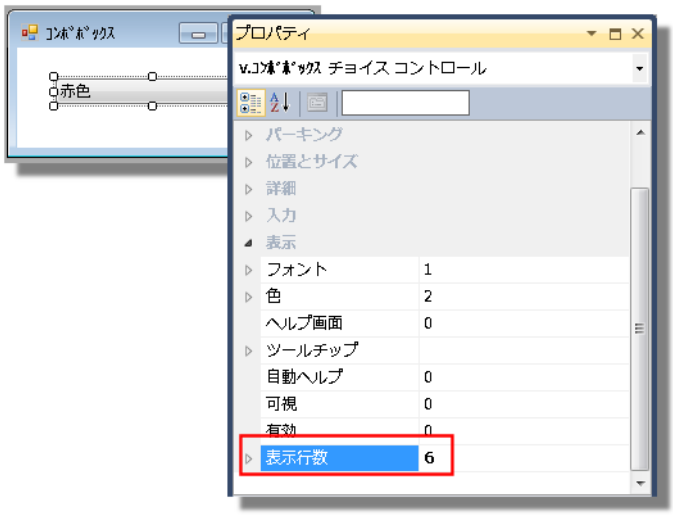


チョイスコントロール（タブコントロール）上の各項目は、アクセラレータキーを定義することができます。アクセラレータキーを設定すると、*Alt* キーを押下しながら該当キーを押下することで自動的にフォーカスをその選択項目に移動させることができます。上図の例では、*Alt* キーを押下しながら *E* を押下すると*グレイ*が選択されます。

アクセラレータキーは、表示文字の前にアンパサンド（&）を追加することで表示されます。

注： アンパサンドを表示する必要がある場合、その前にバックスラッシュ（¥）を付加してください。

コンボボックスに表示されるドロップダウンリストの長さを制限するには

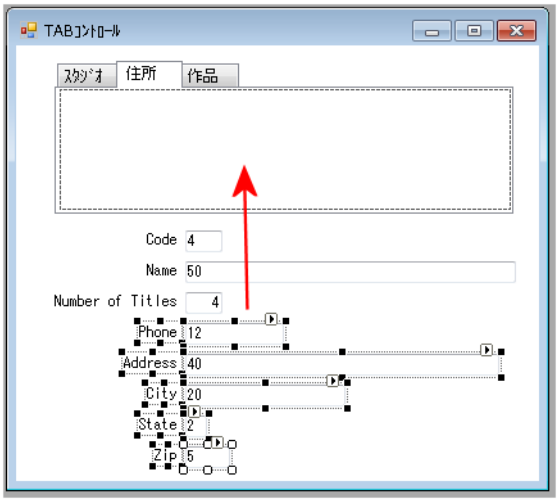


たくさんの選択肢が**コンボボックス**に定義されている場合があります。選択肢の表示行数を制限するには、**表示行数**特性に値を設定します。この例では、選択肢として最大 6 つの行が表示されます。

タブコントロールの異なるタブにコントロールを関連付けるには

タブコントロールを作成する場合、コントロールを 1 つのタブとリンクさせることができるます。リンクすると、そのタブが選択された場合のみ表示されます。

各タブでコントロールを表示させる

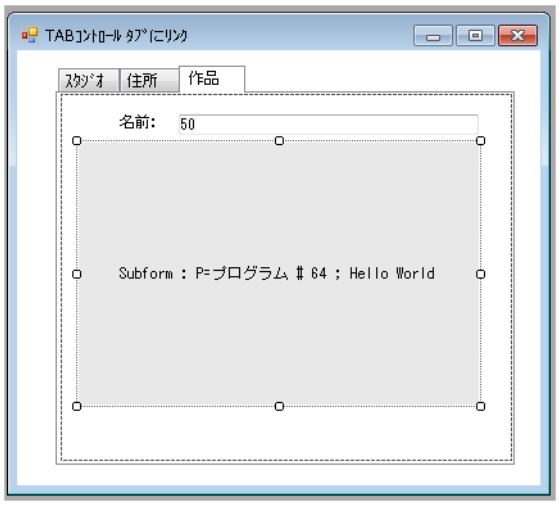


1. タブコントロール上のリンクさせたいタブをクリックして表示を切り替えます。
2. リンクさせたいコントロールを選択し、タブコントロール上に選択したコントロールを移動します。

これで、このコントロールは選択されたタブでのみ表示されます。この例では、アドレス項目はアドレスタブ上のみ表示されます。

タブコントロールの指定されたタブにタスクを関連付けるには

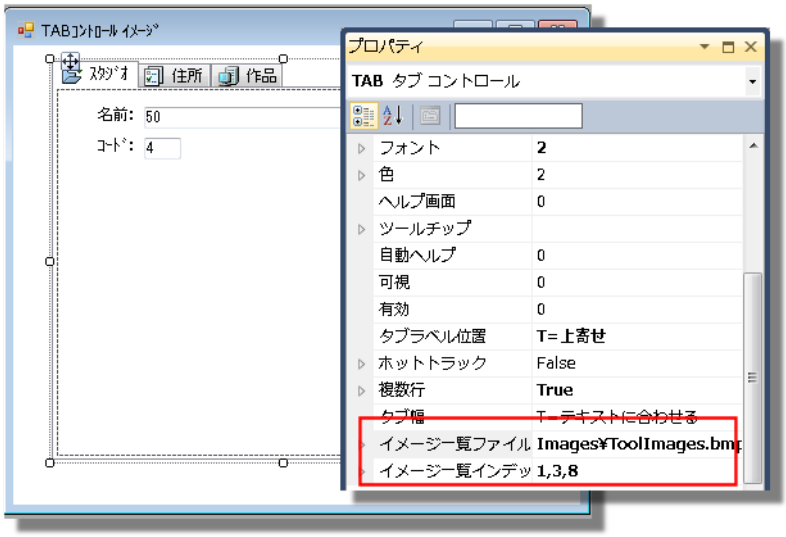
現在のタスクには定義されていないデータをタブ上に表示させる場合があります。表示されるデータがデータソースの内容であったり、別のレコードからの複合データの場合、特に必要になります。サブフォームコントロールをタブにリンクさせることでこのような処理を簡単に実現することができます。



1. タブコントロール上のリンクさせたいタブをクリックして表示を切り替えます。
2. ツールボックスペインからサブフォームアイコン  をクリックして選択します。
3. サブフォームをタブにドロップします。
4. タブフォームに実行したいタスク（プログラム）を定義します。

参照： 第 8 章 :「タブコントロールに配置したサブフォームの表示を制御するには」（189 ページ）

タブコントロールの各タブにイメージを割り当てるには



テキストと一緒にイメージをタブと結び付けることができます。

タブコントロールの**イメージ一覧ファイル名**特性を指定することで実現できます。このイメージファイルは、基本的に**ツリー**コントロールや**ボタン**で使用されるイメージを 1 つのファイルに連結したものです。このファイルは例えば、以下のような形式になります。

例の図のように、ファイルには、たくさんのイメージを定義することができます。各イメージのサイズは 16 X 16 ピクセルです。

イメージ一覧インデックス特性でどのイメージを表示させるかを指定することができます。この例では、３つのタブに対して 1、3、および 8 番目のイメージを指定しています。

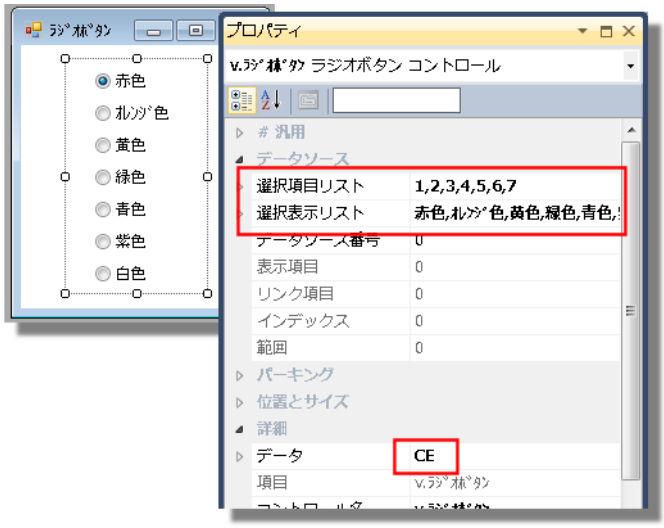
ラジオボタンを定義するには

２つの方法でラジオボタンを定義することができます。

1. 一般的な矩形表示コントロール内にボタンを表示する方法
2. 複数のコントロール内にボタンを含めるより柔軟な方法

２つのオプションの定義方法を以下に説明します。

1 つのコントロール内にボタンを含める

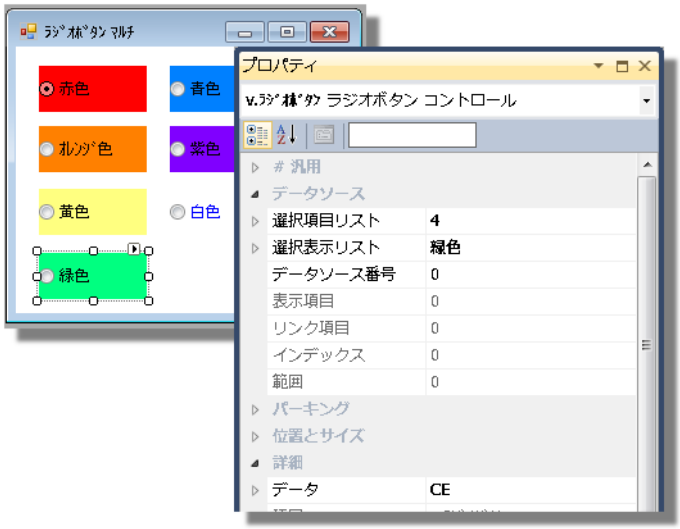


1 つのコントロール内にボタンを含めるラジオボタンを作成します。このタイプのラジオボタンは矩形のボックスで表示されます。作成するには以下のようにします。

1. 選択項目を格納する変数項目を作成します。この例では、項目 **CE** になります。
2. ツールボックスペインからラジオボタン  を選択し、フォームにドロップします。
3. **データ**特性に、変数項目を定義します。
4. **選択項目リスト**特性に、変数項目に設定したい値の選択肢を入力します。
5. **選択表示リスト**特性に、ラジオボタンに表示する選択肢を入力します。
6. **表示行数**や**フォント**などの表示特性を定義します。

これで実行時にラジオボタンから有効な値を選択することができます。

複数のコントロール内にボタンを含める



複数のコントロール内にボタンを含めるラジオボタンを作成します。このタイプはフォーム上の複数の部分にボタンを分けて表示させることができます。作成するには以下のようにします。

1. 選択項目を格納する変数項目を作成します。この例では、項目 CE になります。
2. **ツールボックス**ペインから**ラジオボタン**  を選択し、フォームにドロップします。
3. **データ**特性に、変数項目を定義します。
4. **選択項目リスト**特性に、変数項目に設定したい値の選択肢のうちの 1 つを入力します。この例では、*4* が設定されています。
5. **選択表示リスト**特性に、ラジオボタンに表示する選択肢のうちの 1 つを入力します。この例では、***緑色***が設定されています。
6. **表示行数**や**フォント**などの表示特性を定義します。

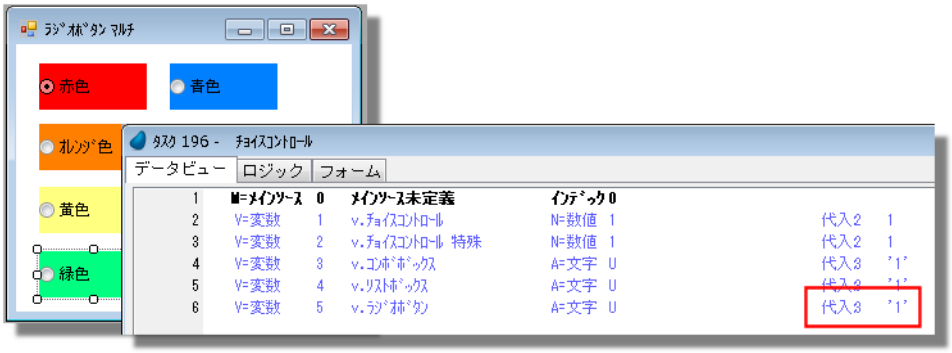
これで実行時に**ラジオボタン**から有効な値を選択することができます。

各々の**ラジオボタン**に同じ変数項目を定義することで、これらをつなぎ止める状態になり、これらは 1 つのユニットとして動作します。すなわち、実行中にボタンの 1 つを**クリック**すると以前の選択肢が解除され、1 つのボタンのように動作します。しかし、これらは個別のコントロールであり、個別に修正したり移動したりすることができます。

チョイスコントロールにデフォルトオプションを設定するには

チョイスコントロールを作成する場合、デフォルトでは何も選択されていない状態で、空白や何も選択されていない状態で表示されます。これでは、よく選択する項目を表示させたいユーザには不親切なアプリケーションとなってしまいます。

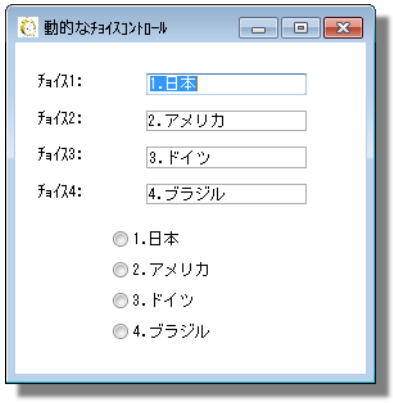
デフォルトの選択肢を指定する



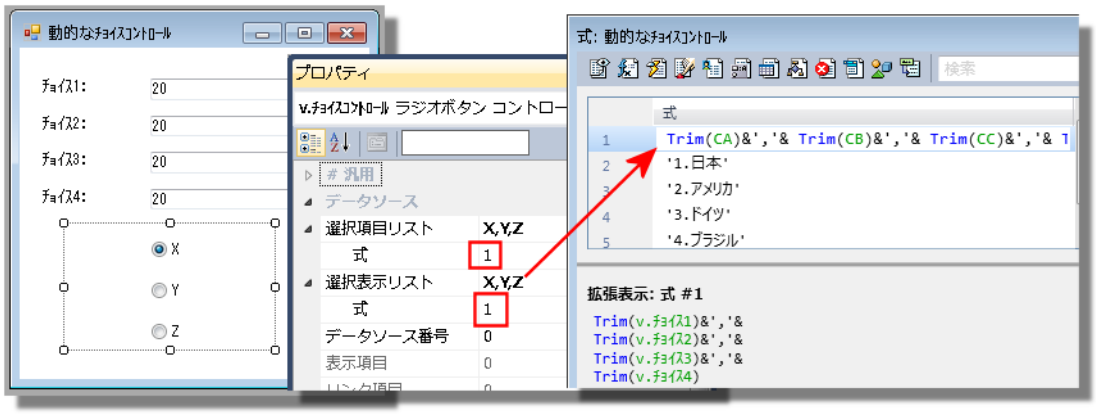
チョイスコントロールにデータ項目が定義され、その項目の初期設定ができれば、デフォルトオプションは簡単に設定できます。この例では、***赤色***の選択肢（**選択表示リスト**特性）は、*1*（**選択項目リスト**特性）とリンクしています。従って、**代入**特性に *1* を設定することで、***赤色***がデフォルトになります。

チョイスコントロールの選択肢を動的に設定するには

選択肢は固定して設定しますが、式を使用することで動的に指定することも可能です。この例では、動的な 4 つのユーザ入力をラジオボタンに設定しています。



動的な選択肢を設定する

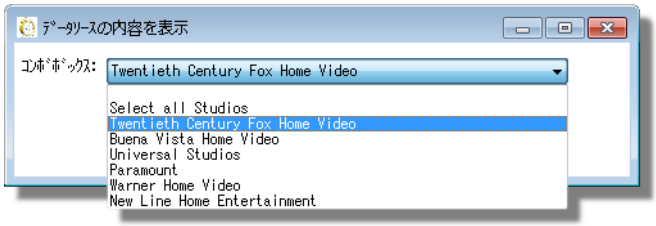


動的なチョイスコントロールを作成するには以下のようにします。

1. 選択項目を格納する変数項目を作成します。
2. ツールボックスペインからラジオボタン  を選択し、フォームにドロップします。
3. **データ**特性に、変数項目を定義します。
4. **選択項目リスト**特性を式で指定し、変数項目に設定したい値の選択肢を入力します。
5. **選択表示リスト**特性を式で指定し、ラジオボタンに表示する選択肢のうちの 1 つを入力します。この例では選択項目リストと同じになっています。
   オプションで、選択表示リストに文字列を入力することができます。この値は開発環境でのみ表示され、開発者が表示内容を確認しやすくなります。上記の例では、**x、y、z** が設定されています。

   これで、実行時に式で評価された値がチョイスコントロールの選択肢となります。

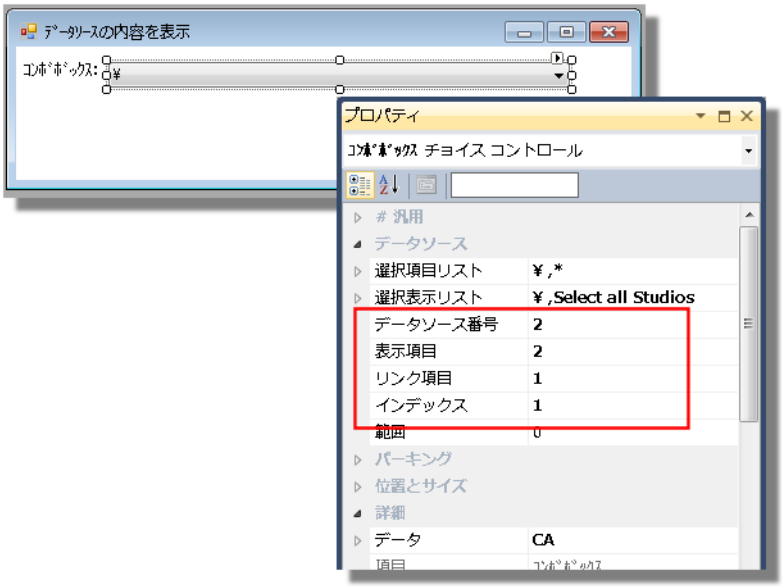
データソースの内容を選択肢にするには



データベーステーブルからデータを選択できるコントロールがあれば便利です。この方法を利用すると、テーブルが更新されると選択肢も自動的に変わります。

この例では、Studio テーブルからスタジオを選択肢しています。ユーザがスタジオ名を選択すると、タスク内ではスタジオコードが返るようになっています。これはどのようにして実現するかを説明します。

データベーステーブルにリンクしたチョイスコントロールを作成する



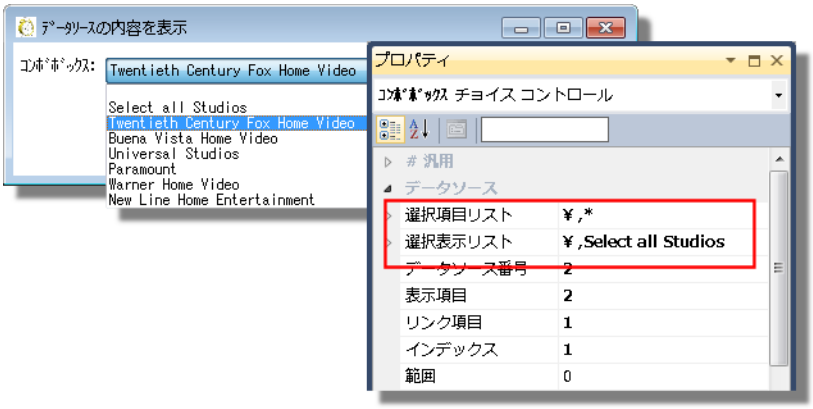
1. 最初に、データを格納する偏す項目を作成します。この例では、**v.Studio Code** を使用しています。
2. **ツールボックス**ペインから**コンボボックス**（または他の**チョイス**コントロール）を選択し、フォーム上にドロップします。
3. **チョイス**コントロールの**データ**特性に変数を定義します。（例では、***CA***）
4. **データソース番号**特性から**ズーム**（***F5***）して、使用したいデータソースを選択します。この例では、***Studios*** を使用しています。
5. **表示項目**特性から**ズーム**（***F5***）して、表示するカラムを選択します。この例では、***Studio Name*** を使用しています。
6. **リンク項目**特性から**ズーム**（***F5***）して、変数に値を返すカラムを選択します。この例では、***Studio Code*** を使用しています。
7. **インデックス**特性から**ズーム**（***F5***）して、選択肢の表示順を決めるインデックスを選択します。範囲指定が定義されている場合、範囲定義のインデックスと合うようにしてください。合わない場合、パフォーマンスに影響します。この例では、***Studio Code*** をインデックスにしています。
8. **範囲**特性から**ズーム**（***F5***）して、選択肢の範囲を制限する条件を指定します。この例では範囲指定を行っていません。

これで、タスクが実行されると、**チョイス**コントロールはテーブルから使用するデータ項目に移ります。

注： データソースが大きい場合、**チョイス**コントロールにリンクして使用すると性能が劣化する可能性があります。この方法は、比較的小さなデータソースまたは効率的なインデックスが定義されているデータソースで使用してください。また、**チョイス**コントロールにリンクされたデータソースは、宣言テーブルリストの中には含まれないことに注意してください。

データソースにリンクされたチョイスコントロールに追加オプションを結合するには

チョイスコントロールは、データをもとに選択肢を表示します。例えば、項目の範囲特性をもとに選択肢を表示したり、データソースに動的にリンクしている場合もあります。しかし、コンボボックスとリストボックスでは、範囲特性やデータソースの値に加え、より多くの値を選択肢として追加するオプションを持っています。



この例では、ソーステーブル **#2** にリンクされています（Studios テーブルです）。

1. **選択表示リスト**特性に２つの項目（***空白***と ***Select all studios*** という文字列です。）を追加します。空白は、バックスラッシュ（¥）を付加します。
2. **選択項目**リストにも、２つの項目（***空白***と ***アスタリスク***）を追加します。空白はリセットで、アスタリスクは全て選択することを表します。

参照： 「チョイスコントロールに特殊文字を表示するには」（274 ページ）

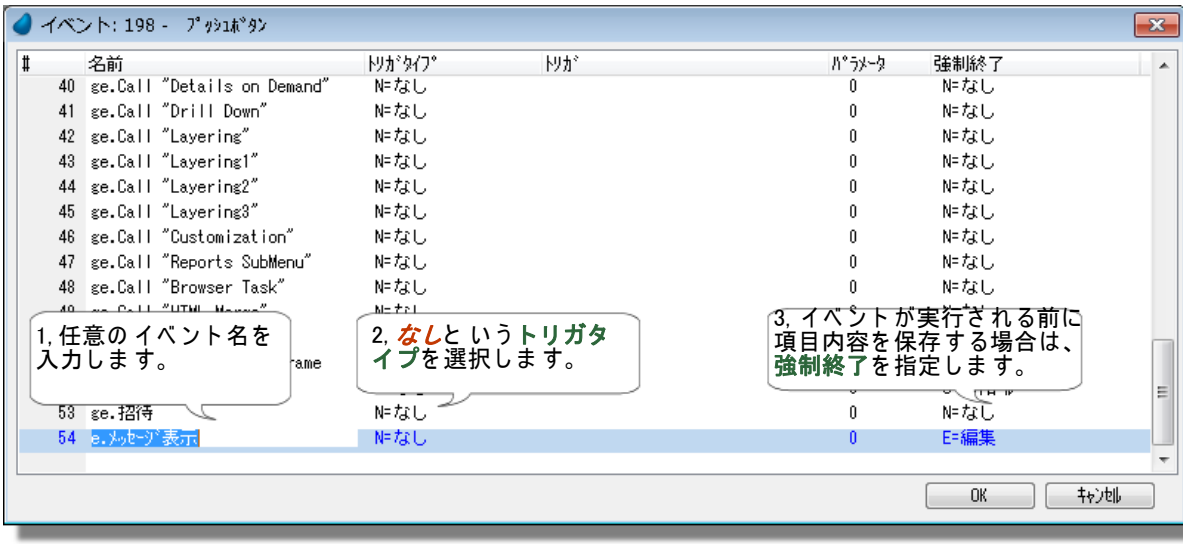
プッシュボタンとロジックを組み合わせるには

ボタンをクリックするとロジックが実行されるようにするには以下の手順で行います。

1. ユーザイベントを作成します。
2. ボタンコントロールをフォームに配置します。
3. イベントが実行されたときに処理するロジックユニットを作成します。

以下のセクションは、関連する手順を説明しています。

ユーザイベントを作成する

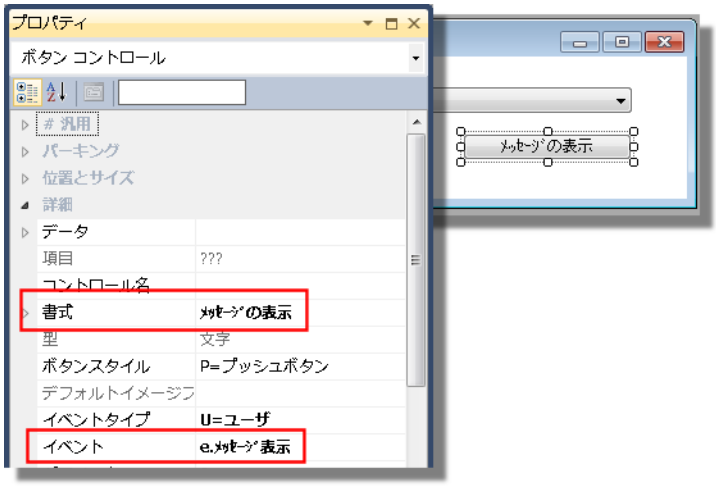


最初に、ユーザイベントを作成する必要があります。ユーザイベントテーブル（*Ctrl+U*）で作成します。1 行作成（*F4*）し、以下のようにパラメータを設定します。

1. 名前カラムに、*イベント名*を入力します。
2. トリガタイプでは、*N= なし*を選択します。このイベントがプッシュボタンで実行されるため、トリガは必要ありません。
3. 強制終了カラムでは、*E= 編集*または *N= なし*を選択します。*E= 編集*の場合は、プッシュボタンがクリックされてイベントが実行される前に、編集中のフィールドの値がデータ項目に格納されます。
4. ボタンコントロールに割り当てるイベントを必要なだけ追加します。

注： グローバルなイベントを再利用することもできます。上の例において、グローバルなイベントは ge.. いう接頭辞が付加されています。ここには、ge. 印刷 や ge. 実行などの多くのプログラムで使用する標準的なグローバルイベントが定義されています。グローバルイベントを使用する場合、自動的に指定されたイベントを実行するボタンを作成することができるように、これらをモデルに割り当てることができます。

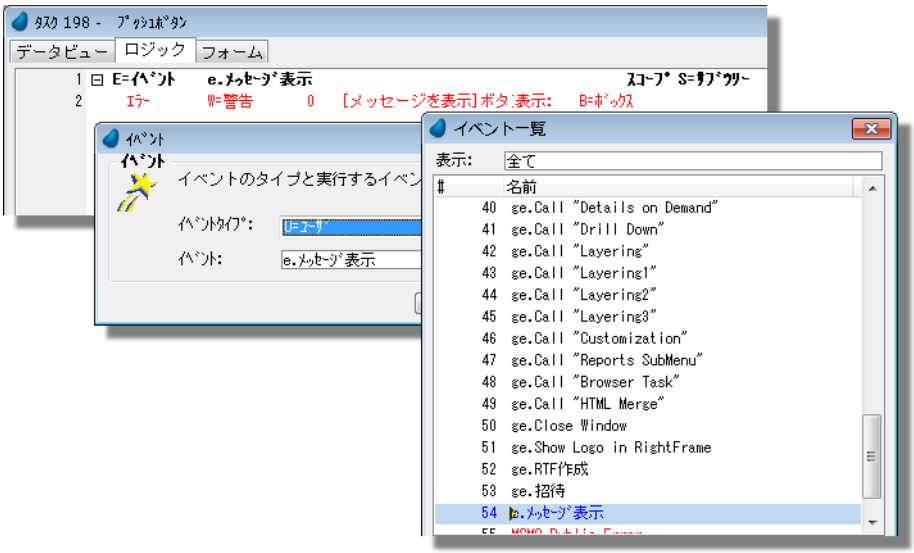
フォームにプッシュボタンを配置する



1. **ツールボックス**ペインから**ボタン**を選択し、フォームにコントロールをドロップします。必要に応じてサイズを変更します。
2. **ボタン**コントロールの**書式**特性を入力します。ここには、プッシュボタンに表示されるテキストを入力します。
3. **イベント**特性で**ズーム**（*F5*）してイベントを選択します。

   これで、ユーザが**プッシュボタン**を**クリック**するとイベントは実行されます。この例では、ユーザがメッセージの表示ボタンを**クリック**すると、**e. メッセージ表示**イベントが実行されます。次に、イベントを処理する**ロジックユニット**を作成する必要があります。

イベントが発行されると実行されるロジックユニットを作成する



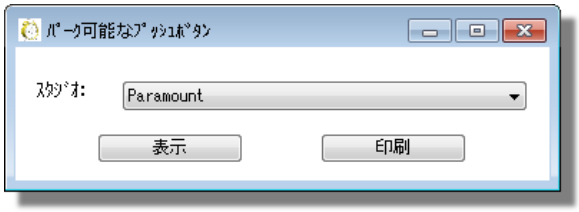
1. **ロジックエディタ**を開きます。
2. ヘッダ行（*Ctrl+H*）を作成します。
3. コンボボックスから、***E= イベント***を選択します。**イベント**ダイアログが表示されます。
4. **イベントタイプ**として、***U= ユーザ***を選択します。**イベント一覧**が表示されます。
5. 実行させたいイベントを選択します。
6. イベントが実行されたときに、実行される処理コマンドを**イベント**ロジックユニットに追加します。

   これで、実行可能な**プッシュボタン**が定義できました。

注： この例では、*Tab* で移動できない**プッシュボタン**を使用しています。**データ**特性に項目を定義することで、*Tab* 移動を可能にすることができます。項目を設定すると、**書式**特性はデータ項目の書式が設定されるため、後で修正する必要があります。

参照： 「プッシュボタンにキーボード操作を許可させるには」（289 ページ）

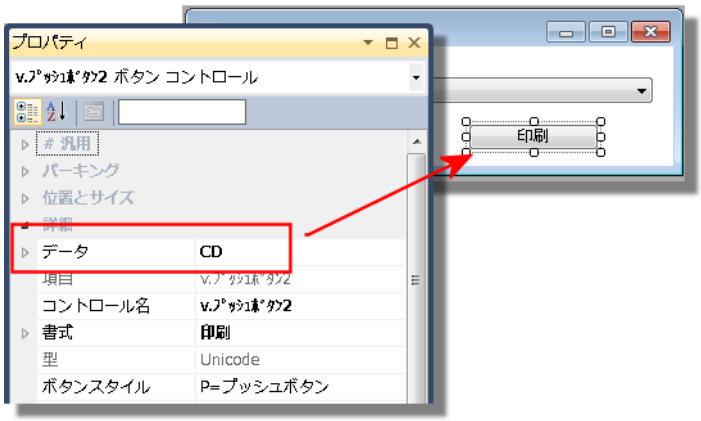
プッシュボタンにキーボード操作を許可させるには



ボタンコントロールをフォームにドロップし、**実行イベント**特性を設定するだけで**プッシュボタン**を定義することができます。ユーザがマウスで**クリック**するか、アクセラレータキー（ホットキー）を押下することで**プッシュボタン**は動作します。しかし、ユーザは *Tab* で移動することはできません。

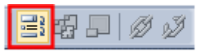
必要であれば、**ボタン**に項目を設定することで *Tab* による移動が可能になります。**ボタン**コントロールの**データ**特性に項目を指定するには、以下の操作を行います。

パーク可能なプッシュボタンを作成する



1. タスクの**データビューエディタ**を開き、**ボタン**に設定する変数項目を作成します。
2. フォーム上に**ボタン**をドロップします。
3. **データ**特性で**ズーム**（*F5*）して、この**プッシュボタン**に設定する変数項目を選択します。

   これでユーザは、この**プッシュボタン**に *Tab* でパークさせることができます。**ツールバー**から**タブオーダー**

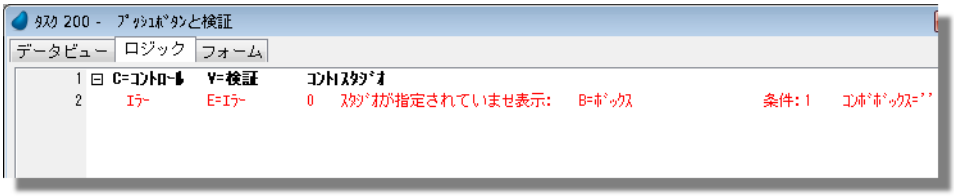
   

   をクリックすると、*Tab* 移動が可能で **TAB 順序**の変更が可能であることを示す数字が表示されます。

注： *Tab* 移動可能な**プッシュボタン**を作成すると、ボタン上のテキストは設定された項目に格納されているテキストが表示され、*Tab* 移動できない**プッシュボタン**とは異なって表示されるので注意してください。この対応には色々な方法があります。

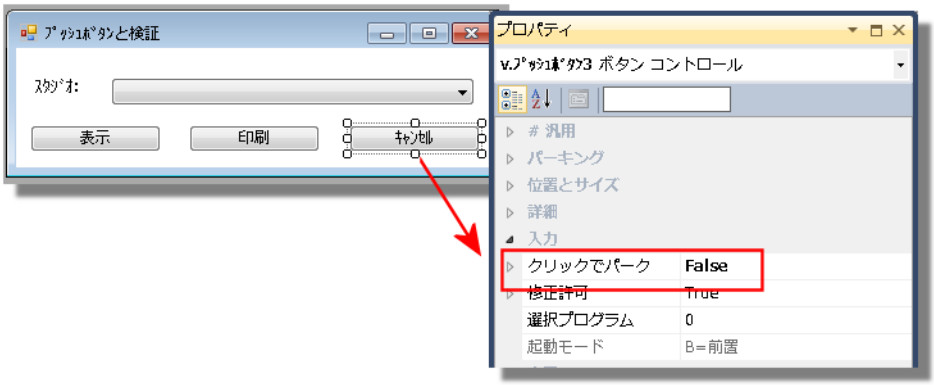
参照： 「パーク可能なプッシュボタン上にテキストを指定するには」（294 ページ）

プッシュボタンがクリックされたときに実行される検証ロジックをスキップするには



フォーム上のコントロールに対応する検証ロジックを定義する場合があります。この例では、スタジオを選択しないでスタジオコントロールを通過することができません。通常は、このような機能が必要となります。例えば、ユーザは、印刷データを選択しないで印刷することは無いはずです。

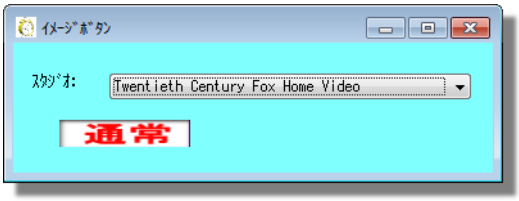
しかし、入力内容に問題があってもプッシュボタンを実行できるようにさせる必要があるかもしれません。例えば、終了やキャンセルのボタンを実行できるようにする場合が考えられます。



このような動作を実行させるには、**クリックでパーク**特性を *False* に設定します。プッシュボタンは設定前と同じように動作しますが、コントロールを通過してもコントロールロジックユニットは実行されません。

注： ユーザがキーボードを使用している場合は、効果がありません。従って、スタジオフィールドを空白の状態のまま *Tab* 移動しようとした場合、エラーメッセージが表示されます。このような場合、プッシュボタンにアクセラレータキー（*Alt+C*）を定義することで回避できます。

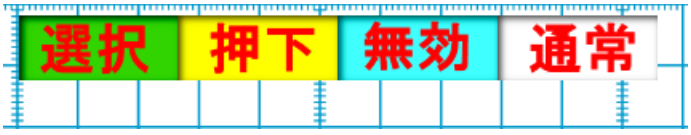
イメージボタンを作成するには



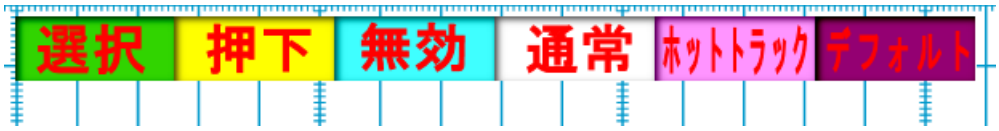
Magic xpa でイメージボタンを使用することができます。***イメージボタン***には、同じサイズの４つ（または６つ）のセクションを持つビットマップ（.bmp）ファイルを定義します。この例では、イメージエディタで 1 つのボタン用のイメージを作成し、同じイメージを繰り返しそれぞれ異なる色を設定しています。

実行時、ボタンの状態にもとづいて、Magic xpa はイメージファイルの異なる部分を使用します。パークされておらず有効な状態の場合、4 番目のボタン（白）が表示されます。ボタンが使用不可の場合、3 番目のボタン（空）が表示されます。ボタンがクリックされた状態の場合、2 番目のボタン（黄）が表示されます。（項目が割り当てられており）*Tab* 移動でフォーカスがコントロール上にある場合に、最初のボタン（緑）が表示されます。

６種類のイメージを使用している時は、ボタン上にマウスカーソルが通過した場合、５番目のボタン（桃）が表示され、フォームの**デフォルトボタン**特性に設定されている場合、６番目のボタン（紫）が表示されます。



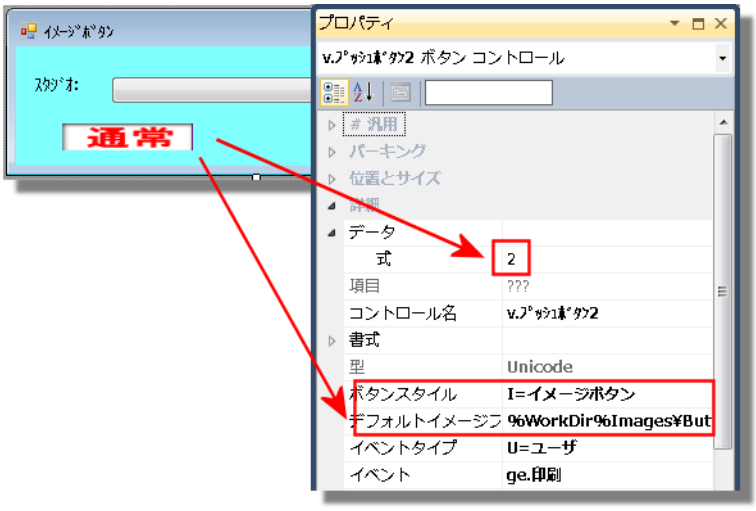
４つのボタンイメージの場合（PBImageNumber=4）



６つのボタンイメージの場合（PBImageNumber=6）

ボタンイメージの数は、**動作環境**の**動作設定→プッシュボタン用イメージ**で設定することで切り替わります。アプリケーション毎に表示数を切り替える場合は、**Iniput()** 関数で **PBImagesNumber** パラメータの値を設定してください。**Iniput()** 関数を実行した後でオープンされたフォームから有効になります。

イメージボタンを作成する



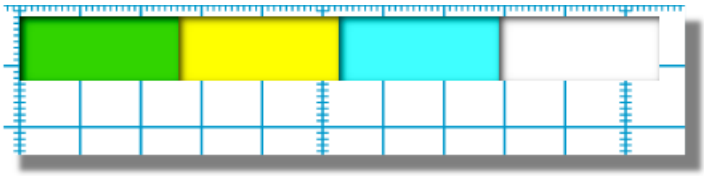
1. フォーム上に**ボタン**をドロップします。
2. **ボタンスタイル**特性で、***I= イメージボタン***を選択します。

3. **デフォルトイメージファイル**特性で**ズーム**（*F5*）して、イメージファイルを選択するか、直接ファイル名を入力します。ファイル名やパス名には論理名が使用できます。
   ここで設定されるイメージファイルは、**フォームデザイナ**上でのみ表示されます。
4. **データ**特性にイメージファイル名が設定されるデータ項目を定義するか、式でイメージファイル名を定義します。
   ここで設定されるイメージファイルは、プログラムの実行時に表示されます。

   これで、**プッシュボタン**を表示するために指定されたイメージファイルが使用されます。

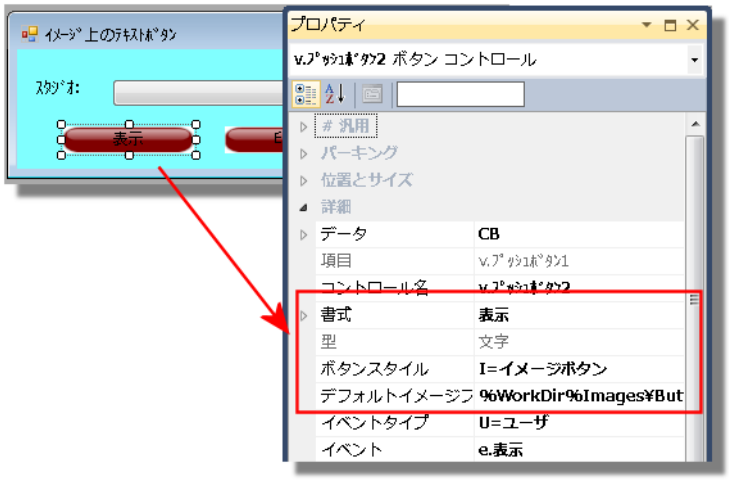
イメージとテキストを組み合わせてボタンを表示するには

Magic xpa では、ボタンに表示されるテキストの背景にイメージを指定することができます。これにより、作成するイメージファイルが少なくなり、多言語機能を利用することで実行時に別の言語表記に変換することが可能になります。



イメージ上のテキストで使用するイメージファイルは、イメージボタンと同じ書式で作成しますが、テキストは指定しません。イメージファイルの作成方法については、「イメージボタンを作成するには」（291 ページ）を参照してください。

テキスト上のイメージボタンを定義する



1. フォーム上にボタンをドロップします。
2. **ボタンスタイル**特性で、***I= イメージボタン***を選択します。
3. **デフォルトイメージファイル**特性でズーム（*F5*）して、イメージファイルを選択するか、直接ファイル名を入力します。ファイル名やパス名には論理名が使用できます。
   ここで設定されるイメージファイルは、フォームデザイナ上でのみ表示されます。
4. **書式**特性でボタンに表示するテキストを入力します。この例では、***表示***が入力されています。

   これで、背景にテキストが重なって表示されるプッシュボタンが表示されます。

パーク可能なプッシュボタン上にテキストを指定するには

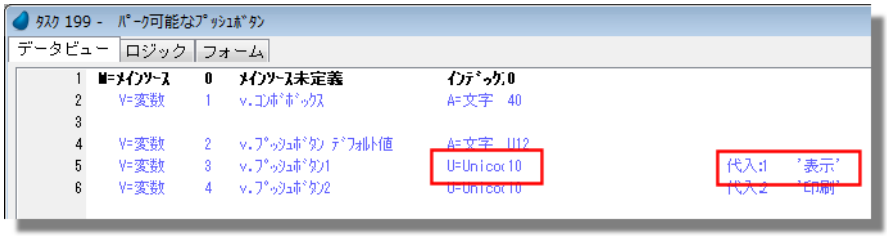
パーク不可の**プッシュボタン**を作成する場合、**ボタン**コントロールの**書式**特性にテキストを入力することで、ボタン上に表示されます。しかし、パーク可能な**プッシュボタン**の場合、データ項目を定義するため、項目の内容が**プッシュボタン**に表示されます。

従って、この種類のボタンでは以下のようにしてテキストを指定します。

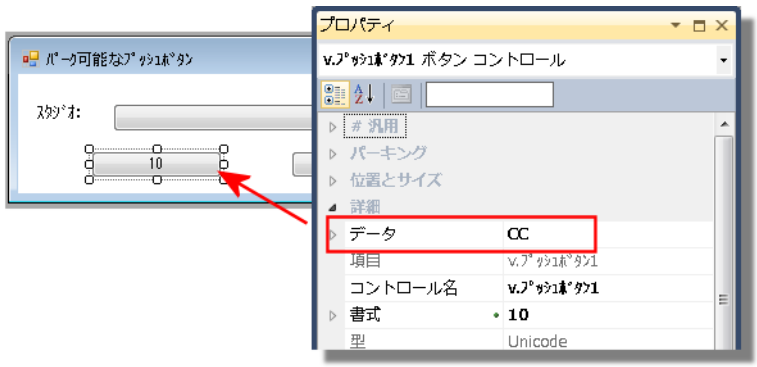
- 項目の**代入**特性で指定する。
- 項目の**デフォルト値**を指定する。
- **ボタン**コントロールの**書式**特性にテキストを指定する。

詳細は、以下のセクションで説明します。

プッシュボタンのテキストを指定するために代入を使用する

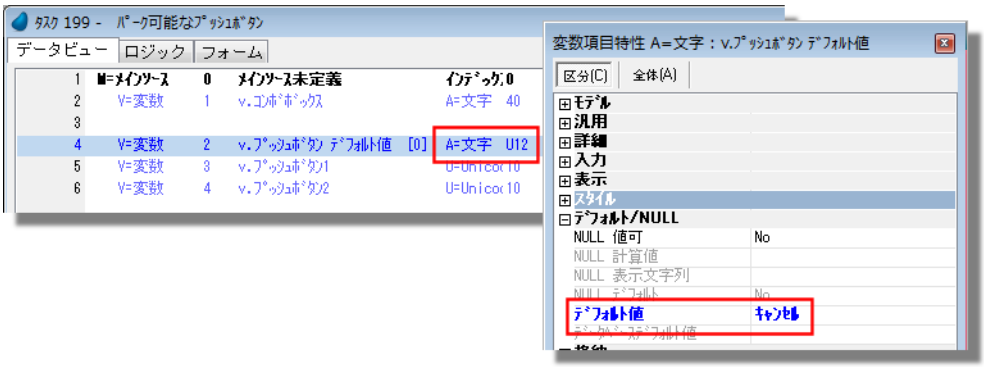


1. **プッシュボタン**のテキストが格納可能な長さの変数項目を作成します。
2. **代入**特性を使用してテキストを指定します。この例では、***表示***が設定されています。



3. フォーム上に**ボタン**をドロップします。
4. **ボタン**の**データ**特性に変数項目を定義します。
5. **フォームデザイナ**上では、このボタンには指定されたテキストが表示されません。**書式**特性の内容が表示されます。しかし、プログラムを実行するとテキストが表示されます。

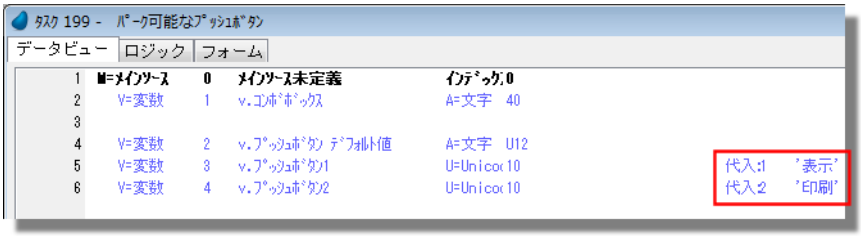
プッシュボタンのテキストを指定するためにデフォルト値を使用する



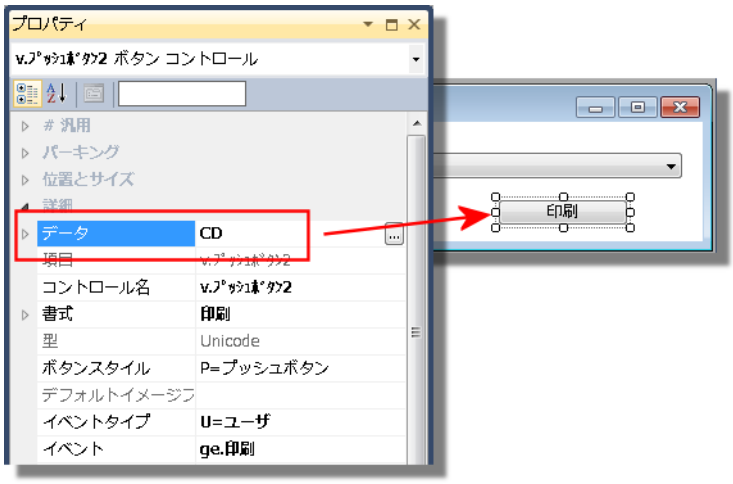
代入特性にテキストを指定することで、**プッシュボタン**にテキストが表示されます。しかし、**代入**特性の代わりに項目の**デフォルト値**特性にテキストを入力することでも同様に表示させることができます。

この方法は、モデルを使用する場合に便利です。しかし、**データビューエディタ**上でテキスト内容が参照できないため、ボタンに値が設定されているかどうかを確認できません。

プッシュボタンのテキストを指定するために書式特性を使用する

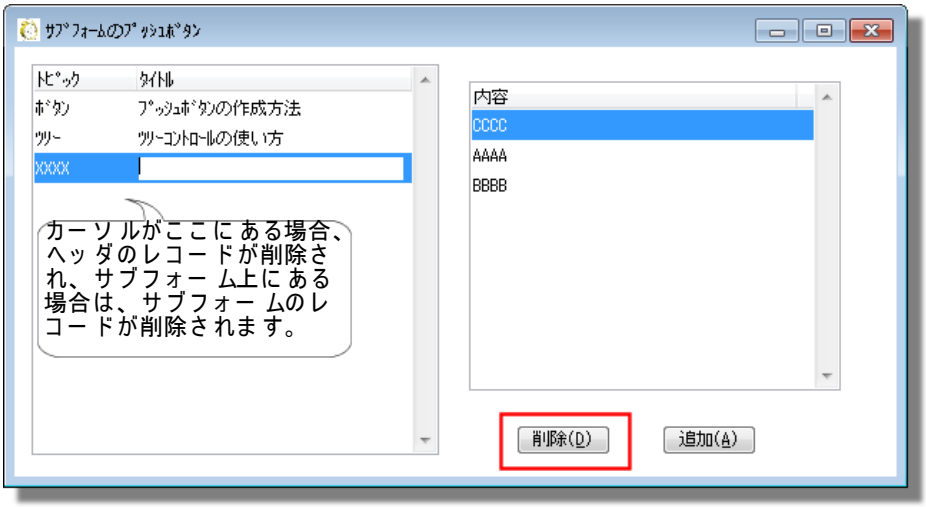


1. 文字型変数項目を作成します。
2. 少なくとも 1 つのプレイスフォルダ文字（この例では、U）を持つように、書式を指定します。残りの書式データをボタンに表示するテキストとして入力します。大文字は、前にバックスラッシュ（¥）を付加する必要があります。



3. フォーム上に**ボタン**を**ドロップ**します。
4. **ボタン**の**データ**特性に変数項目を定義します。
5. これで、**ボタン**コントロールの書式が変数項目から引き継がれるます。この方法で、**フォームデザイナ**上で、ボタンにテキストが表示されることを確認できます。

サブフォームやその親タスクに影響するプッシュボタンを定義するには



定義されたプッシュボタンに対して、現在のフォーカスがどこに位置しているかによって、異なる処理を実行させることができます。例えば、レコード一覧を表示させている親タスクがあり、各レコードが複数の子レコードを持っている場合、削除ボタンをクリックすると、カーソルがパークしている場所に応じて、親レコードや子レコードを削除するような処理が考えられます。

上の例において、削除ボタンがクリックされた場合、XXXX レコードを参照して処理すべきでしょうか？それとも CCC レコードを参照するべきでしょうか？ユーザは、カーソルが XXX の上にあるため、親レコードとその子レコードが全て削除されることを期待するはずです。しかしこのような場合は、カーソルがどこにパークしているかによって、異なる処理をさせる必要があります。

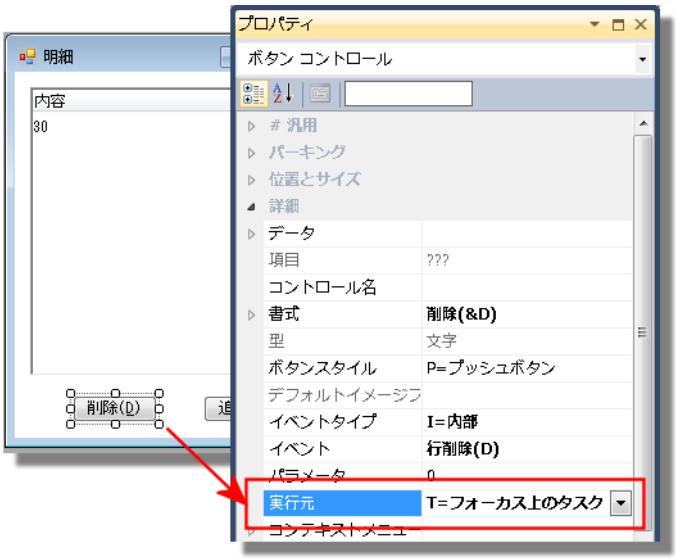
実行元特性は、この問題に対応するものです。この特性には、以下の２つのオプションがあります。

- *C= コンテナタスク*：イベントは、プッシュボタンが定義されているタスクで実行されます。
- *T= フォーカス上のタスク*：イベントは、フォーカスのある場所（タスク）で実行されます。

この例では、*T= フォーカス上のタスク*を設定する必要があります。

注： ***フォーカス上のタスク*のオプションは、パーク不可（*パーク可 =No*）のボタンでのみ有効です。**

実行元特性をフォーカス上のタスクに設定する



1. 変更が必要なプッシュボタンのコントロール特性を開きます。
2. **実行元**特性でズームして、*T= フォーカス上のタスク*を選択します。

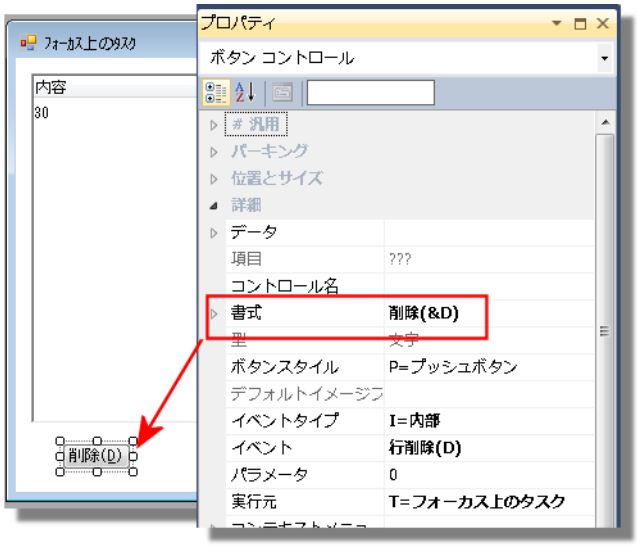
これで、現在フォーカスがどこにあるかによって、イベントが実行されるタスクが（親または子タスク）変わります。

アクセラレータをプッシュボタンに設定するには

大部分の人はマウスよりキーボードで操作するため、***アクセラレータ***（ホットキー）をプッシュボタンに設定すると便利です。パーク不可のプッシュボタンを使用している場合、特に必要です。アクセラレータを使用すると、ユーザはマウスからキーボードに手を移動する必要がなくなります。

Magic xpa ではアクセラレータを定義することができます。アクセラレータキーを設定するには、ボタンに表示するテキストの前にアンパサンド（&）を設定します。例えば、&C という文字が含まれるボタンは、C と表示され、*Alt+C* を押下することでそのボタンが押下されたことになります。

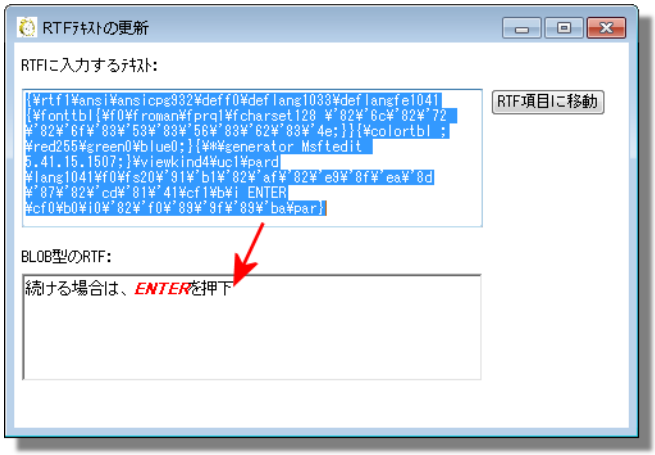
プッシュボタンにアクセラレータを設定する



1. コントロールの**書式**特性に移動します。
2. アクセラレータキーとして使用したい文字の前に & を挿入します。この例では、文字 D を使用しています。
3. 指定された文字にはアンダーラインが表示され、*Alt+ 文字*によってボタンが押下される動作になります。この例では、*Alt+D* がアクセラレータキーとなります。

注： 先頭文字は通常アクセラレータキーで、通常大文字で設定され、バックスラッシュを付加する必要があります。このためパーク可能なボタンを使用している場合、構文は多少異なります。上記の例では、U&¥ 削除となります。

動的なリッチテキストを作成するには



リッチテキスト（RFT）は基本的に HTML または XML のようなテキストマークアップ言語です。従って、フォーマットされたテキストを項目に格納することができ、指定されたフォントや色によって表示することができます。

ユーザは、コンテキストメニューを使用して実行時に RTF フィールド内のテキストを編集することができます。ほとんどのアプリケーションでは、フォーム上に RTF フィールドが定義されており、ユーザがデータのフォーマットを修正できるようになっています。もしフィールドの初期設定が必要であれば、他の文字型項目で行うように、**代入**特性や**デフォルト値**特性にテキストを設定することになります。

しかし、この例のように予めフォーマットされたテキストでフィールドを初期設定することができます。ここでは、テキストフィールド内に格納された RTF テキストを使用しています。

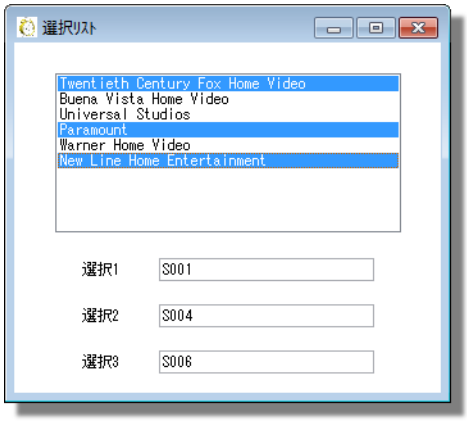
ユーザがボタンを**クリック**すると、**項目更新**処理コマンドによって BLOB 項目の値が更新されます。

テキストを BLOB 項目に格納するために、特別な作業は必要ありません。**項目更新**処理コマンドか**代入**特性を使用したり、**データ**特性に式を定義することで実現できます。

フォーム上に BLOB を表示させるには、**リッチエディット**を選択するか、GUI 表示形式の**スタイル**特性に ***I= リッチエディット***と定義した BLOB 項目をフォームに配置します。

ヒント : *リッチテキストを使用するには、このフォーマットのについて深く理解する必要はありません。Magic xpa で RTF スタイルの BLOB 項目を編集し、* **Blb2File()** *関数でテキストファイルに保存するだけです。この結果はカット＆ペーストで使用することができます。*

複数選択のリストボックスからデータを取得するには



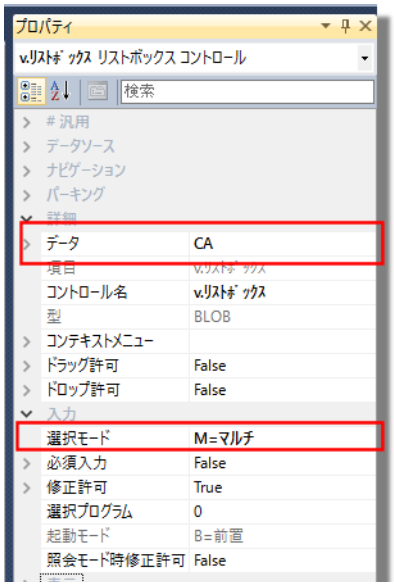
リストボックスから一度に複数の項目を選択でさせることができます。複数選択のリストボックスは単一選択と同じように設定しますが、以下の特性の値が異なります。

- **選択モード**：*M= マルチ*
- **データ**：ベクトル型の項目

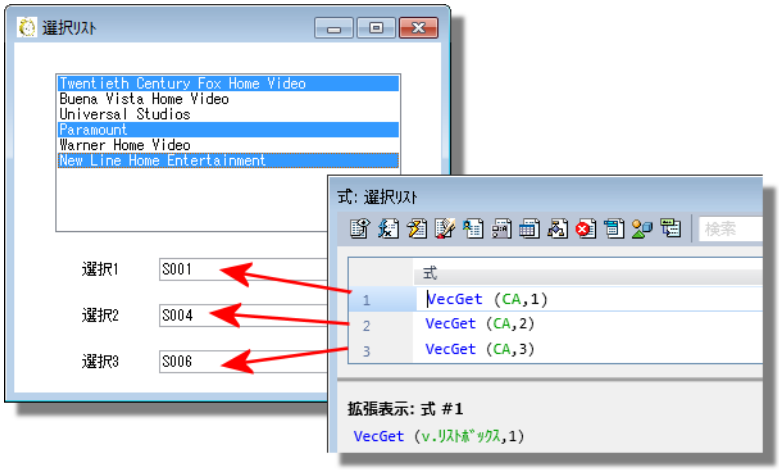
以下のセクションでは、設定例を説明しています。

複数選択のリストボックスを使用する

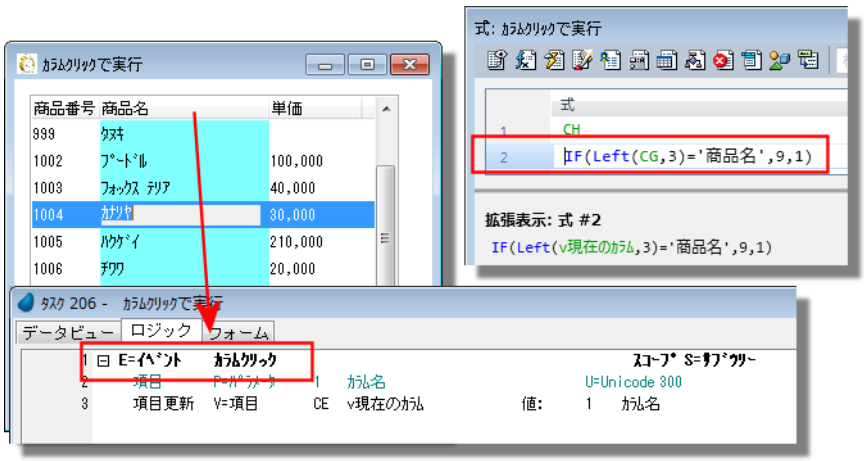
1. **データ**特性には、ベクトル項目を設定します。この例では、テキストを格納する簡単な文字型ベクトル項目を使用しています。
2. **選択モード**特性を複数に設定します。



これで、**VecGet()** 関数を使用して更新されたベクトル項目の内容を取り出すことができます。



ユーザがカラムをクリックした後にロジックを実行させるには

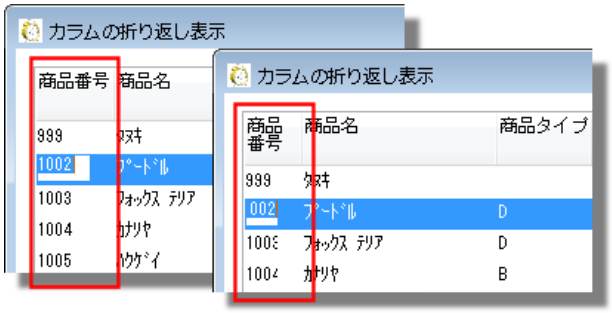


ユーザがカラムヘッダをクリックした直後に、ロジックを実行したい場合があるかもしれません。この例では、「現在のカラム」が異なる色で表示されています。このようなロジックは、ソート処理や範囲指定のダイアログ表示をカスタマイズする場合に使用することができます。これは、**カラムクリック**という内部イベントを使用することで実現できます。

カラムクリックイベントには 1 つのパラメータがあります。ここには、カラムの名前が渡されます。この例では、カラム名パラメータの内容が変数項目に格納され、色を設定するために、**カラム**コントロールの**色**特性で使用されています。色の変更は、クリックした即時に反映されます。

注： カラムデータは、透過色が指定されています。

カラムヘッダを折り返し表示させるには

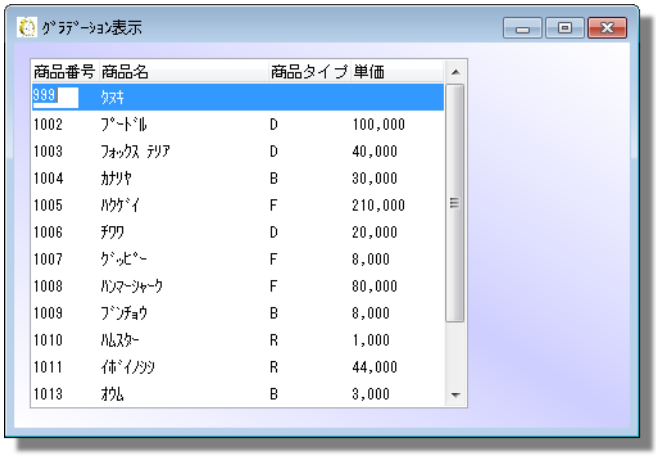


カラムヘッダに表示させる内容が、実際のカラム幅より長い場合があります。この場合、2 行以上に渡って折り返し表示させることですべて表示させることができます。折り返し表示がインタラクティブに行われた方が都合がいいはずです。もし、ユーザがカラムやテーブルの幅を変更したことで表示幅が小さくなった場合、折り返し表示する必要がでてきます。

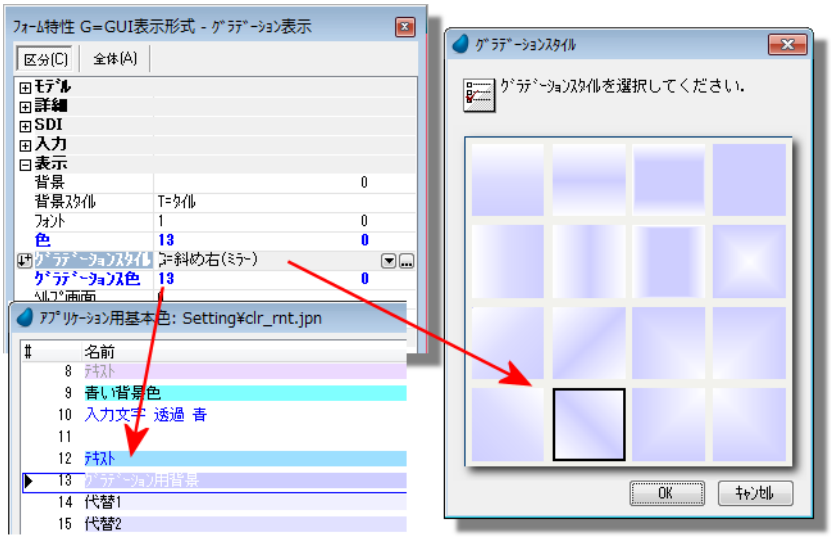
これは、Magic xpa で簡単に実現することができます。

1. テーブル特性で、テキストを 2 行以上表示させるための**タイトル高さ**を設定します。
2. 各カラムのカラム特性で、**垂直整列**特性を ***T= 上寄せ***に設定します。
3. 各カラム名において、テキストが折り返し表示できるだけのスペースがあることを確認してください。

グラデーション表示を定義するには



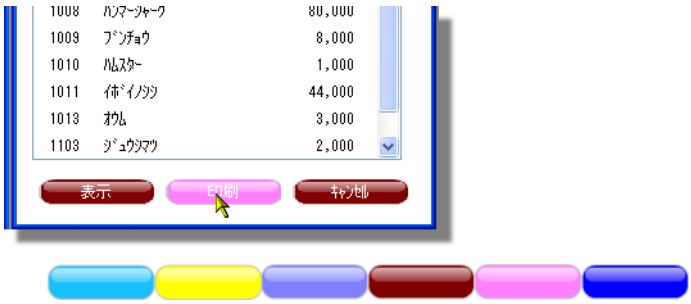
色彩効果を高めた表示を定義するために、グレデーションを指定することができます。これらは、いくつかのコントロール（例えばフォームの背景やスタティックコントロール）で使用することができます。



1. **オプション→設定→基本色**を選択し、グラデーションで使用する色を定義します。**前景色**はグラデーションの一部分として使用され、**背景色**は他の部分で使用されます。色テーブルに既に定義された様々な色を使用することができます。しかし、将来のメンテナンスのために、グラデーション用に使用する色を別途定義しておくことを推奨します。
2. **フォーム**または**コントロール特性**において、**グラデーションスタイル**を***なし***以外の値に設定してください。
3. **グラデーション色**特性に表示させたい色を定義します。

設定内容は、即時にフォームに反映されます。このため、簡単に確認することができます。

プッシュボタンにホットトラック効果を定義するには



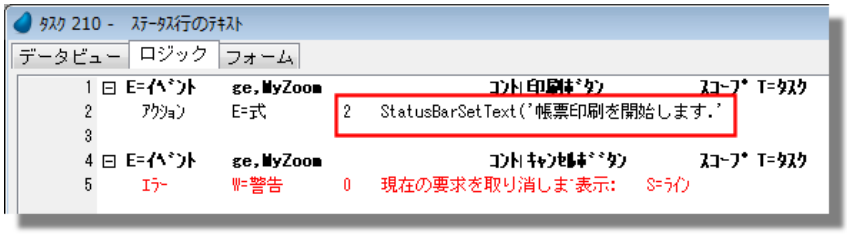
マウスがプッシュボタン上でホバリング状態になっていることを表示するホットトラックボタンを使用することで、使いやすくすることができます。

1. **オプション→動作環境→動作設定→プッシュボタン用イメージ数**を ***6*** に設定します。
2. 画像編集ツールを使用して、6 つのボタンイメージを 1 つに結合した画像を作成します。各イメージがボタンの状態を表しています。5 番目の状態がホットトラックを示しています。
3. ボタンイメージをビットマップ（BMP) ファイルで保存します。保存方法は、使用するツールに依存します。
4. **ボタン**コントロールの**ボタンスタイル**特性は、***イメージボタン***を選択します。
5. **デフォルトイメージファイル名**特性では、作成したボタンイメージを選択します。

注： アプリケーションが四分割のイメージを使用する既存のイメージボタンを使用するようになっている場合、それらを特定して、ボタン用のイメージファイルを変更する必要があります。

ステータスバーにテキストを設定するには

エンドユーザにメッセージを伝えるには、色々な方法があります。エラー処理コマンドは、ポップアップボックスや、ステータス行でメッセージ表示するために使用されます。しかし、エラー処理コマンドは、不快な音も鳴らすため、頻繁に実行すると、イライラさせることがあります。

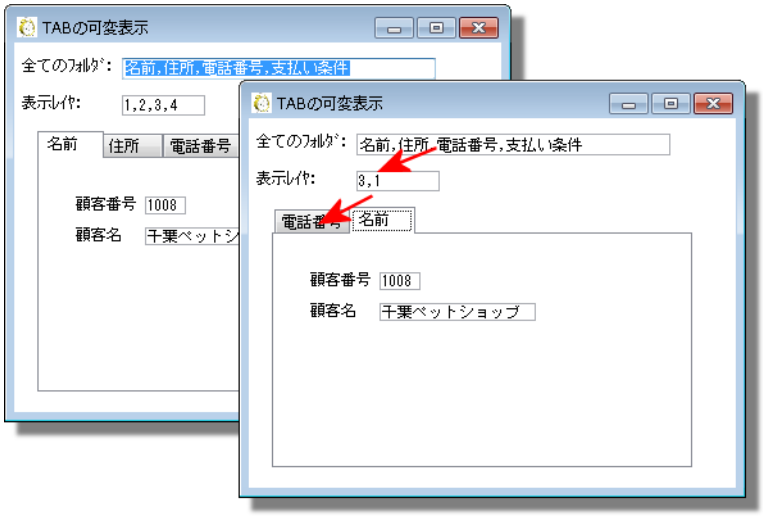


したがって、単にステータス行にメッセージを表示させたいだけであれば、**StatusBarSetText()** 関数を使用してください。

この例では、**StatusBarSetText()** 関数がアクション処理コマンド内で使用されています。

注： フォームのウィンドウタイプ特性を ***SDI*** に設定した場合、この例のように個別のウインドウのステータス行に表示されます。デフォルトに設定した場合は、MDI のステータス行に表示されます。

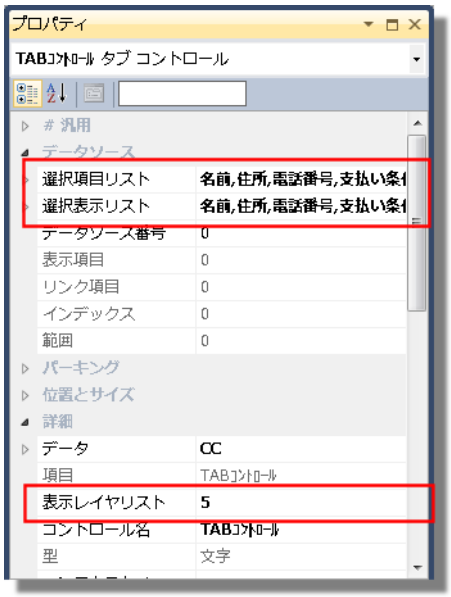
タブコントロールのタブ順序と可視を動的に定義するには



タブコントロールは、2 つの主要な特性から成り立っています。

- **選択項目リスト**……タブが選択されたときに実際に項目に反映される値を設定します。
- **選択表示リスト**……エンドユーザに表示する値を設定します。

しかし、データが 1 つのタブにリンクされている場合、これらの値を動的に変更することは容易でありません。したがって、動的にタブの表示順序を変更する必要がある場合、3 番目の特性として**表示レイヤリスト**特性を使用します。



表示レイヤリストは、数値を並べた文字列が定義された式番号を設定します。各々の数値は、**選択項目リスト**または**選択表示リスト**の 1 つの選択肢に対応しています。この例では、名前は 1 番目の選択肢 1、電話番号は、選択肢 3 に対応しています。

この特性を使用して、タブの可視と順序を簡単に定義することができます。

［ブラウザ］コントロールで動作している Web ページと連動するには

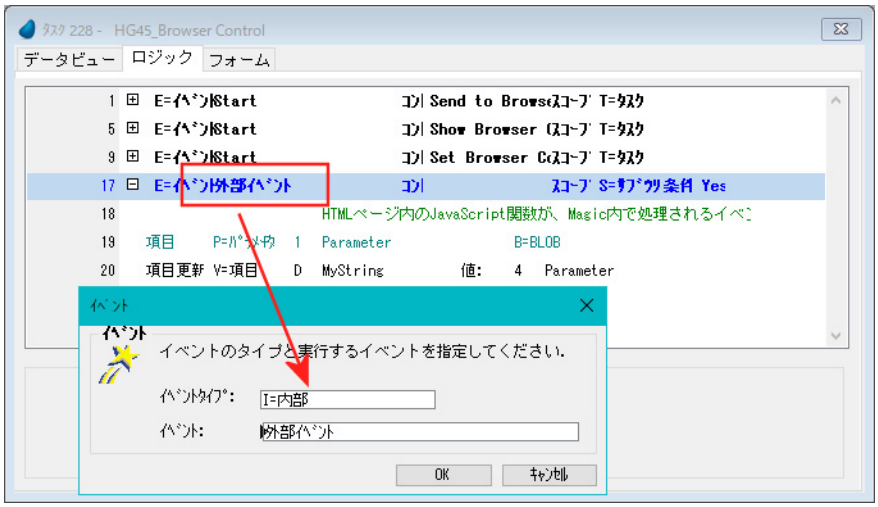
［ブラウザ］コントロールで Web ページを表示しているとき、その Web ページから情報を送信させたり、Web ブページに情報を送信したりすることができます。

情報の送信

Web ページは、MGExternalEvent と呼ばれる特別なイベントを使用することで、プログラムに情報を送信することができます。これは、Web ページのコードにコード化されています。例えば、Magic xpa に日付の引数を送り返す VBScript は以下の通りです：

```
<script LANGUAGE=ÅhVBScriptÅh>
// This VBScript segment handles the click event of the Calendar object.
// Upon a click on the object the MGExternalEvent function is called with
//   the expected arguments.
// This allows external modules on the page to interact with the Magic xpa
//   engine.
Sub Calendar1 click()
call window.MGExternalEvent(
  document.Calendar1.Day & Åe|Åf
  document.Calendar1.Month & Åe|Åf
  document.Calendar1.Year   )
end Sub
</script>
```

これにより MGExternalEvent が発生し、3 つの日付の値が '|' で区切られた 1 つの文字列で送信されます。これは、1 つのパラメータである BLOB のみを返すことができるため重要です。



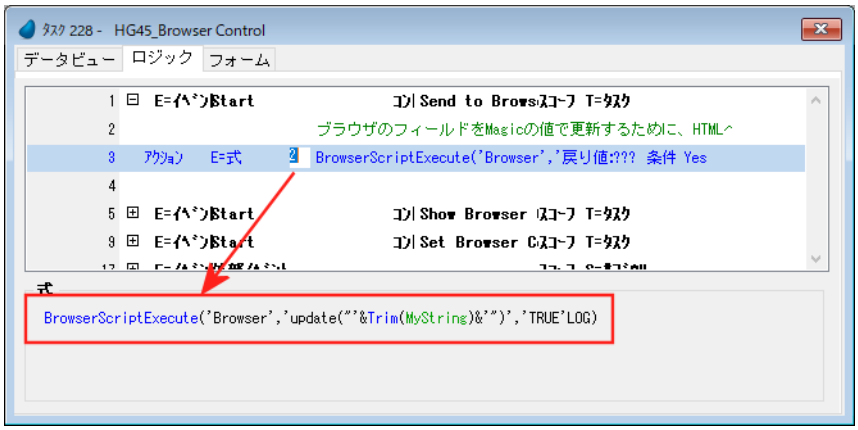
プログラムでは、このイベントを処理する必要があります。これは、［外部イベント］という内部イベントに対する［イベント］ロジックユニットを作成することで実現します。

次に、送信された内容に従って BLOB を解析するだけです。この例では、BLOB はテキストなので、文字として定義され、BLOB 上で直接 DVal 関数を使用することができます。

Magic xpa のさまざまな文字列解析関数（StrToken() や変換関数など）は、このような解析に非常に便利です。

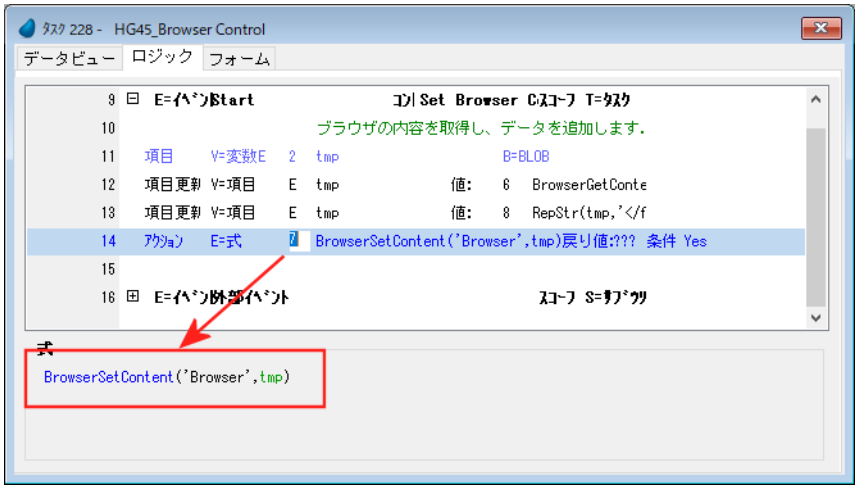
ブラウザーにデータを送る

Web ブラウザにインタラクティブにデータやコマンドを送信することもできます。



その一つの方法として、BrowserScriptExecute() 関数を使用します。これは、ブラウザ内でスクリプトを送信して実行するものです。

デフォルトでは JavaScript が使用されますが、他の言語も指定することができます。



また、BrowserSetContent() 関数を使用することで、ページのコンテンツ全体を変更することができます。これは、［ブラウザ］コントロールが表示しているすべての HTML を置き換えます。同様に、BrowserGetContent() は、ページの全コンテンツを返します。

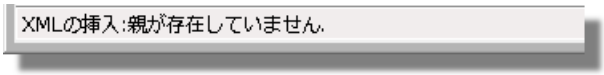
注： .NET のブラウザコントロールの System.Windows.Forms.WebBrowser を使用することもできます。これは、他の［.NET］コントロールと同様に、より低レベルのコントロールが可能ですが、学習曲線は高くなります。

［このページは意図的に空白にしています。］

第 13 章 : XML

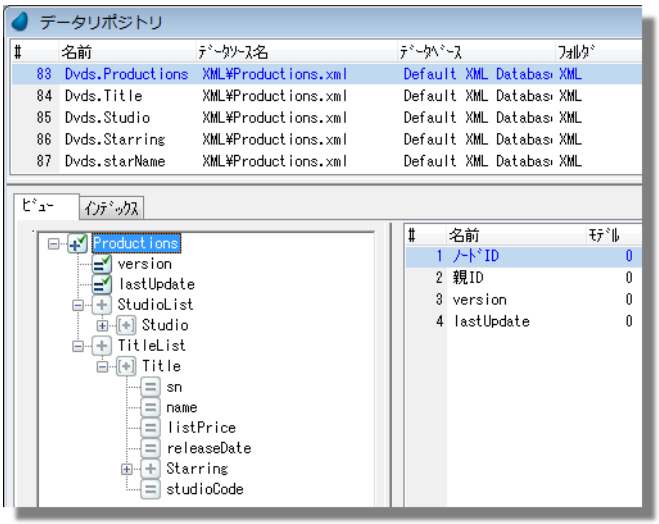
最初から XML ドキュメントを作成するには

XML ビューを定義することで、他のデータソースのように XML ドキュメントを処理することができます。XML ドキュメントは、平面的に見えますが全て階層構造になっています。また、すべての XML ファイルは、ルートディレクトリで初期設定を行う必要があります。初期設定しない状態でデータを入力しようとするとステータス行に以下に表示されているようなエラーメッセージが表示されます。



XML ドキュメントを初期設定する

例えば、この例のように、Productions.xml ファイルの中には３つのビューがあります。すべて同じ XML ドキュメントをアクセスしていますが、階層の異なる部分がそれぞれ異なるビューとして扱われます。現在のビューで使用されている部分は、対応する項目上で**緑色**のチェックマークが表示されます（右図を参照）。



この XML ファイルにタイトルを書き込みたい場合は、最初にデータソース（Products）のルートディレクトリを作成する必要があります。現実的には、タイトルをデータソース **#84** に追加する前にデータソース **#83** に 1 つのレコードを書き込むことを意味しています。XML ファイルがまだ存在していない場合、ルートを作成するために書込リンクや簡単なバッチタスクを使用することができます。

データリポジトリで **APG** （***Ctrl+G***）を使用してデータ参照したり、レコードを作成することでこの処理を確認することができます。

ルートレベル（**#83**）でレコードを作成すると、より低いレベル（**#84** と **#85**）でレコードを作成することができ、レコードが正しく挿入されます。しかし、逆の場合は動作しません。

XML スキーマを見つけるには

XML ドキュメントを作成するには、最初に XML スキーマを準備する必要があります。スキーマはデータベース定義（データベース内の項目の書式を定義するもの）のようなものです。以前に XML を扱ったことがない場合は、事前に勉強しておくことをお勧めします。

XML ドキュメントを読み込むように依頼されているのであれば、スキーマはすでに存在するであろうし、それを使用することになります。他の人に渡すために XML ドキュメントを作成している場合、手動でスキーマを設計するか、*XML Spy* などの外部の製品を使用することになります。スキーマは、***.xsd*** という拡張子を持ったテキストファイルです。

どちらの場合も、スキーマが存在しているのであれば、Magic xpa で簡単に XML 定義を作成することができます。

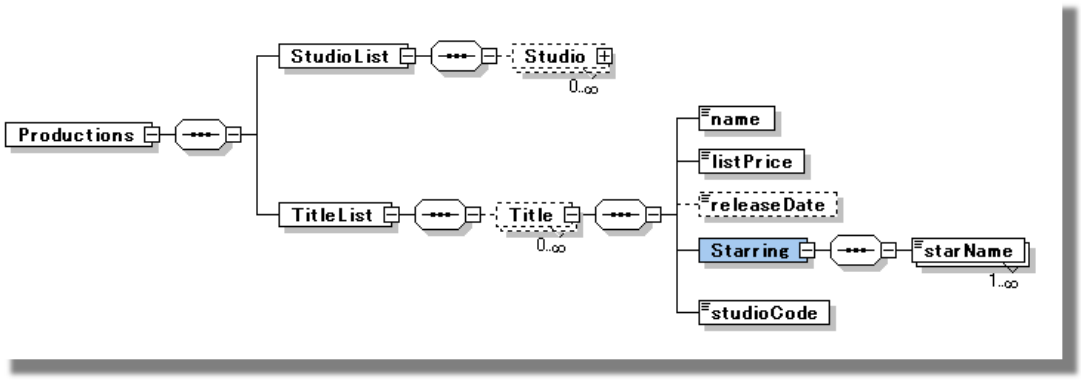
スキーマは複合的なデータの関係を記述しています。本質的に、スキーマはデータベース全体や複数に分離された構成内容、ネストされたテーブルについて記述することができます。データソースとしてこれらを使用するには、スキーマの一部を選択し、それらを **XML ビュー**と呼ばれる個別ののフラットなデータソースとして記述する必要があります。

XML スキーマ

ここでは、以下のスキーマファイルを使用しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
   <xs:element name="Productions">
      <xs:complexType>
         <xs:sequence>
            <xs:element name="StudioList">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="Studio" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="name" type="xs:string"/>
                           </xs:sequence>
                           <xs:attribute name="code" type="StudioCodeType" use="required"/>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
            <xs:element name="TitleList">
               <xs:complexType>
                  <xs:sequence>
                     <xs:element name="Title" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                           <xs:sequence>
                              <xs:element name="name" type="xs:string"/>
                              <xs:element name="listPrice" type="xs:float"/>
                              <xs:element name="releaseDate" type="xs:date" minOccurs="0"/>
                              <xs:element name="Starring">
                              <xs:complexType>
                              <xs:sequence>
                              <xs:element name="starName" type="xs:string" maxOccurs="unbounded"/>
                              </xs:sequence>
                              </xs:complexType>
                              </xs:element>
                              <xs:element name="studioCode" type="StudioCodeType"/>
                           </xs:sequence>
                           <xs:attribute name="sn" type="xs:string" use="required"/>
                        </xs:complexType>
                     </xs:element>
                  </xs:sequence>
               </xs:complexType>
            </xs:element>
         </xs:sequence>
         <xs:attribute name="version" type="xs:int" use="required"/>
         <xs:attribute name="lastUpdate" type="xs:date" use="required"/>
      </xs:complexType>
   </xs:element>
   <xs:simpleType name="StudioCodeType">
      <xs:restriction base="xs:string"/>
   </xs:simpleType>
</xs:schema>
```

このスキーマの内容を図式化すると以下のようになります。



また、表にすると以下のようになります。

要素	属性	書式
Productions	**Complex type**	
StudioList	**Complex type**	
Studio	**Complex type**	
Name	Simple Type	xs:string
Code	Simple Type	StudioCodeType
TitleList	**Complex type**	
Title	**Complex type**	
name	Simple Type	xs:string
listPrice	Simple Type	xs:float
releaseDate	Simple Type	xs:date
Starring	**Complex type**	
starName	Simple Type	xs:string
studioCode	Simple Type	StudioCodeType
sn	Simple Type	xs:string
version	Simple Type	xs:int
lastUpdate	Simple Type	xs:date
StudioCodeType	Simple Type	

項目のいくつかは **Complex type** でそれ以外は **Simple type** です。Complex type は、テーブルの行（レコード）のようなものです。Simple type は、テーブルのカラムのようなものです。

従って、Complex type に対して個々のデータソース定義を作成します。この例では、Complex type の Studio に対してデータソースを作成します。

Simple type は、通常 type= という定義を含んでいます。これは、データが string、float、integer または他のスキーマ要素（StudioCodeType）で記述されたものです。Magic xpa が、データソース内で各項目に対してデフォルトデータ型を作成する場合に使用されます。

参照： 「スキーマを使用しないで XML ドキュメントを処理するには」（331 ページ）

XML ビューを作成するには

XML ビューは、Magic xpa で XML を使用して処理するためのキーとなるものです。いったん **XML ビュー**が作成されると、他のデータソースと同じように XML ドキュメントを使用することができます。

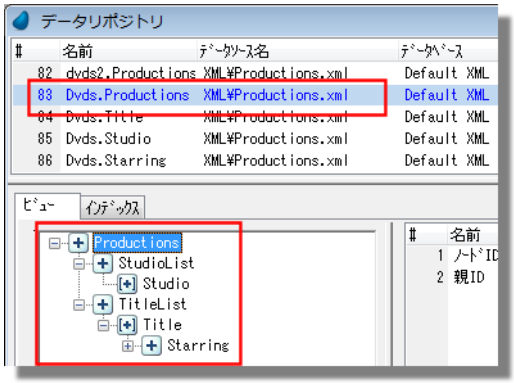
必要条件： 以下が必要になります。

- **データベース**テーブルに、***XML File*** というタイプが定義されたデータベース
- XML スキーマ（「XML スキーマを見つけるには」（310 ページ）を参照してください）。

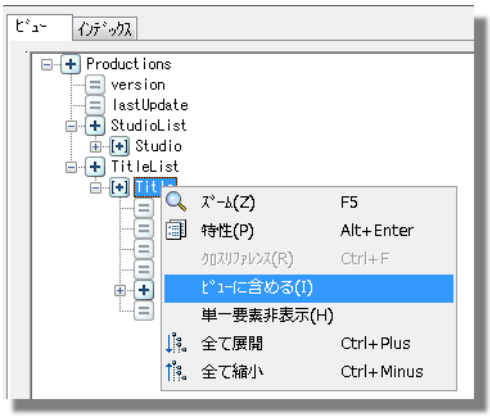
XML ビューを作成する

1. **データ**リポジトリに 1 行追加します（***F4***、または**編集→行作成**）。
2. **データベース**カラムで、タイプが ***XML File*** に設定されているデータベースを選択します。

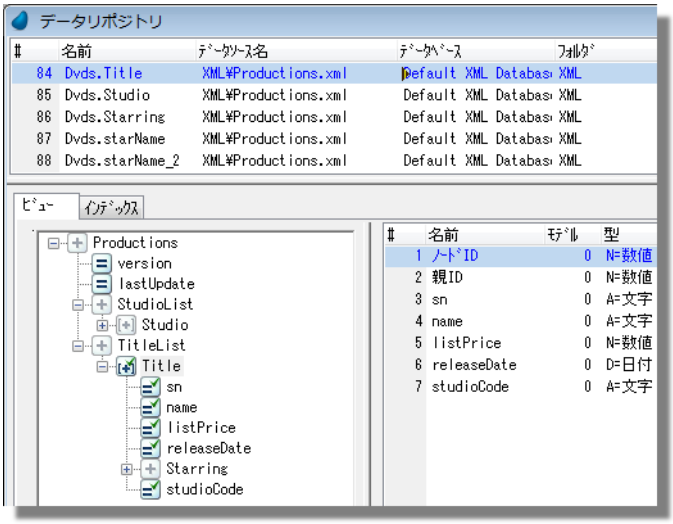
スキーマ情報を取得するために ***F9***（**オプション→定義取得**）を押下し、**ファイル選択**ダイアログを開きます。
スキーマファイル（***.xsd***）を選択すると、**名前**と**データソース名**のカラムにスキーマの最初の要素名がデフォルトとして設定されます。この例では、Productions です。



3. **ビュー**タブには、***.xsd*** 内の情報をもとに自動的に値が設定されます。
   複合要素（+ アイコンを持つもの）のみ参照できます。また単一要素を表示するには、コンテキストメニューから**単一要素表示**を選択してください。右の図では、単一要素が表示されます（メニューは**単一要素非表示**に変わります）。



4. タイトル要素内にビューを作成した場合、タイトル要素に移動し、コンテキストメニューの**ビューに含める**を選択します。これで、**ノード ID** と**親 ID** によってルート要素とリンクされるサブ要素を持つことになります。



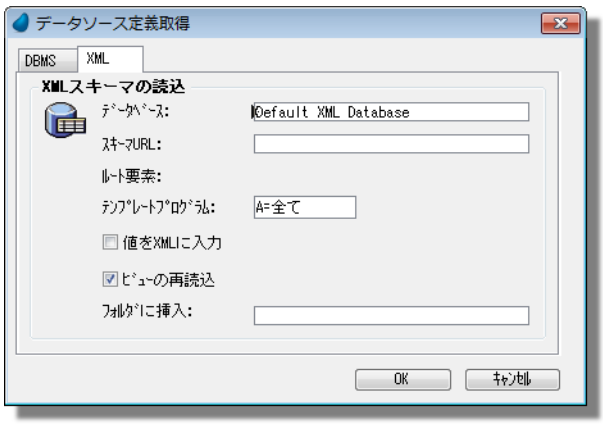
5. データが格納される実際の XML ドキュメントの名前を反映するために、**データソース名**を変更します。パス名を直接記述しないで論理名や相対パスを使用することを推奨します。
6. この **XML ビュー**を呼び出す際に必要な**名前**を（必要であれば）変更します。

これで他のデータソースと同じように、TitleList にアクセスすることができます。

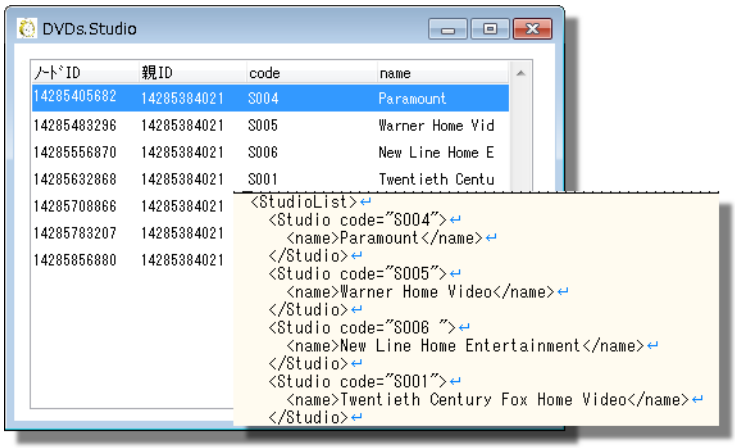
注： XML ドキュメント内の他の要素についても同じように **XML ビュー**を作成する必要があります（「XML ファイル内の合成要素にアクセスするには」（315 ページ）を参照してください）。また、XML ドキュメントに書き込む前にルート要素を初期化する必要があります（「最初から XML ドキュメントを作成するには」（309 ページ）を参照してください）。

複数の XML ビューを作成する

1. **データ**リポジトリの**タイトル**（#）行に移動し、**定義取得**ダイアログ（*F9* または**オプション → 定義取得**）を開きます。
2. **XML** タブに切り替えます。
3. **データベース**欄で、XML ファイルのデータソース・タイプに定義されたデータベースを設定します。
4. スキーマを選択して、OK をクリックします。新しいビューが、**データ**リポジトリに追加されます（また、任意でプログラムが**プログラム**リポジトリに追加されます）。



ノード ID と親 ID を更新するには



更新はできません。**ノード ID** と**親 ID** は Magic xpa 内部で使用され、プログラムで更新しようとしても無視されます。

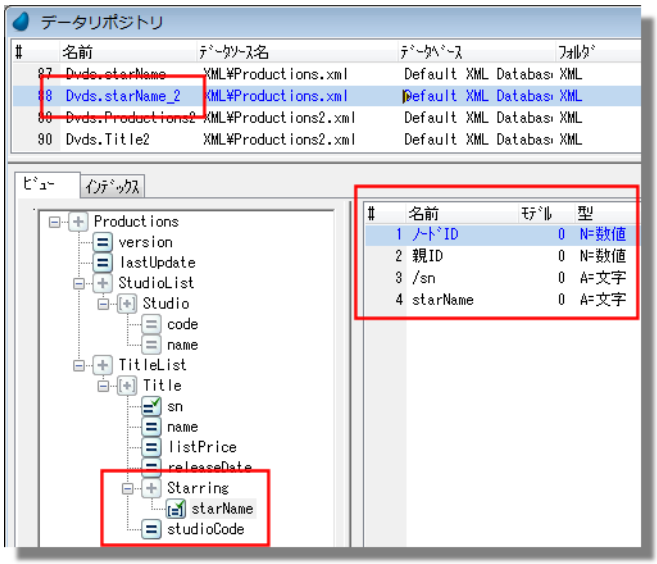
この例のように、実際のドキュメント内に明示的に保存されませんが、XML ドキュメントがオープンされると、繰り返し要素内の各レコードはユニークな**ノード ID** が設定されます。**親 ID** は XML ドキュメント内の親要素を参照します。この例では、ルートノードになります。

しかし、XML プログラムの作成にはこれらのことを理解する必要がありません。**ノード ID** と**親 ID** は無視できます。

XML ファイル内の合成要素にアクセスするには

XML ドキュメントを使用するということは、定義上、階層型データベースを使用することになります。一般的に、XML ドキュメントは複数のレベルを持っています。これらが一緒に保存されていても、SQL や ISAM テーブルの階層的なデータのように処理されます。

繰り返し要素



この例では、Productions という XML ドキュメントには複数の繰り返し要素が定義されています。

- ルートレベル：Productions は 1 レコードのみ持っています。
- Productions 以下
- StudioList- 複数の要素があります。
- TitleList- 複数の要素があります。
- TitleList 以下
- Starring list- 複数の要素があります。

各レベルに対して 1 つの **XML ビュー**が必要です。Magic xpa は、このような設定にもとづいてレコードが書き込まれると、階層を正しく維持するための処理を実行します。

合成要素を選択する

1. 含めたい単一要素上の 1 つのレベルであるノードを選択します。
2. コンテキストメニューから、**ビューに含める**を選択します。
3. 子レコードのために、リンクフィールドも含めるようにします。「既存の XML ドキュメントを修正するには」（316 ページ）を参照してください。

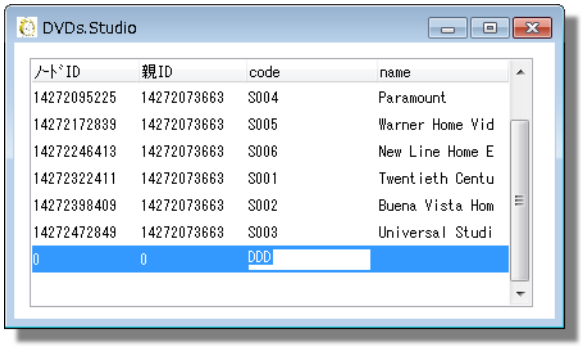
これで、単一要素は **XML ビュー**にフィールドとして表示され、Magic プログラムからアクセスすることができます。詳細は、「XML ビューを作成するには」（312 ページ）を参照してください。

注：　単一要素は、要素名の前に = アイコンが表示され、複合要素は + アイコンが表示されます。このため単一要素と複合要素は区別することができます。単一要素がビュー上に表示されない場合、コンテキストメニューから単一要素表示を選択します。

既存の XML ドキュメントを修正するには

いったん XML ドキュメントが作成され、**XML ビュー**が設定されたら、ISAM や SQL テーブルのようにアクセスできます。基本的なレコードの処理は自動的に行われます。

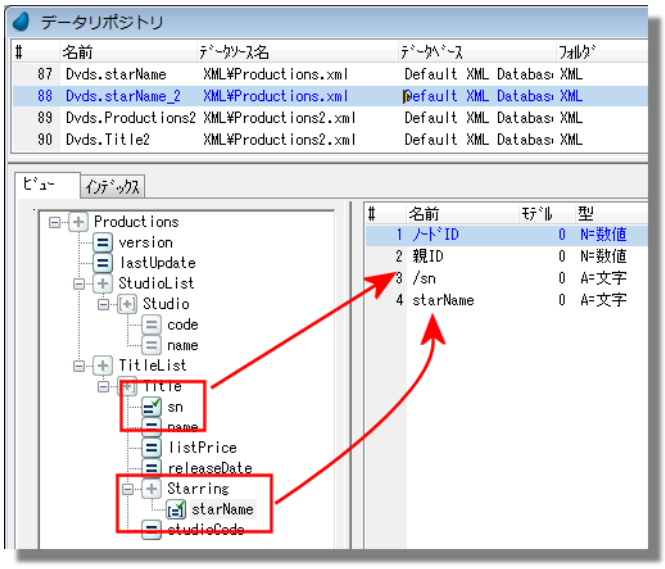
親レコードにアクセスする



他のテーブルと同じように親レコードを使用することができます。また、登録、修正、削除の処理を行うこともできます。この例では、リッチクライアントのプログラムを作成するために、**APG**（***Ctrl+G***）を実行し、テストレコードを追加しています。ノード ID や親 ID に対しては何も行いません（「ノード ID と親 ID を更新するには」（314 ページ）を参照してください）。

ただし、先頭のレコードは、XML ドキュメントのルートディレクトリではありません。この例では、Studio データ（これは ISAM または SQL の DBMS では子レコードとして考慮されません）にアクセスしていますが、XML ドキュメント内では、ProductctionsXML ドキュメントの 1 つの要素になります。レコードを追加する前に、ルートディレクトリ Productions を書き込まなければなりません。「最初から XML ドキュメントを作成するには」（309 ページ）を参照してください。

子レコードにアクセスする



さて、子レコードを作成する際、親とリンクさせるにはどのようにすればいいでしょうか？

この場合、２つの手順で実現することができます。

最初に、**XML ビュー**を作成している時に、子レコードに対してビューにリンク項目を含めます。

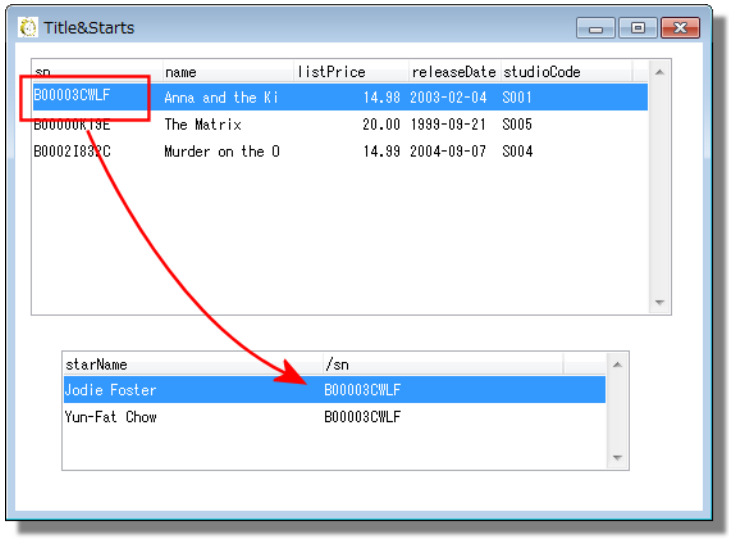
以下の手順で使用したいリンク項目を選択します。

1. リンクで使用する親レベルの要素上にカーソルを置きます。
2. コンテキストメニューから、**ビューに含める**を選択します。

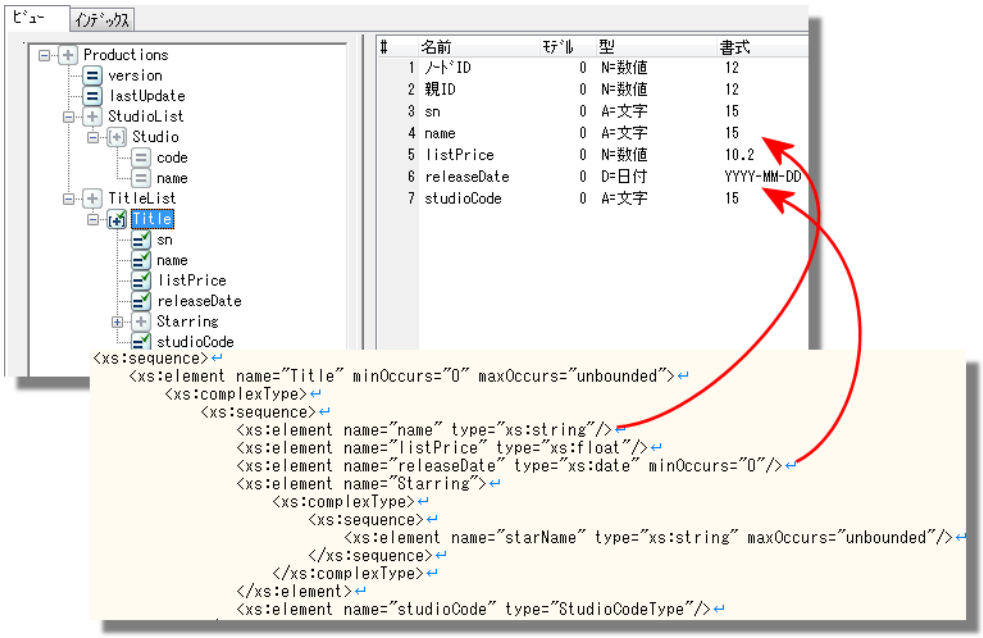
これで、リンク項目がビューに含められます。この例では、タイトルの sn（serial number）項目が子レコードで選択されます。それがリンク項目であることを示すために、名前の前にスラッシュ（/）が追加されます。

これで、レコードがサブタスク内で使用される場合、リンク項目は自動的に更新されます。

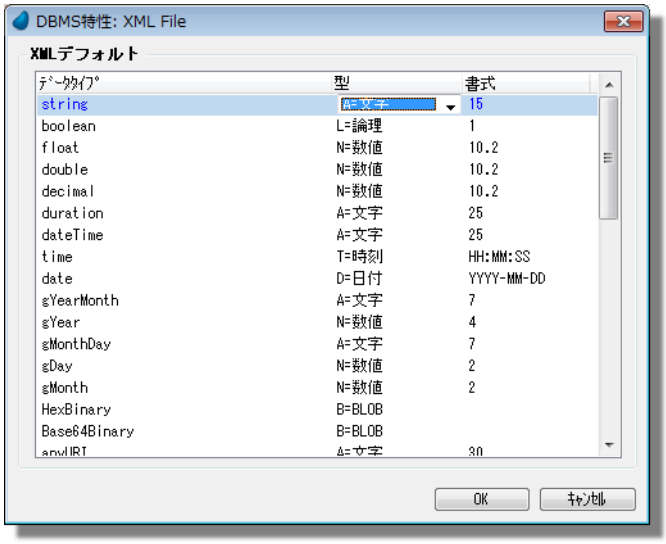
この例では、この映画と関連する俳優のみ表示するように範囲パラメータを使用しています。しかし、/sn 項目には**代入**特性が設定されていません。それにもかかわらず、プログラムを実行した場合、/sn 項目は範囲のために使用された親 sn によって初期設定されます。



XML データ型に対応する Magic xpa のデータ型を決定するには

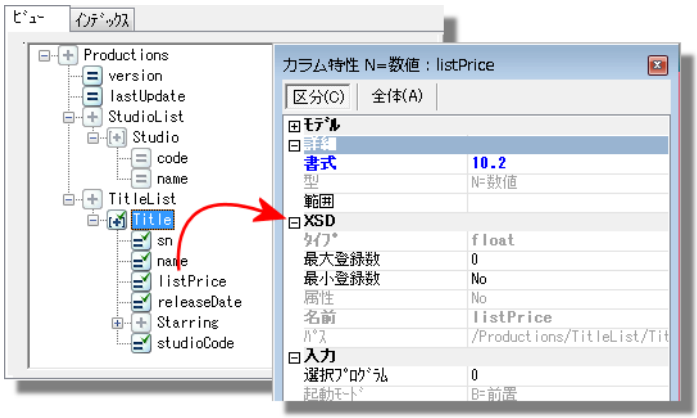


XML ビューを作成する場合、Magic xpa は XML スキーマを参照して、**XML デフォルト特性**（**設定→DBMS →特性**）の定義内容にもとづいてデータが変換されます。この例では、XML データ型 float は、Magic xpa のデータ型の 10.2 に変換されます。



他のツールで作成された XML ドキュメントがある場合、**XML ビュー**の項目定義を変更することで、実際のデータを反映する **XML ビュー**を変更することができます。例えば、実際の XML ドキュメントではより長いタイトル名が必要なため、名前を 15 文字から 40 文字に変更しています。

スキーマの設定を表示する

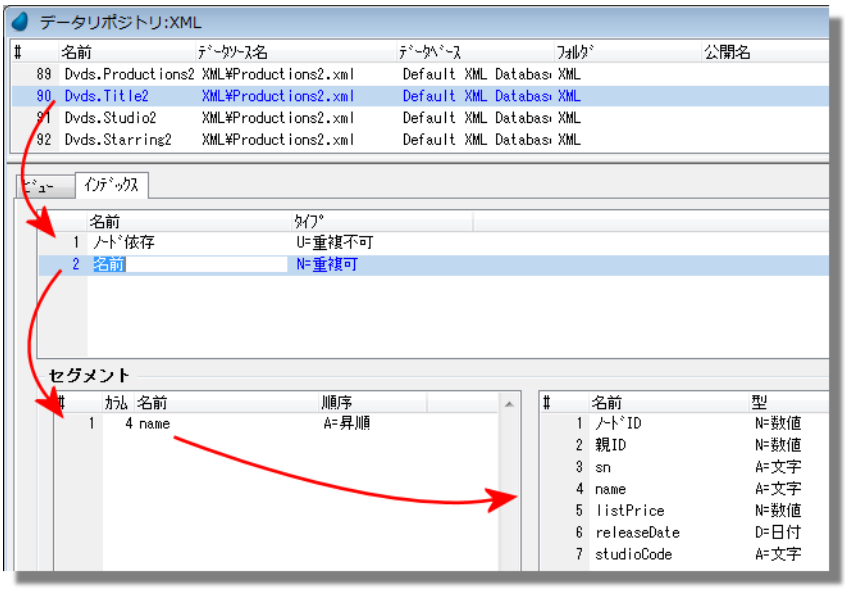


要素が **XML ビュー**に含まれている場合、スキーマ内にどのように定義されているか確認するためにスキーマを直接参照する必要はありません。**特性シート**の **XSD** セクションには、スキーマ設定が表示されます。

任意の順番で XML ドキュメントからデータを取得するには

XML ビューは、データを格納するために一時的なテーブルを作成していることを知っておいてください。一時的なテーブルのため、必要であればインデックスを追加して他のデータソースと同じように使用することができます。

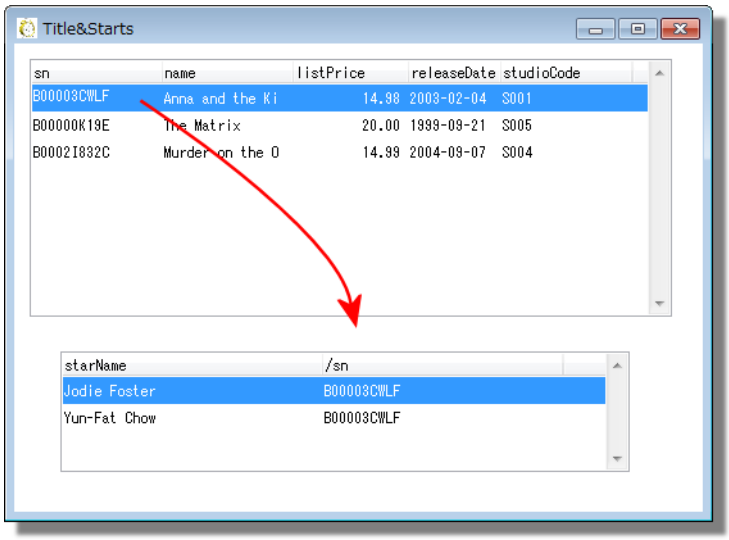
XML ビューのために代替インデックスを作成する



1. **データ**リポジトリで使用する **XML ビュー**を選択します。
2. **インデックス**タブを**クリック**します。
3. ***F4*** を押下して 1 行追加します。
4. インデックスの名前を入力します。右に ***Tab*** 移動します。
5. 重複レコードを許可する場合は、**タイプ**カラムで ***N= 重複可***の選択します。
6. **セグメント**エリア（画面の左下部分）を**クリック**します。***F4*** を押下して 1 行追加します。
7. **ズーム**（***F5***）して、**項目一覧**に移動し、インデックスセグメントに追加する項目を選択します。
8. 必要な項目に対して、ステップ **#6** と **#7** を繰り返して追加します。

これで、プログラムで作成されたインデックスを使用すると、データはそのインデックスにもとづく順番に表示されます。

XML ドキュメント内のマルチ発生要素を処理するには

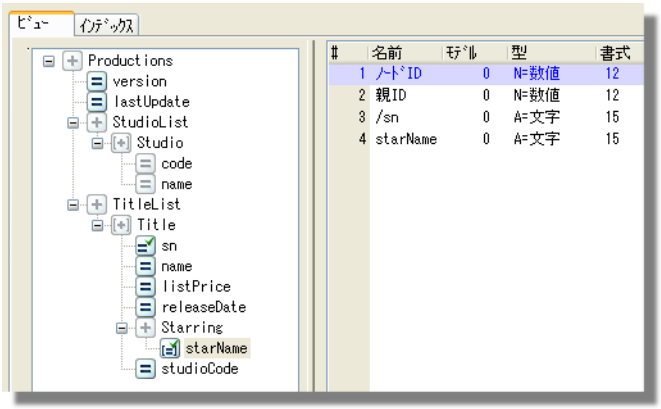


XML ドキュメント内のマルチ発生要素は、ISAM/SQL のデータソースと同じ方法で子レコードとして処理されます。子レコードは子タスク（子タスクのフォームやサブフォーム）で表示されます。範囲が指定されるため、親のレコードと関連するレコードのみ表示されます。

以下は、循環型の XML 要素を使用してプログラムを作成する方法について説明しています。

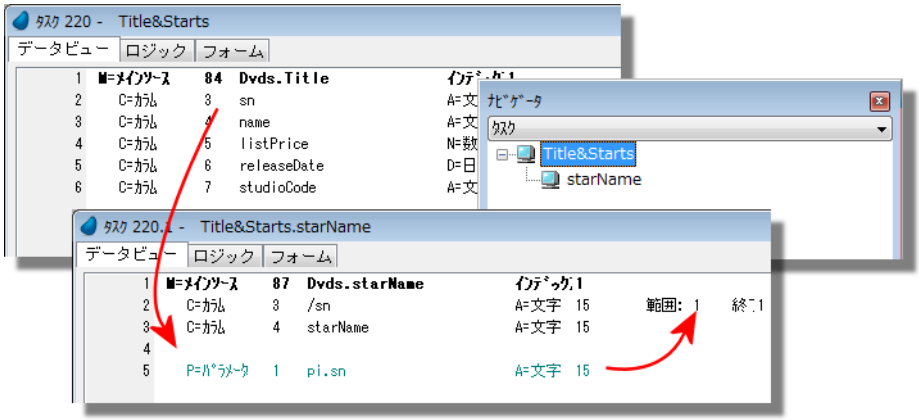
ヒント : *アイコンの表示内容で、スキーマ内の繰り返し要素を確認できます。繰り返し要素は、アイコンが角括弧で囲まれた状態で表示されます。これらは maxOccurs=unbounded または maxOccurs=>1 と定義されているため、スキーマ内で繰り返し要素が確認できます。*

繰り返し要素を表示する



1. 最初に、繰り返し要素のための **XML ビュー**を定義する必要があります。この例では、リンク項目である sn と繰り返し要素 starName が含まれています。

2. XML ビュー上で sn によって要素にアクセスすることができるように、インデックスを作成します。



3. 親要素 Title を表示するタスクを作成します。このタスクで、/sn 項目を定義します。/sn 項目をパラメータとしてサブタスクに渡します。
4. また、starName XML ビューにアクセスするサブタスクを作成します。ここで使用されるインデックスは、/sn 項目で、ビューの範囲指定のためにパラメータ同じ項目を使用します。
5. サブフォームは、サブタスクを表示するために親タスクのフォーム上で使用されます。（第 8 章：「サブフォーム」（185 ページ）を参照してください）。

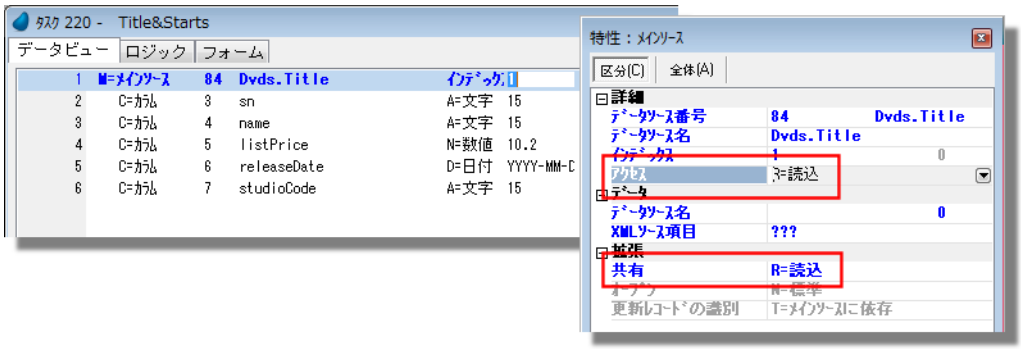
これで、繰り返し要素がサブフォームに表示されます。

同じ XML ドキュメントを異なるユーザで共有させるには

複数のユーザが、XML ドキュメントにアクセスする必要がある場合、読み込み専用モードでドキュメントがオープンされているかを確認する必要があります。

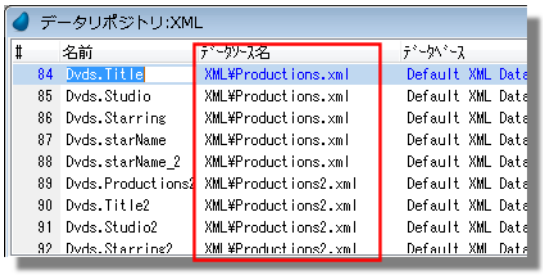
XML ドキュメントは、DBMS で管理されていないため、複数のユーザで同時に更新するはできません。同時に複数のユーザによって更新される可能性のある XML ドキュメントを作成する必要がある場合、SQL/ISAM データソースとしてデータを格納し、必要に応じて保存データから XML ドキュメントを作成するようにしなければなりません。

XML ドキュメントにアクセスモードを設定する

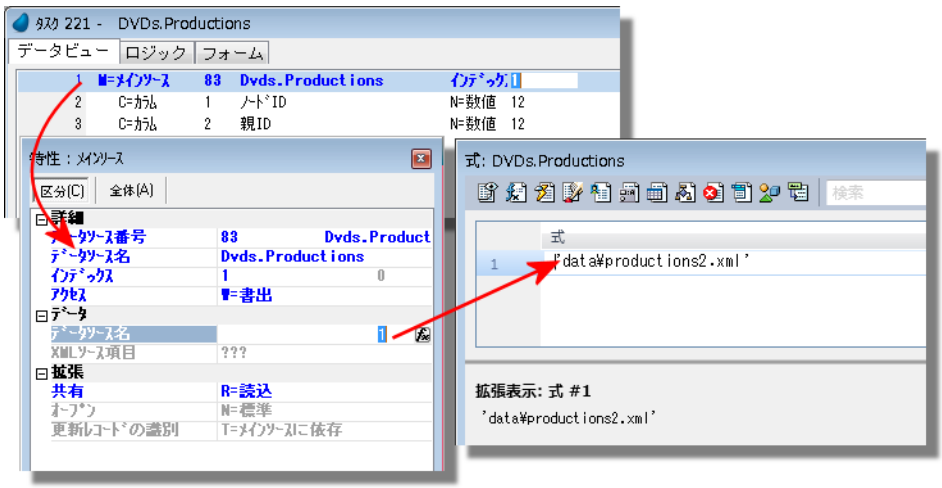


1. データソースの設定を変更したいメインソースまたはリンクのヘッダ行に移動します。
2. *Alt+Enter* を押下して、特性シートを開きます。
3. **アクセス**特性を、***R= 読込***に設定します。このタスクは XML ドキュメントを読み込み専用モードでアクセスします。これで、異なるユーザが同時にこのタスクを使用することができます。
4. **共有**特性を、***R= 読込***か ***W= 書込***に設定します。このタスクが XML ドキュメントを読み込んでいる間、他のタスク（または外部の製品）が、XML ドキュメントを更新できるようにする必要があれば、書込に設定します。***N= なし***には設定しないでください。この設定にした場合、一度に 1 人のユーザだけがデータを参できるようになります。

同じスキーマを使用した異なる XML ドキュメントを作成するには



XML ビューは、**定義取得機能**（*F9*）を使用して設定されたスキーマにもとづいて作成されます。しかし、データは、**データ ソース名**カラムに入力された名前によって定義されたファイルに格納されます。

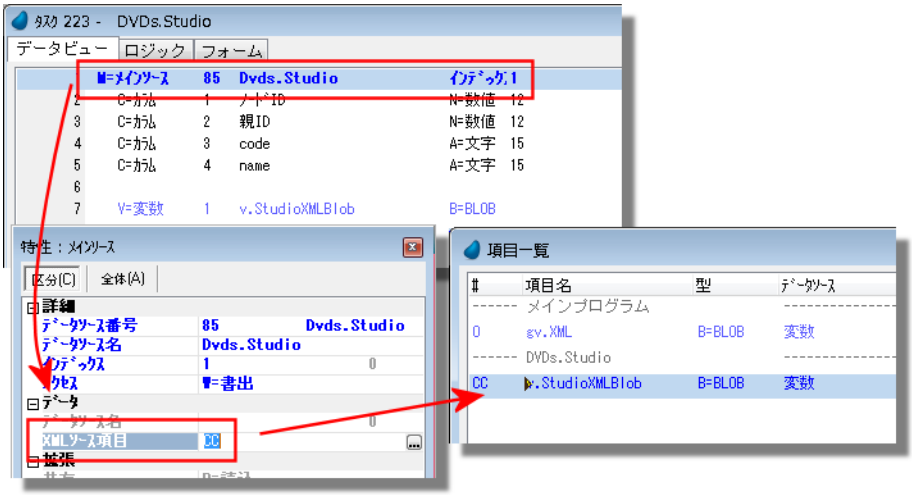


注： ドキュメントにアクセスするプログラムで XML ドキュメント名を上書きすることができます。これは、SQL/ISAM ファイルの場合と同じように**データ ソース名**特性に式を定義することで可能です。この例では 、ファイル名が直接設定されていますが論理名やデータ項目を使用することもできます。

Magic xpa のデータ項目に格納された XML ドキュメントにアクセスするには

BLOB データをファイルに変換せずに、BLOB から直接 XML ドキュメントにアクセスすることで **XML ビュー**を使用することができます。

データソースとして BLOB を使用する



1. メインソースに **XML ビュー**を指定します。
2. **メインソース特性**（*Alt+Enter*）を開きます。
3. **XML ソース項目**特性に移動し、**ズーム**（*F5*）して項目一覧を開きます。
4. XML ドキュメントを含んでいる BLOB を選択します。

これで、ファイルと同じ方法で XML ドキュメントを使用することができます。

参照： 「同じスキーマを使用した異なる XML ドキュメントを作成するには」（323 ページ）

入出力ファイルとして BLOB 項目を使用する

XML データビューではなく XML 関数を使用する場合、**入出力ファイル**テーブルの**式 / 項目**カラムに BLOB 項目を定義しておく必要があります。

1. *Ctrl+I* を押下して**入出力ファイル**テーブルを開きます。
2. *F4* を押下して 1 行追加します。
3. **メディア**カラムで *V= 項目*を選択します。
4. **式 / 項目**カラムで、アクセスしたい BLOB 項目を選択します。

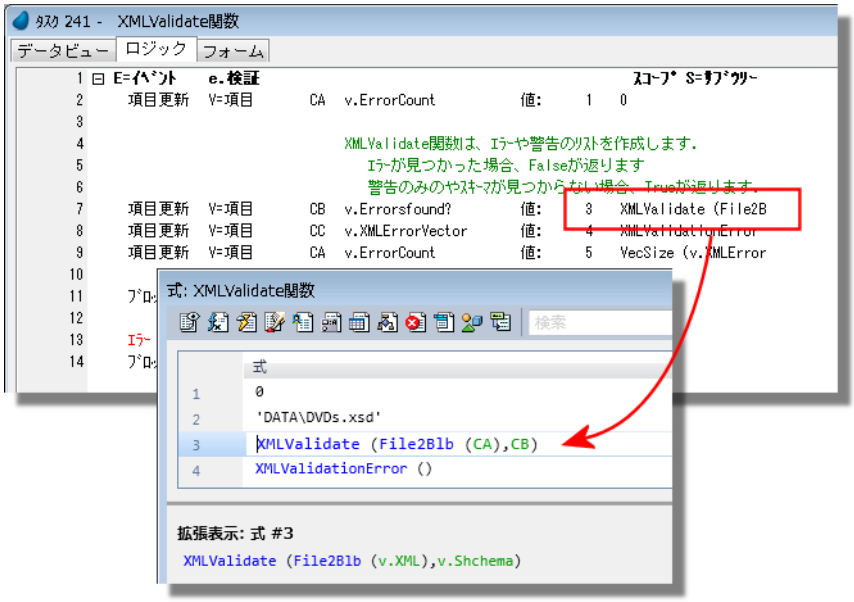
これで、XML 関数を使用して入出力ファイルにアクセスすることができます。タスクの世代番号とファイルの連番を指定することで入出力ファイルを参照できます。

参照： 「スキーマを使用しないで XML ドキュメントを処理するには」（331 ページ）

XML ドキュメントを検証するには

XML ドキュメントを使用する前に、予め検証しておくことを推奨します。XML が正しい書式になっていなかったり、スキーマ定義と合っていない場合、正しい結果が得られません。

XML ドキュメントを検証する場合、**XMLValidate()** 関数を使用します。この関数は、XML ドキュメントとスキーマを比較し、エラーのリストを作成します。



XMLValidate() 関数を使用する

XMLValidate() 関数を使用する基本的な手順は以下の通りです。

1. **XMLValidate()** 関数を実行し、結果を論理データとして格納します。エラーがある場合は、False が返ります。警告だけであったり、スキーマが指定されなかった場合は、True が返り、メッセージも出力されます。
2. **XMLValidationError()** 関数を起動してエラーや警告内容をベクトル項目に格納します。
3. **VecSize()** 関数を使用して、どれだけのエラーや警告がベクトル項目内にあるかを確認します。
4. ベクトルデータを読み込むために、**ブロック While** 処理コマンドを使用してループ処理を実行します。ここで、例にあるような警告ボックスを使用して 1 つずつメッセージを表示したり、テーブルに格納してまとめて参照できるようにします。

「XML ドキュメントの検証エラーを取得するには」（326 ページ）で説明されているような検証エラーを処理するグローバル関数を作成することもできます。

XMLValidate() 関数の構文は以下の通りです。

XMLValidate(*XMLBLOB*, *[XSDSchema]*)

パラメータ：

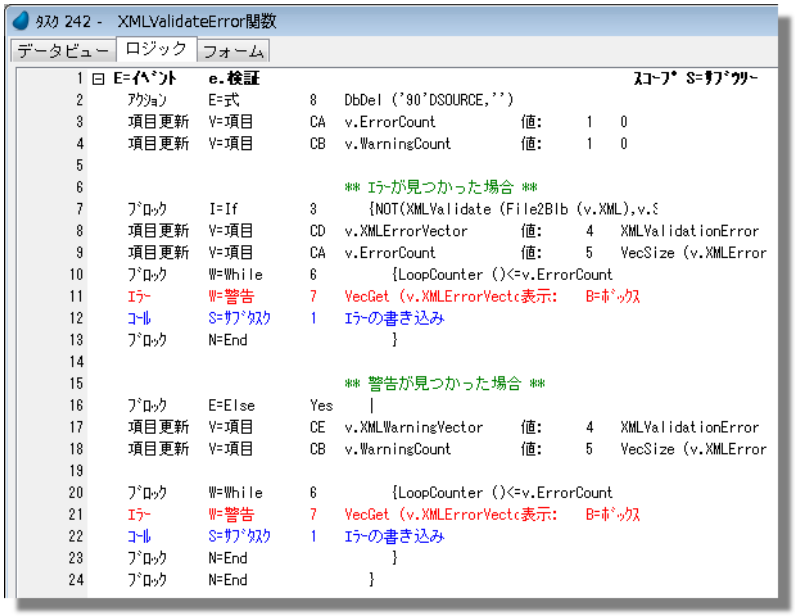
- *XMLBLOB* ：XML ドキュメントが含まれている BLOB 項目
- *XSDSchema* ：スキーマの位置を URL 形式で指定する

戻り値： XML ドキュメントにエラーが見つかった場合は、False が返ります。エラーがないか警告内容のみの場合は、True が返ります。

参照： 『リファレンスヘルプ』の XMLValidate 関数のトピック
「XML ドキュメントの検証エラーを取得するには」（326 ページ）

XML ドキュメントの検証エラーを取得するには

XMLValidate() 関数を実行した後、エラーがあれば **XMLValidationError()** 関数を使用してエラー情報を取得することができます。この関数はテキストメッセージ（Unicode）のベクトルデータが返ります。**VecSize()** 関数を使用することでエラーメッセージの数を取得することができ、テーブルへの格納や、印刷や表示を行うことができます。



参照： 「XML にアクセスしている間に発生したエラーを処理するには」（327 ページ）

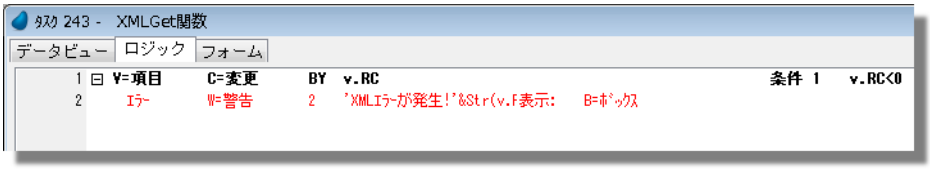
XML にアクセスしている間に発生したエラーを処理するには

XML 関数のいくつかは、エラーが発生したことを確認するための戻り値を返します。例えば、**XMLSetEncoding()** 関数は適切に動作した場合は、0 が返り。そうでなければ負数が返ります。

注： 戻り値は負数になる場合があるため、格納する項目は、負値に対応した書式（例えば N や 5N などを含む書式）にする必要があります。

グローバルなエラーハンドラを作成する

すべてのエラーコードは負の整数です。従って、ここで示されているように、戻り値が負数の場合に実行されるハンドラを作成することができます。



メインプログラム内で戻り値を使用することで、すべての XML 関数のエラーを処理することができます。同じ戻り値がすべての XML 関数で使用されるため、ユーザフレンドリなエラーメッセージを作成することもできます。エラーコードは、『リファレンスヘルプ』に記述されています。

XML ドキュメント対するエンコードを指定するには

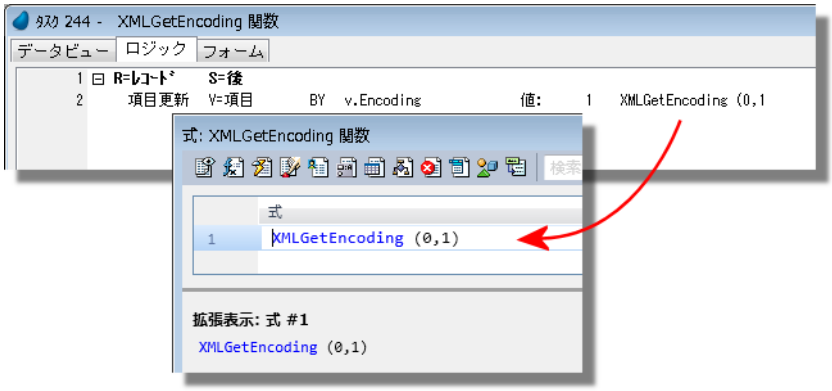
以下に示されるように、XML ドキュメントのエンコーディングは XML ドキュメントのヘッダ（の最初の行）に指定します。

```
<xml version="1.0" encoding="Shift-Jis"?>
```

エンコーディングが指定されない場合、utf-8 として処理されます。エンコーディング指定は、XML を構文解析する際にパーザによって使用されます。文字によっては、あるエンコーディングでは有効であっても、別の指定では無効になる場合があります。

Magic xpa で XML を処理する場合、エンコーディングの指定内容を意識する必要はありませんが、**XMLGetEncoding()** 関数を使用することで取得することができます。

XMLGetEncoding() 関数を使用する



XMLGetEncoding() 関数の構文は以下の通りです。

XMLGetEncoding(*generation*, *I/O entry*)

パラメータ：

- *generation*：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- *I/O entry*：XML ドキュメントが割り当てられている入出力ファイルの番号です。

戻り値は、XML ドキュメントに設定されている encoding 指定の値が文字列で返ります。

参照： 『リファレンスヘルプ』の **XMLGetEncoding()** および **XMLSetEncoding()** 関数のトピックを参照してください。

Base64 でエンコードされた XML データを処理するには

XML ビューを使用している場合、項目が Base64Binary としてスキーマで定義されていれば、Magic xpa はテキスト⇔バイナリ値の自動変換を行います。

例えば、ここに、2 つの Base64 の要素が追加された Sample.xsd があります。

```
<xs:element name="Studio" minOccurs="0" maxOccurs="unbounded">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="RTF" type="xs:base64Binary"/>
      /xs:sequence>
      <xs:attribute name="code" type="StudioCodeType" use="required"/>
```

このスキーマを使用して **XML ビュー**を作成すると、以下のように表示されます。

この例では、RTF テキストを格納するために BLOB が使用されていますが、イメージやその他のバイナリデータを含めることもできます。

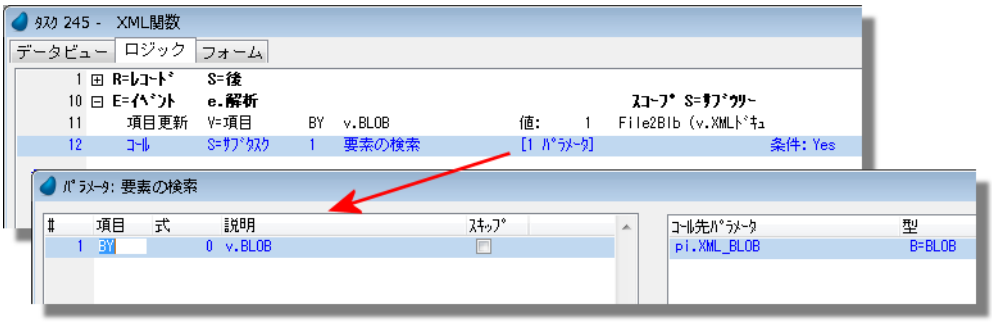
RTF 項目にデータを入力し、XML ファイルに格納された場合、以下のように保存されます。

```
<Studio code="S001">
    <name>Universal Studios</name>
    <RTF>c2QgIGFzZGYgYWRmIGQKDWFkZiBhc2RmCg1hZHNmYXNkZmFkc2YKDWFzZCBmNZg==</RTF>
</Studio>
```

Magic xpa は、上記の処理を全て行うことできます。

注：　XML ビューに対して 1 つの BLOB のみ定義できます。

XML ドキュメントをパラメータとして渡すには

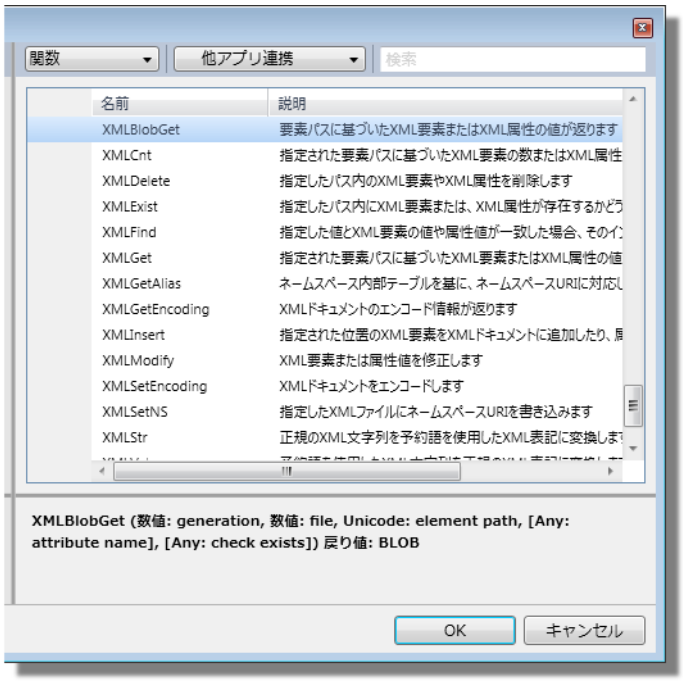


パラメータとして XML ドキュメントを渡す場合、最も簡単な方法は、ドキュメントを BLOB 項目に格納した上で、直接アクセスすることです。「Magic xpa のデータ項目に格納された XML ドキュメントにアクセスするには」（324 ページ）を参照してください。

ファイルを BLOB に変換するには、**File2BLB()** 関数を使用します。使用方法は、『リファレンスヘルプ』を参照してください。

スキーマを使用しないで XML ドキュメントを処理するには

データリポジトリで **XML ビュー**を使用するには、スキーマを使用しなければなりません。スキーマのない XML ドキュメントがある場合、サードパーティの製品を使用するか、手動で作成する必要があります。



これ以外には、Magic xpa の組み込み機能を利用してアクセスすることもできます。これらの機能は、スキーマを使用しない代わりに XML ファイルの内部フォーマットに関する情報を取得しなければなりません。これらの機能の使用方法は、以下で説明しています。

「XML ドキュメント内のデータの取得、更新、挿入を行うには」（332 ページ）

「XML ドキュメント内のデータ要素と階層構造を識別するには」（334 ページ）

XML ドキュメント内のデータの取得、更新、挿入を行うには

XML ファイルにアクセスするために **XML ビュー**を定義することで、他のデータソースと同じように XML データソースをタスクで使用することができます。**XML ビュー**の使用については、「最初から XML ドキュメントを作成するには」（309 ページ）を参照してください。

しかし、XML ファイルが定義された入出力ファイルを処理するために XML 関数を使用することもできます。これは **XML ビュー**を使用する場合より作業量が増えることになります。これらの関数は、主に Magic V9Plus との互換性を維持するために存在しています。

必要条件： データの更新や挿入を行う場合、ファイルのオープンモードを ***書込***（***追加***ではない）にする必要があります。また、データの内容が属性かどうかによって構文が多少異なります。

フォーマット要素パスと属性名の詳細については、「XML ドキュメント内のデータ要素と階層構造を識別するには」（334 ページ）を参照してください。 これらはかなり複雑な関数です。『リファレンスヘルプ』の関数のトピックを参照してください。

XML データを取得する

XMLGet() 関数は、XML ファイル内の要素を取得します。構文は以下のとおりです。

XMLGet(*generation*, *file*, *element path*, *attribute name***)**

パラメータ：

- *generation*：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- *file*：XML ドキュメントが割り当てられている入出力ファイルの番号です。
- *element path*：要素パスを表す文字列です。
- *attribute name*：要素内の属性を表す文字列です。

戻り値：関数の処理が成功した場合、要素または属性を表す文字列が返ります。処理が失敗した場合は、空白が返ります。

XML データを更新する

XMLModify() 関数は、XML ファイル内の要素を更新します。構文は以下のとおりです。

XMLModify(*generation*, *I/O entry*, *element path*, *attribute*, *value [*, *auto convert]***)**

パラメータ：

- *generation*：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- *I/O entry*：XML ドキュメントが割り当てられている入出力ファイルの番号です。
- *element path*：要素パスを表す文字列です。
- *attribute*：要素内の属性を表す文字列です。
- *value*：修正される要素／属性の値を含む文字列です。
- *auto convert*：（オプション）True が設定された場合、正規の XML フォーマットに変換します。

戻り値：関数の処理が成功した場合、0 が返ります。処理が失敗した場合は、負数のエラーコードが返ります。

XML データを挿入する

XMLInsert() 関数は、XML ファイル内に要素を追加します。構文は以下のとおりです。

XMLInsert (*generation*, *I/O entry*, *element path*, *attribute*, *value [*, *before/after flag*, *reference element*, *auto convert]***)**

パラメータ：

- *generation*：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- *I/O entry*：XML ドキュメントが割り当てられている入出力ファイルの番号です。
- *element path*：要素パスを表す文字列です。
- *attribute*：要素内の属性を表す文字列です。
- *value*：挿入される要素／属性の値を含む文字型、BLOB 型（RTF）のデータです。
- *before/after flag*：（オプション）追加する場所（*reference element* の前 / 後）を指定します。
- *reference element*：（オプション）追加する際の参照用に指定する要素名
- *auto convert*：（オプション）True が設定された場合、正規の XML フォーマットに変換します。

戻り値：関数の処理が成功した場合、0 が返ります。処理が失敗した場合は、負数のエラーコードが返ります。

XML データを削除する

XMLDelete() 関数は、XML ファイル内の要素を削除します。構文は以下のとおりです。

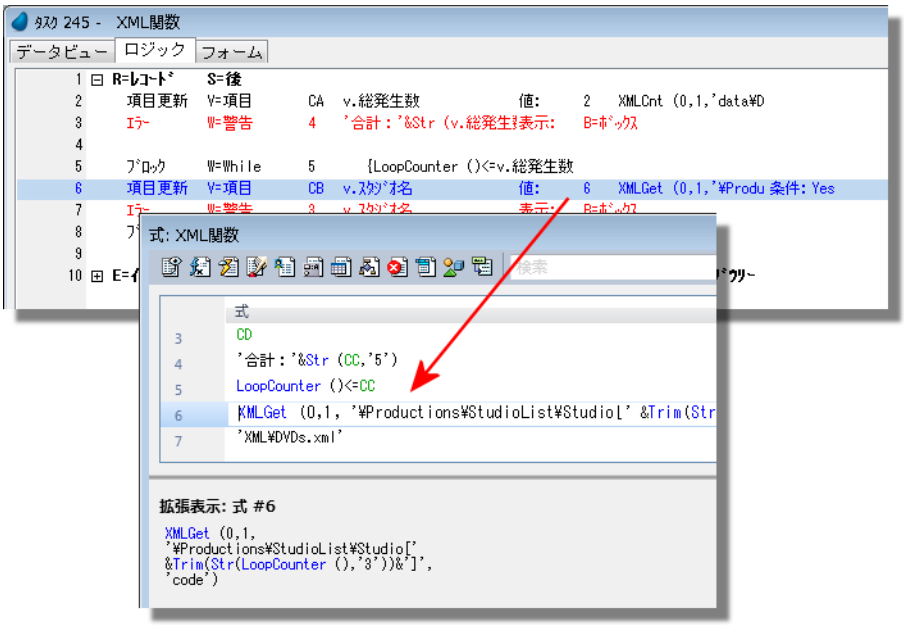
XMLInsert (*generation*, *I/O entry*, *element path*, *attribute***)**

パラメータ：

- *generation*：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- *I/O entry*：XML ドキュメントが割り当てられている入出力ファイルの番号です。
- *element path*：要素パスを表す文字列です。
- *attribute*：要素内の属性を表す文字列です。

戻り値：関数の処理が成功した場合、0 が返ります。処理が失敗した場合は、負数のエラーコードが返ります。

XML ドキュメント内のデータ要素と階層構造を識別するには



XML データソースにアクセスするために、**XML ビュー**を使用している場合、他のデータソースのカラムと同じように要素にアクセスすることができます。階層構造は、**データ**リポジトリで簡単に参照することができます。「最初から XML ドキュメントを作成するには」（309 ページ）を参照してください。

XML 関数を使用する場合は、XML ファイル内の要素パスを指定する必要があります。

これらの関数の詳細情報は、「XML ドキュメント内のデータの取得、更新、挿入を行うには」（332 ページ）を参照してください。例として **XMLGet()** 関数を使用してみます

XMLGet(*generation*, *file*, *element path*, *attribute name*)

パラメータ：

- ***generation***：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- ***file***：XML ドキュメントが割り当てられている入出力ファイルの番号です。
- ***element path***：要素パスを表す文字列です。
- ***attribute name***：要素内の属性を表す文字列です。

パス指定された各要素は、前のスラッシュによって分離されたその名前とインデックス（複数存在する場合）によって識別されます。最後のパラメータは（もしあれば）属性のために使用されます。

例として以下の XML から取得してみます。

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Productions lastUpdate="2006-01-01" version="1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
 <StudioList>
  <Studio code="S004">
   <name>Paramount</name>
  </Studio>
  <Studio code="S005">
   <name>Warner Home Video</name>
  </Studio>
  <Studio code="S006 ">
   <name>New Line Home Entertainment</name>
```

例えば、要素名 Studio code の 3 番目の値を取得する場合以下のような式を指定します。

```
XMLGet (0,1, '¥Productions¥StudioList¥Studio[3]','code')
```

この場合、S006 が返ります。

要素を取得する場合は、以下の式を指定します。

```
XMLGet (0,1, '¥Productions¥StudioList¥Studio[3]/name', '')
```

この場合、New Line Home Entertainment が返ります。

通常は、インデックスはハードコーディングされず、項目や **LoopCounter()** 関数などを使用することになります。

参照： 「XML ドキュメント内のデータの取得、更新、挿入を行うには」（332 ページ）

XML ドキュメント内の混在内容を扱うには

混在内容の要素には、要素とテキストの両方が含まれています。例えば以下のような XML になります。

```
<?xml version="1.0" encoding="UTF-8"?>
<notice>
    To
    <name>Fred Flicker</name>
    Your DVDs are are overdue. They were due on
    <duedate>07/21/2003</duedate>
    Please bring them in ASAP.
    We really appreciate it. Thanks!
    <greeting>Sincerely,</greeting>
    <outlet>AAA Rentals</outlet>
</notice>
```

この例では、4 つのテキスト項目（通常表示）と 4 つの要素項目（**ボールド表示**）が定義されています。

最初に、複合ルートを持つ XML ドキュメントを受け取った場合。例えば、**XML ビュー**を使用する場合に、混在内容を処理するには、以下の 2 つの関数を使用します。

- **DbXmlMixedGet()**
- **DbXmlMixedSet()**

［このページは意図的に空白にしています。］

第 14 章 : COM

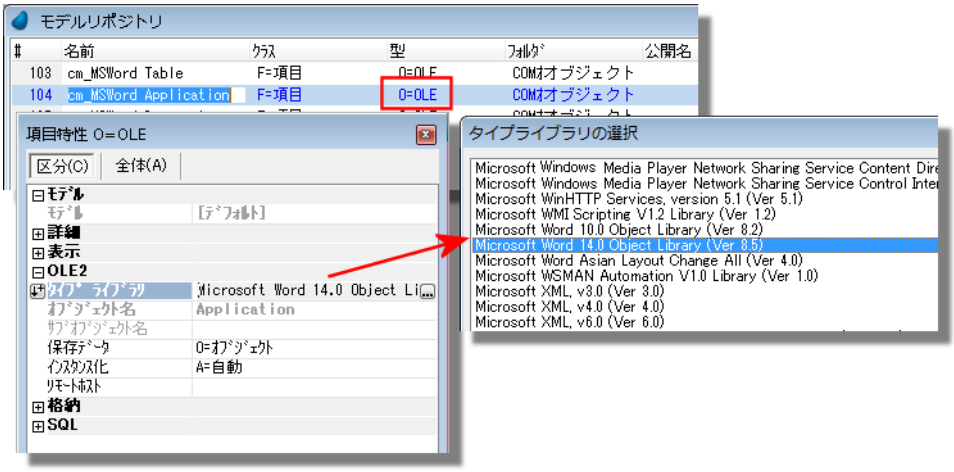
使用する COM オブジェクトを定義するには

COM オブジェクトを使用するには、あらかじめオブジェクトを定義しておく必要があります。COM オブジェクトは、他の項目と同じようにタスクのデータビューエディタで定義することができます。

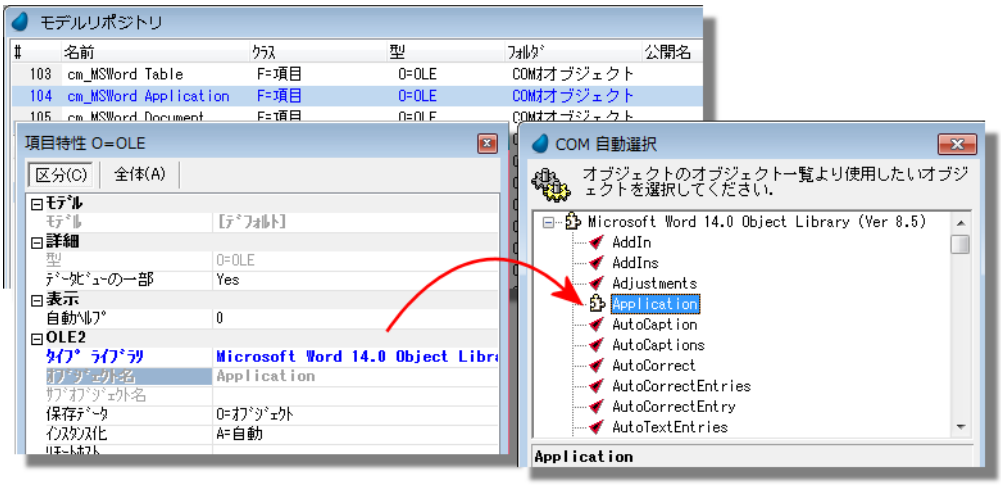
COM オブジェクトの定義は、日付型や文字型の項目よりは多少設定する作業が多くなります。ライブラリをリストから選択する必要があり、また各ライブラリからオブジェクトを選択する必要があります。COM オブジェクトは、Word や Excel などの他の製品とリンクするために使用することが多いため、それらの製品がバージョンアップされた場合 COM 定義を修正する必要が発生する可能性があります。

このような理由のために、モデルに COM を定義し、必要に応じてそれらを再利用することを推奨します。詳細は、「COM オブジェクト定義を再利用するには」（354 ページ）を参照してください。

COM オブジェクトを定義する



1. **モデル**リポジトリで一行追加（*F4*、または**編集→行作成**）します。
2. **名前**カラムに任意の名前を入力してください。
3. **クラス**カラムはデフォルト（***F= 項目***）のままにします。
4. **型**カラムは ***O=OLE***（フォームに表示させる場合は、***X=ActiveX***）に設定します。
5. **タイプライブラリ**特性から**ズーム**して、使用したいライブラリを選択します。PC にインストールされているすべてのライブラリが表示されるため、非常に大きなリストになる場合があります。選択したいセクション名の先頭の文字を入力することで位置付けを行うことができます。***Enter*** を押下して使用したいライブラリを選択します。
6. **オブジェク名**特性から**ズーム**して、このライブラリから使用したいオブジェクトを選択します。

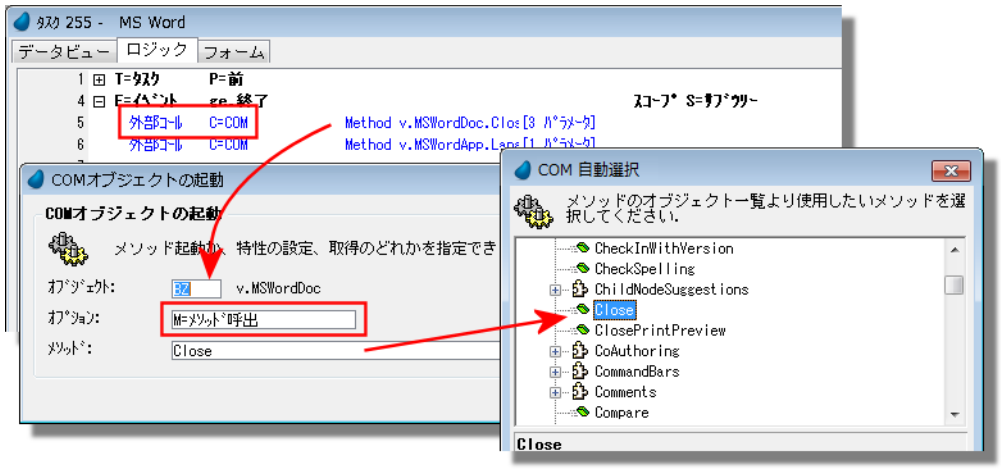


7. 必要であれば、同じ方法で**サブオブジェクト名**を選択します。

これで、COM オブジェクトを宣言するためにこのモデルを使用することができます。この場合、項目を定義する際にこのモデルを設定するだけで済みます。

注： COM ライブラリのリストを見て、何を選択していいのかが分からなくなる場合があります。COM ライブラリは、基本的には膨大な関数リストのようなものです。ライブラリに関するドキュメント（例：Word の開発者用ドキュメント）を参照しない限りどの関数が必要なのかは、実際分かりません。しかし Magic xpa は、オブジェクトを簡単に試してみることで、オブジェクトの内容を確認することができます。

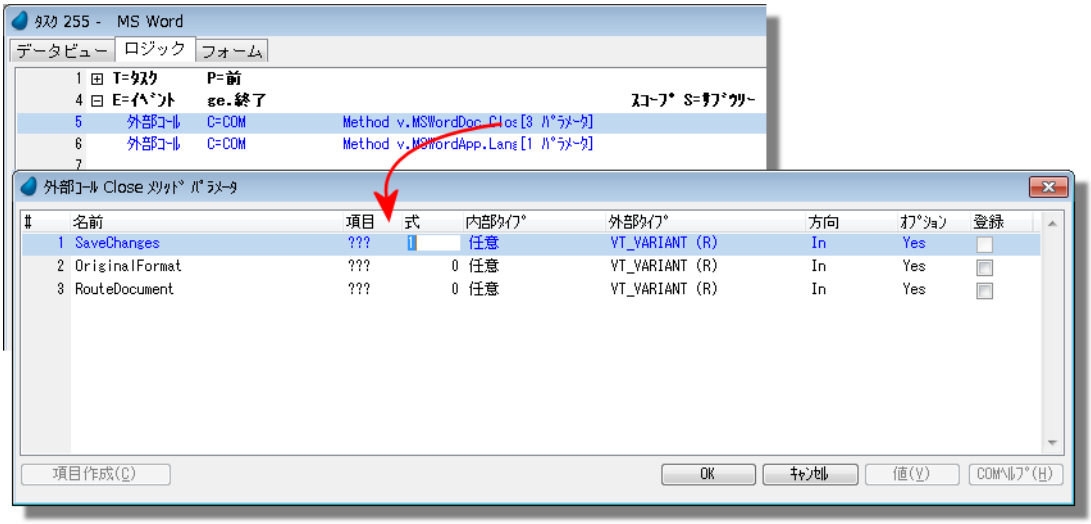
COM オブジェクトのメソッドを呼び出すには



COM オブジェクトが宣言されると、プログラムやサブタスクを起動するように、オブジェクトのメソッドを呼び出すことができます。ただし、**外部コール**という異なる処理コマンドが使用されます。この処理コマンドは Magic xpa 以外のプログラムを呼び出す場合に使用されます。

COM メソッドを呼び出す

1. **ロジックユニット**内で 1 行追加します。
2. ***I*** を入力するかドロップダウンリストから選択して、**外部コール**処理コマンドを設定します。
3. ***C*** を入力するかドロップダウンリストから選択して、***COM*** を起動タイプとして設定します。***Tab*** 移動します。
4. **ズーム**して **COM オブジェクト**ダイアログを表示します。
5. **オブジェクト**から**ズーム**して COM オブジェクトを選択します。
6. **オプション**でドロップダウンリストから ***M= メソッドの呼び出し***を選択します。
7. **メソッド**から**ズーム**して、起動するメソッドを選択します。
8. ***Esc*** を押下して **COM オブジェクト**ダイアログを閉じます。**パラメータ**カラムに ***Tab*** 移動し、**ズーム**します。

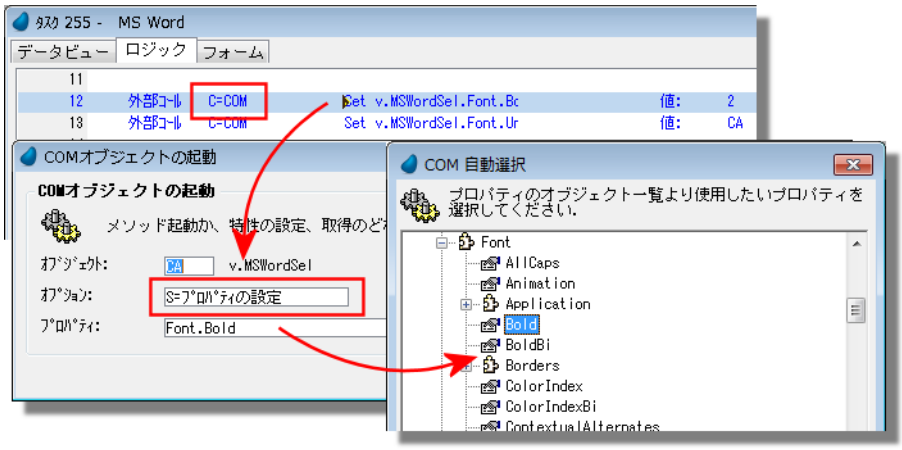


9. このメソッドに渡す**パラメーター一覧**が表示されます。一覧上に表示されるパラメータの情報（入力用か出力用か、オプションかどうか、データタイプは何なのか）を確認してください。
   登録カラム内のボックスを**クリック**すると、パラメータのデータ型に対応した項目が作成されます。Magic xpa によってデータ変換が行われるため、他のプログラミングツールを使用する場合ほどはデータ型について正確に知る必要はありません。例えば、この例では VT_VARIANT タイプ（ポインタ内に渡す必要があることを意味しています）を使用します。値を式で指定（この場合、項目カラムは 0 になります）するだけで、あとは Magic xpa が処理します。数値を渡す場合も同様に、その数値が float 型か long 型か、あるいは packed integer かどうかということを意識する必要はありません。

これで、COM オブジェクト内のメソッドを呼び出す処理が作成されました。

ヒント :工数を減らすために、コール COM 処理コマンドをコピーすることができます。同じメソッドにを繰り返し呼び出すような場合（例えば、COM オブジェクトを使用して Word 文書をフォーマットしていたり、段落を追加したりする必要がある場合）、このような操作は便利です。既存の処理コマンドをコピーするには、Ctrl+Shift+R か Ctrl+C を使用し、貼り付ける場合は Ctrl+V を使用します。

COM オブジェクトのプロパティを設定 / 取得するには



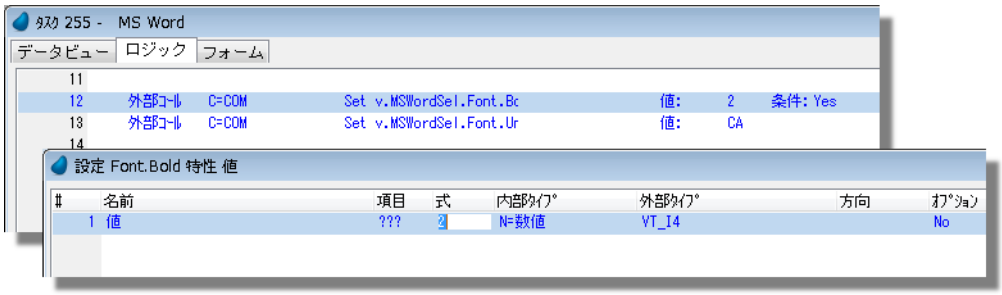
COM オブジェクトが宣言されたら、***プロパティの設定***オプションを使用して特性値を変更したり、***プロパティの取得***オプションを使用して特性値を取得することができます。

プロパティの設定オプションは、オブジェクトを変更するために使用されます。例えば、ActiveX オブジェクトに対して、オブジェクトの色やどのように表示されるかを変更するためにために使用できます。この例（Word 文書の作成）では、現在のフォントスタイルをボールドに変更するために***プロパティの設定***オプションが使用され、これらの特性値を取得するために***プロパティの取得***オプションが使用されます。

プロパティの設定 / 取得オプションは、オブジェクトの値を変更したり取り出す場合にも使用されます。例えば、カレンダーオブジェクト内で、デフォルト日付を設定するために設定オプションが使用され、ユーザが選択した日付を取り出すために取得オプションが使用されます。

COM オブジェクトでプロパティの設定 / 取得オプションを使用する

1. ロジックユニット内で 1 行追加します。
2. *I* を入力するかドロップダウンリストから選択して、外部コール処理コマンドを設定します。
3. *C* を入力するかドロップダウンリストから選択して、***COM*** を起動タイプとして設定します。*Tab* 移動します。
4. ズームして **COM オブジェクト**ダイアログを表示します。
5. **オブジェクト**からズームして COM オブジェクトを選択します。
6. **オプション**でドロップダウンリストから ***S= プロパティの設定***（***G= プロパティの取得***）を選択します。
7. **プロパティ**からズームして、アクセスするプロパティを選択します。
8. *Esc* を押下して **COM オブジェクト**ダイアログを閉じます。**パラメータ**カラムに *Tab* 移動し、ズームします。



9. このプロパティに設定 / 取得するパラメーター一覧が表示されます。一覧上に表示されるパラメータの情報（オプションかどうか、データタイプは何なのか）を確認してください。
   登録カラム内のボックスをクリックすると、パラメータのデータ型に対応した項目が作成されます。Magic xpa によってデータ変換が行なわれるため、他のプログラミングツールを使用する場合ほどはデータ型について正確に知る必要はありません。例えば、この例では VT_I4 タイプ（signed long integer）を使用します。値を式で指定（この場合、項目カラムは 0 になります）するだけで、あとは Magic xpa が処理します。

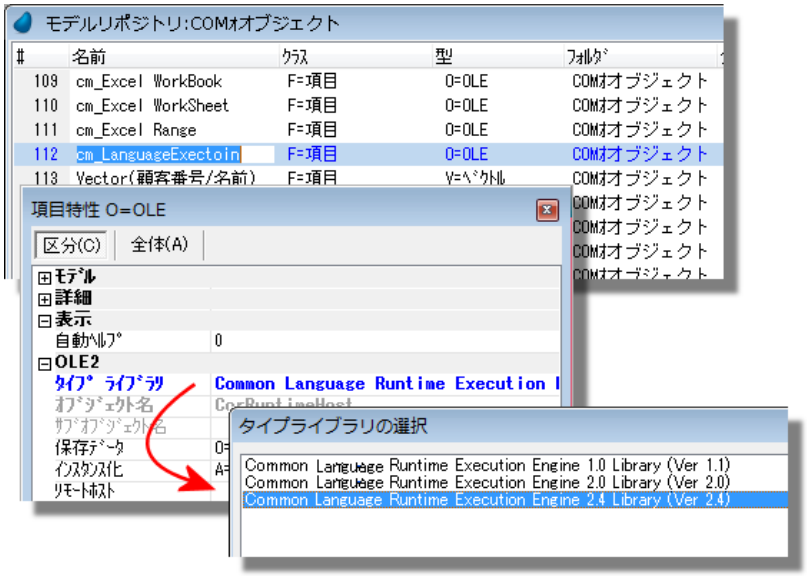
これで、COM オブジェクトのプロパティの設定 / 取得処理が作成されました。

ヒント :工数を減らすために、コール COM 処理コマンドをコピーすることができます。同じメソッドにを繰り返し呼び出すような場合（例えば、COM オブジェクトを使用して Word 文書をフォーマットしていたり、段落を追加したりする必要がある場合）、このような操作は便利です。既存の処理コマンドをコピーするには、Ctrl+Shift+R か Ctrl+C を使用し、貼り付ける場合は Ctrl+V を使用します。

COM オブジェクトの参照先を変更するには

COM 規格の非常に良いところは、それらが、前方互換で設計されていることです。すなわち、COM オブジェクトが規格に準拠している場合、COM オブジェクトの新バージョンをインストールしても、今まで定義していたメソッドがそのまま使用できるということです。規格はこれに関して様々な詳細をカバーし、独自の COM にオブジェクトを作成するときも、この規格に準拠する必要があります。

しかし、ライブラリ名は変更されます。例えば、毎年 PC 上の Microsoft Word をアップグレードした場合、Microsoft の COM ライブラリの全てのバージョンが登録されることになります。COM オブジェクトが古いバージョンを選択されている場合、新バージョンをインストールしてもこの選択内容は自動的に更新されません。この場合、古い COM オブジェクトがインストールされていない別の PC に新しいオブジェクトをインストールしても、オブジェクトへの呼び出しは失敗します。



例えば、ここに、Ver1.1 の COM オブジェクトがあります。Ver.2.4 もこの PC に存在しているとします。**タイプライブラリ**特性から**ズーム**すると、これらの 2 つのライブラリが同じオブジェクトと認識され、新しいライブラリが選択できる状態になっています。

このプロジェクトを古いライブラリが含まれていない PC 上で実行した場合、古いライブラリ名が示され、**ズーム**して新しいライブラリを選択できるようになります。

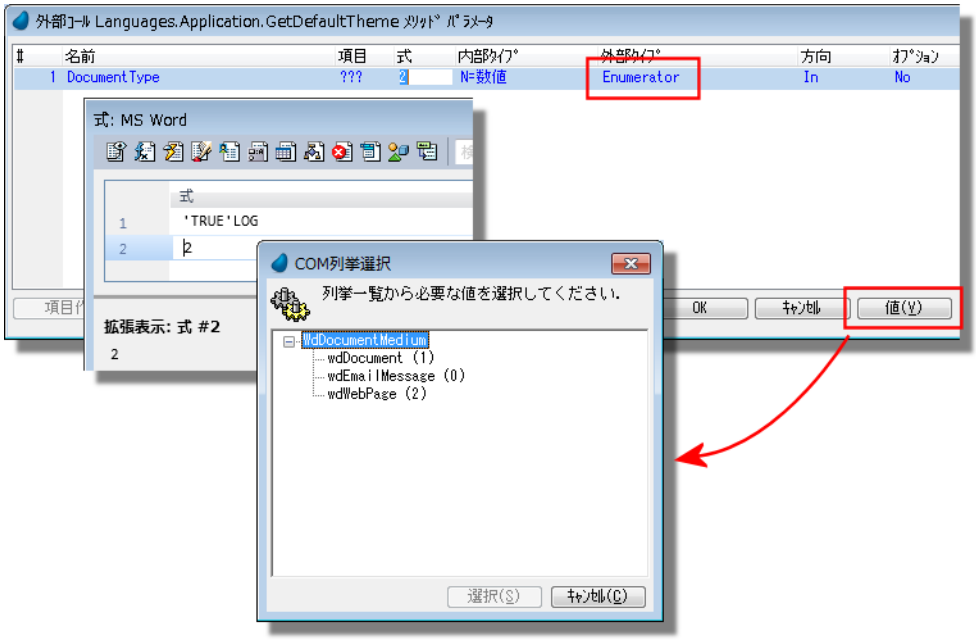
Magic xpa では、オブジェクト名は変更できず、このオブジェクト内の（同じライブラリの別バージョン以外の）別のライブラリにも変更できません。これによって、不用意にオブジェクトが変更されることで、参照処理が失敗するようことを防止しています。

COM ライブラリの参照先を変更する

1. タスクの**データビューエディタ**や**モデル**リポジトリ内の COM オブジェクトが定義されている項目に移動します。
2. **特性**シート（*Alt+F1*）を開き、**タイプライブラリ**特性に移動します。
3. **ズーム**（*F5*、または**ダブルクリック**）して**タイプライブラリ選択**ボックスを開きます。このオブジェクトに対する複数のレビジョンが存在するかどうかによって、1 つだけ表示されたり複数表示されたりします。
4. *Enter* を押下して、アップグレードされたオブジェクトを選択します。

これで、このオブジェクトの呼び出し処理は、アップグレードされたライブラリを使用することになります。

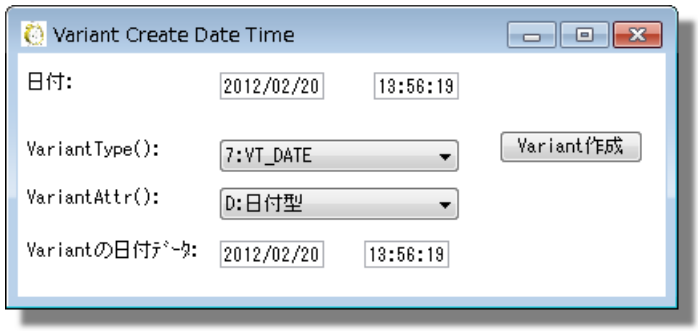
列挙型パラメータ値を設定するには



列挙型は、定数の固定されたセットから構成されるデータタイプです。例えば、週における日数や 1 年における月、または段落整列の形式のようなものです。しばしば、COM オブジェクト内で、複数のオブジェクト間で選択肢を持っている場合、選択をするために、整数を渡します。

どの整数がどの選択肢を表すかは、COM オブジェクトに対してドキュメントを調べるために必要な何かであり、実際に試してみることで何が起こるかを確認することができます。

Variant 型に対してデータの設定 / 取得を行うには



多くの COM オブジェクトは、VT_VARIANT データ型を使用しています。VT_VARIANT を定義することで、本質的には、どのようなデータ型でも保持できます。しかし、使用しているオブジェクトは特定の機能のために一定のフォーマットのデータを期待しているはずです。

通常、Magic xpa は自動的に変換処理を実行します。例えば、特定の VT_VARIANT パラメータが配列を期待している予期していて、ベクトルを設定した場合、ただベクトルを渡し、Magic xpa はそれを処理します（設定例は、「COM オブジェクトから配列値を取り出したり設定したりするには」（349 ページ）を参照してください）。

同様に、オブジェクトが VT_VARIANT で負の整数を返し、負数を有効にした数値型項目で受け取った場合、Magic xpa は正しく変換処理を行います。

しかし、管理水準をより高めたい場合は、受け渡しするデータとして BLOB 型を使用し、Variant 関数を使用することで、送信する Variant タイプを正確に設定することができます。

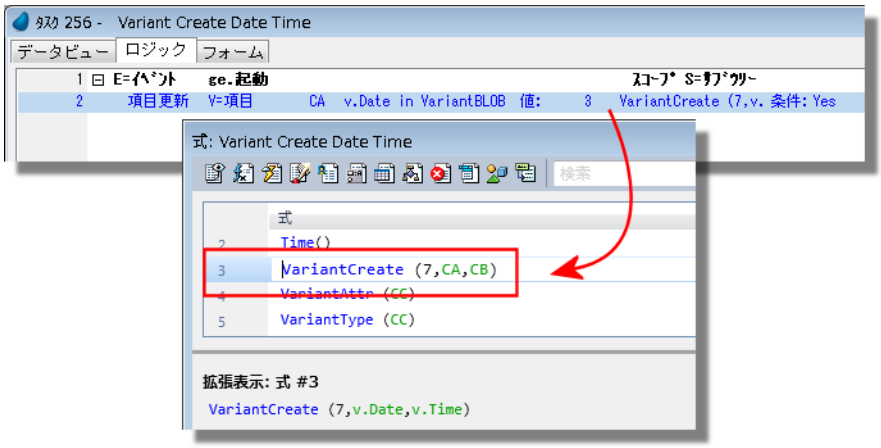
オブジェクトから返される Variant からデータを取得するために同じ関数を使用することもできます。

以下に挙げるのが、使用できる関数です。

- **VariantCreate()**：BLOB Variant にデータをコピーします。「Variant を作成する」（344 ページ）を参照してください。
- **VariantGet()**：BLOB Variant から項目にデータをコピーします。「Variant からデータを取得する」（345 ページ）を参照してください。
- **VariantGetVector()**：BLOB Variant からベクトル項目にデータをコピーします。『リファレンスヘルプ』を参照してください。
- **VariantAttr()**：Variant の Magic データ型を取得します。「VariantAttr() 関数を使用してデータ型を取得する」（347 ページ）を参照してください。
- **VariantType()**：Variant のデータ型（を表す数値）を取得します。「VariantType() 関数を使用してデータタイプを取得する」（348 ページ）を参照してください。

Variant を作成する

VariantCreate() 関数を使用することで Variant データを実装させることができます。この例では、日付と時間を VT_DATE 型の Variant に移動するために **VariantCreate()** 関数を使用しています。これは実際に日付 / 時刻を保持することができます。



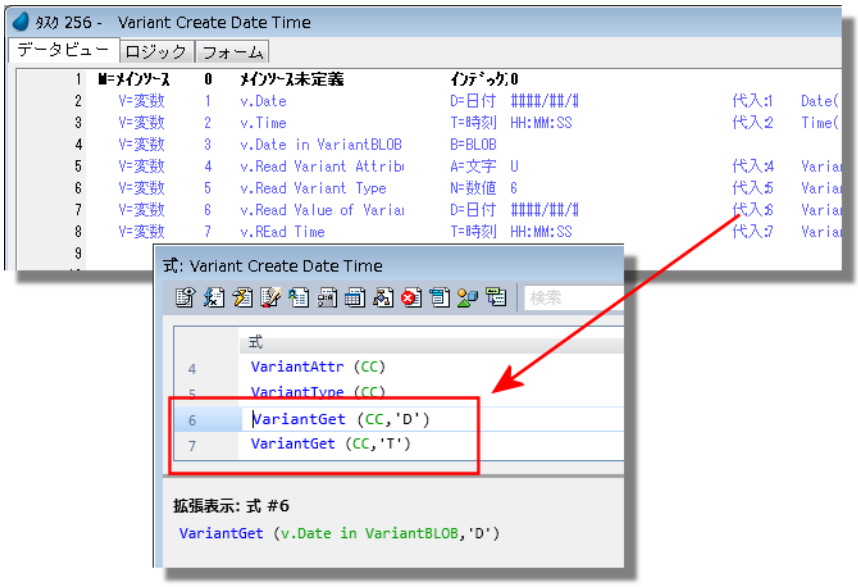
VariantCreate() の構文は以下の通りです。

VariantCreate(*VT Type*, *Value*, *Time value*)

パラメータ：

- *VT Type*：データタイプを表す数値です。例えば、この例では 7 が指定されています。データタイプの一覧は、「Variant のデータタイプ」（345 ページ）を参照してください。
- *Value*：Variant に設定する値です。どのようなデータも指定できます。
- *Time value*：（オプション）VT_Date 型に時刻を設定することが可能になります。この例では、３番目のパラメータとして時刻が設定されています。

Variant からデータを取得する



VariantGet() 関数の構文は以下の通りです。

VariantGet(*Variant Value*, *Attribute*)

パラメータ：

- *Variant Value*：Variant を表す BLOB 項目です。
- *Attribute*：取得するデータの型を表す文字です。コードの一覧は、「Variant の Magic データ型」（345 ページ）を参照してください。

この例では、Variant は日付と時刻を保存する VT_DATE 型で、**VariantGet()** 関数を使用して両方のデータを取得できます。

Variant の Magic データ型

型を表す文字	Magic データ型
A	文字型
N	数値型
L	論理型
D	日付型
T	時刻型
B	BLOB 型
U	Unicode 型

Variant のデータタイプ

Variant 関数で使用されるデータタイプには以下のものがあります。

タイプ値	列挙型のシンボル	内容	
0	VT_EMPTY	値が設定されていない	
1	VT_NULL	SQL スタイルの Null	

タイプ値	列挙型のシンボル	内容	
2	VT_I2	2 バイト整数 （符号付）	–32,768 to 32,767
3	VT_I4	4 バイト整数 （符号付）	–2,147,483,648 to 2,147,483,647
4	VT_R4	4 バイト不動小数点数	1.1E -38 to 3.4E +38 （7 桁）
5	VT_R8	8 バイト不動小数点数	2.2E -308 to 1.7 E +308（15 桁）
6	VT_CY	通貨型	
7	VT_DATE	日付型	
8	VT_BSTR	サイズ情報をもった文字列型	
9	VT_DISPATCH	IDispatch インターフェイス	
10	VT_ERROR	エラー型	
11	VT_BOOL	ブール型	
12	VT_VARIANT	バリアント型 （バリアント型配列にのみ使用）	
13	VT_UNKNOWN	IUnknown インターフェイス	
14	VT_DECIMAL	12 バイト数値（符号付）	
16	VT_I1	1 バイト整数（符号付）	–128 to 127
17	VT_UI1	1 バイト整数（符号なし）	0 to 255
18	VT_UI2	2 バイト整数（符号なし）	0 to 65,535
19	VT_UI4	4 バイト整数（符号なし）	0 to 4,294,967,295
22	VT_INT	int 型	
23	VT_UINT	unsigned int 型	
36	VT_RECORD	ユーザ定義型	
8192	VT_ARRAY		データ型の配列
16384	VT_BYREF		データ型への参照

COM オブジェクトの Variant 値のタイプと対応する Magic データ型を決定するには

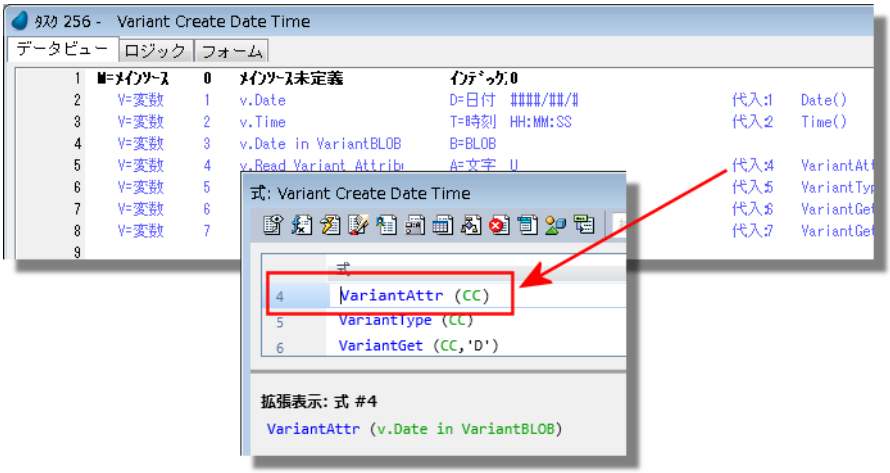
多くの COM オブジェクトは、VT_VARIANT タイプを使用しています。VT_VARIANT を定義することで、本質的には、どのようなデータタイプも保持できます。従って、このタイプのパラメータが設定されている場合、どのようなデータタイプが渡されるかを知る必要はありません。

実際は、試してみたり、オブジェクトに関するドキュメントを参照することで、どのようなオブジェクトが期待され送られるかを知ることができます。Magic xpa はデータの変換処理を行うため、手動で変換処理を行う必要はありません。例えば、数値型項目内に VT_VARIANT タイプを受け取るように設定しオブジェクトが数値を返した場合、その通りに処理は実行されます。

しかし、1 つのオブジェクトがある呼び出しに対して Variant 内に数値を返し、別の呼び出し対しては文字型データを返す場合があるかもしれません。このような場合は、BLOB データとしてデータを受け取り、内容を確認する必要があります。

- **VariantAttr()** 関数を使用することでこのような処理を行うことができます。この関数はパラメータとして BLOB 型を指定でき、Variant のデータタイプを表す文字が返ります。そのコードを **VariantGet()** 関数で使用することによりデータを取得することができます。**VariantType()** 関数を使用することもできます。この場合、Variant のデータタイプ（VT_I8 など）が返ります。

VariantAttr() 関数を使用してデータ型を取得する



VariantAttr() 関数の構文は以下の通りです。

VariantAttr(*Variant*)

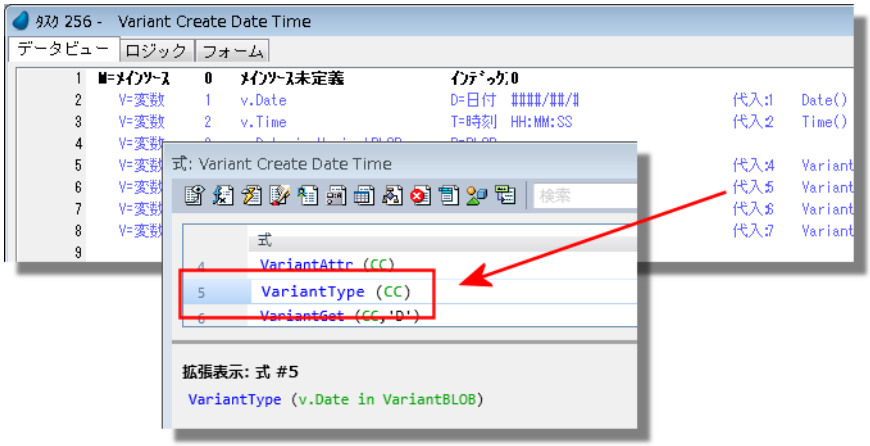
パラメータ：

- *Variant*：Variant が格納された BLOB 項目です。

この関数は、Magic xpa のデータ型を表す文字を返します。これらのデータ型の一覧は、「Variant の Magic データ型」(345 ページ) を参照してください。

この例では、VT_DATE タイプが指定されているため、D が返ります。これは Magic xpa の日付型を表しています。これでデータ型はわかりました。**VariantGet()** 関数を使用することで項目内の Variant からデータを取り出すことができます。

VariantType() 関数を使用してデータタイプを取得する



VariantType() 関数の構文は以下の通りです。

VariantType(*Variant*)

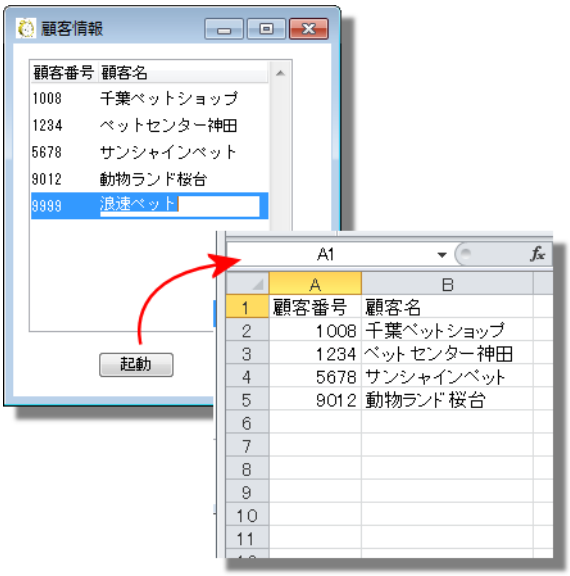
パラメータ：

- ***Variant***：Variant が格納された BLOB 項目です。

この関数は、Variant のデータタイプを表す文字を返します。これらのタイプの一覧は、「Variant のデータタイプ」（345 ページ）を参照してください。

この例では、Variant は VT_DATE タイプのため、7 が返ります。

COM オブジェクトから配列値を取り出したり設定したりするには



配列と COM オブジェクトを使用した処理を実行することで、Magic と COM オブジェクトの本当の力を知ることができます。上記の例では、わずかな記述を行うだけでテーブル内のデータが Excle のシートにコピーする処理を追加することができます。

COM オブジェクトの間で他のパラメータと同じようにベクトルデータを受け渡しします。COM オブジェクトに渡すデータが数値か文字列かを知る必要がありますが詳細内容（数値の場合、float か long かなど）を意識する必要はありません。

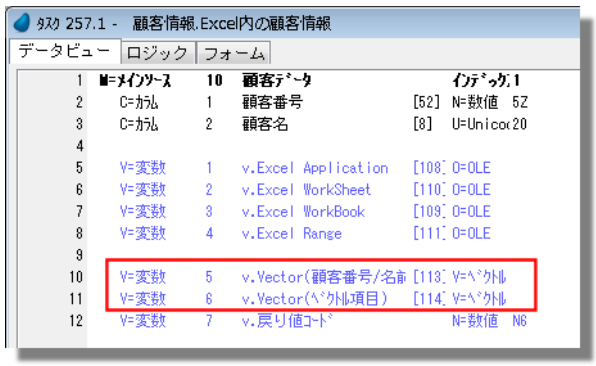
また、ベクトル内にいくつかの項目を定義する必要があります。この例では、何人のユーザがテーブルの中にいたかに見逃さないように、**Counter(*0*)** 関数を使用します。これによって、Excel シート内の正しい数を割り当てることができます。

COM オブジェクトに配列を渡す

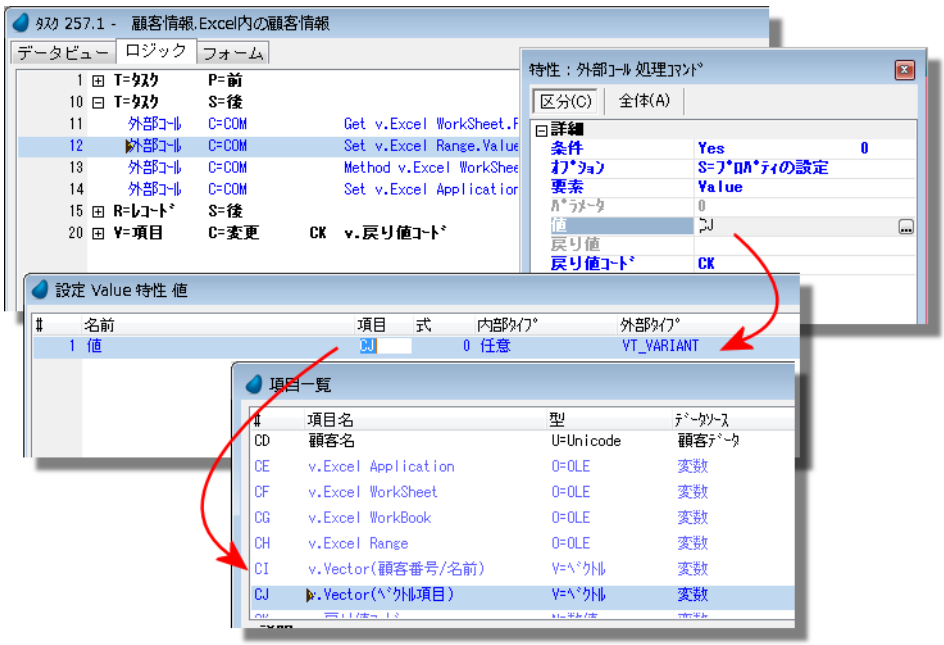
1. 最初に、ベクトル項目を設定します。この例では、２次元の配列を使用しています。各ラインにはユーザ番号とユーザ名を含んだ 1 次元配列を表しています。

   最初のベクトルは文字列の配列です。それがいっぱいの場合、2 つの要素（ユーザ番号とユーザ名）を含んでいます。これは、表計算の 1 行に相当します。

   2 番目のベクトルは、ベクトルで各行は、最初のベクトルのモデルを使用しています。これは表計算のシート全体に相当します。



2. 次に、COM オブジェクト内にベクトルを渡すために、**VecSet()** 関数を使用して、ベクトル内にデータを設定します。



3. ベクトルにデータが設定されると、簡単に渡すことができます。他のデータ項目と同じようにベクトルは単に渡されるだけです。

COM オブジェクト内のコレクションを処理するには

コレクションオブジェクトは、最近使用されたファイルや郵便番号、またはこの例のような「言語」といった項目のリストです。

リストにはユニークな識別子を含めなければなりません。識別子には、以下のものがあります。

- インデックス……1 からリスト内の項目数までの連続番号
- キー……コレクション内の項目を識別するためのユニークな文字列（または数字）

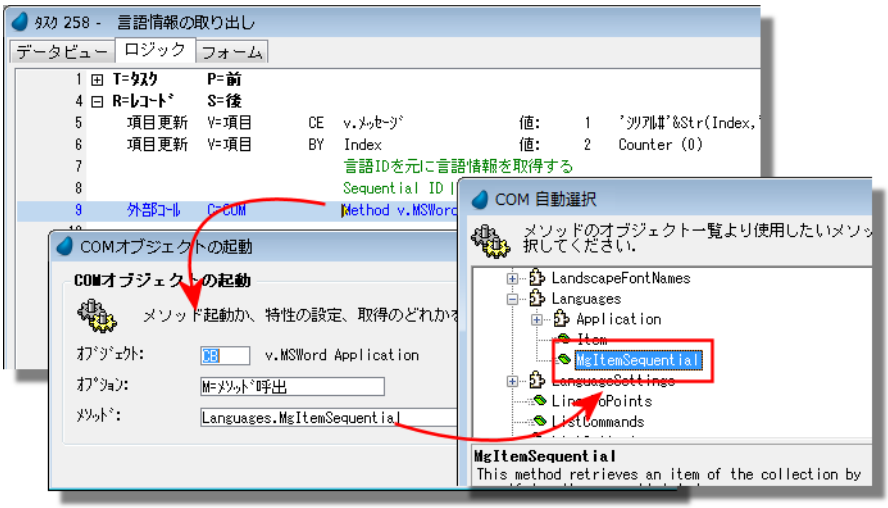
各コレクションはインデックスかキーのどちらかを含みます。両方を含めることはありません。上記の例では、Microsoft Word からの言語リストには ID のキータイプが含まれています。ID が何であるかは分からないので、リスト全体を抽出することは難しくなります。

しかし、**MgItemSequential** と呼ばれるコレクションのために特別なメソッドを使用することで、この問題は解決されます。このメソッドはネイティブのオブジェクトの一部ではありませんが、他のすべてのメソッソと一緒にメソッド一覧に表示されます。それは、1 から項目数分のインデックスを使用して、リストから項目を取り出すことが」できます。項目がテーブル内にある場合、上記のように Magic xpa での他のテーブルと同じように使用することができます。

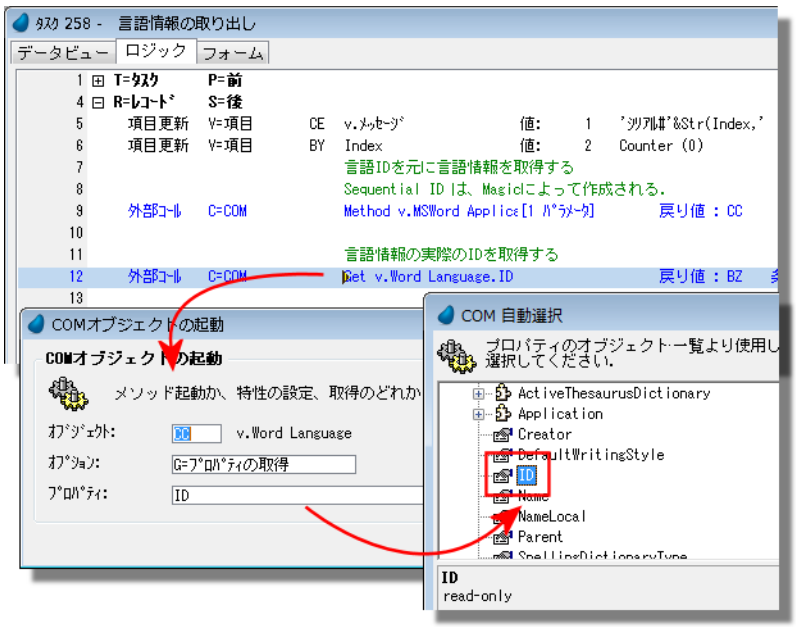
以下の例は、Microsoft Word からサポートされた言語リストを取り出すものです。

コレクションを取り出す

1. 最初に、コレクション全体のサイズを決定します。この例では、Word Application.Languages オブジェクト内に Count と呼ばれるメソッドがあります。このメソッドは言語の総数を返します。このメソッドは**タスク終了条件**特性で使用され、言語リスト全体を取り出すまで、タスクはループされます。



2. リスト内の各項目に対して、項目を取り出すために MgItemSequential メソッドを使用します。MgItemSequential は、ネイティブなオブジェクトには存在していませんが、COM の**自動選択一覧**に表示されます。
   戻りのパラメータはオブジェクトを戻します。この場合は、Word Application.Language オブジェクトが返ります。



3. 次に、オブジェクトのプロパティを取得するために MgItemSequential によって返されたオブジェクトを使用します。この場合に、ID と名前を取得します。これらはテーブル内に格納されています。

タスクが実行されると、利用可能な Magic テーブル内にコレクションが格納されます。

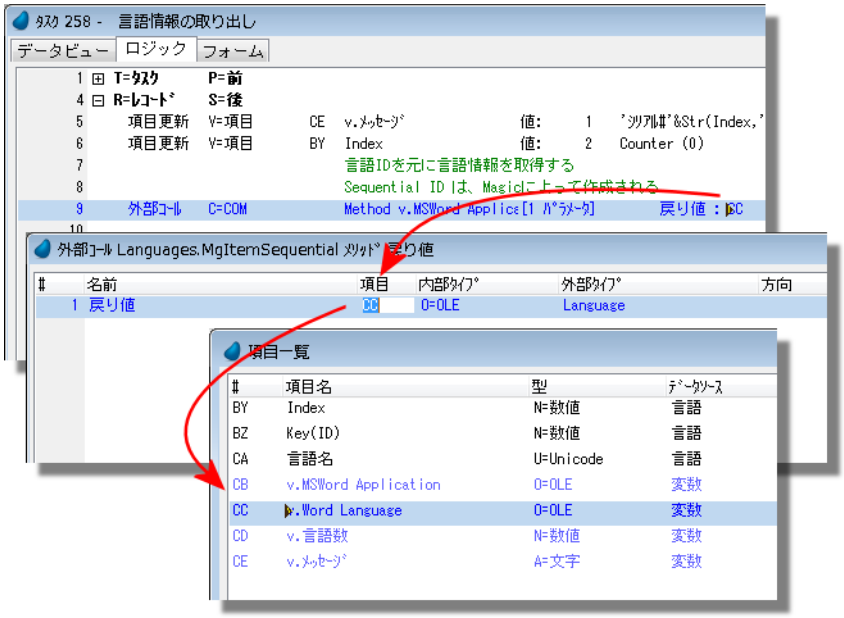
COM オブジェクト間でパラメータとして COM オブジェクトの受け渡しを行うには

COM オブジェクトはパラメータとして頻繁に他の COM オブジェクトを受け渡します。Magic xpa ではこのような処理を簡単に行うことができます。COM オブジェクトはデータ項目のため、他のデータ項目と同じように渡すことができます。

COM オブジェクトを受け取る

この例は、「コレクションを取り出す」（352 ページ）で説明した MgItemSequential メソッドを使用します。

必要条件： プログラムがアクセスできる場所に COM オブジェクトを定義しておく必要があります。



1. パラメータの受け渡しを行う場所に移動します。この場合、戻り値として COM オブジェクトを受け取ります。
2. 他の項目と同じように**項目**カラムから**ズーム**します。COM オブジェクトが一覧表示され、***Enter*** を押すことで選択できます。

これでメソッドが起動されると、次の**外部コール**処理コマンドによって WordApp.Language という COM オブジェクトが利用できるようになります。

COM オブジェクト定義を再利用するには

品質がよく、信頼性が高く、保守が容易なアプリケーションを開発するには、なるべく項目を再利用することです。これはエラーを減らすだけでなく、アプリケーションの保守性を容易にし、またプログラムを早く作成することができます。Magic xpa は、モデルリポジトリを使用することで、データ定義やコントロール定義を簡単に再利用することができます。

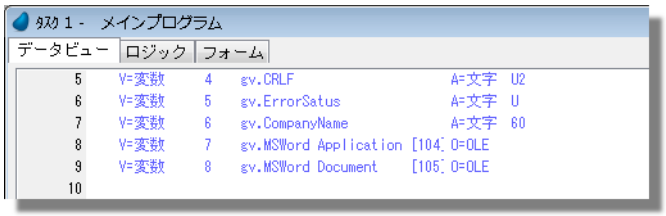
COM オブジェクト定義は、タスク内に直接定義するように、モデルリポジトリに定義します。唯一の違いは、一度定義することで、任意に再利用できるということです。また、モデルリポジトリでオブジェクトが定義された場合、クロスリファレンス（*Ctrl+F*）を使用してそのモデルが使用されているオブジェクトを検索することができます。

モデルリポジトリ内に COM オブジェクトを設定する方法については、「使用する COM オブジェクトを定義するには」（337 ページ）を参照してください。

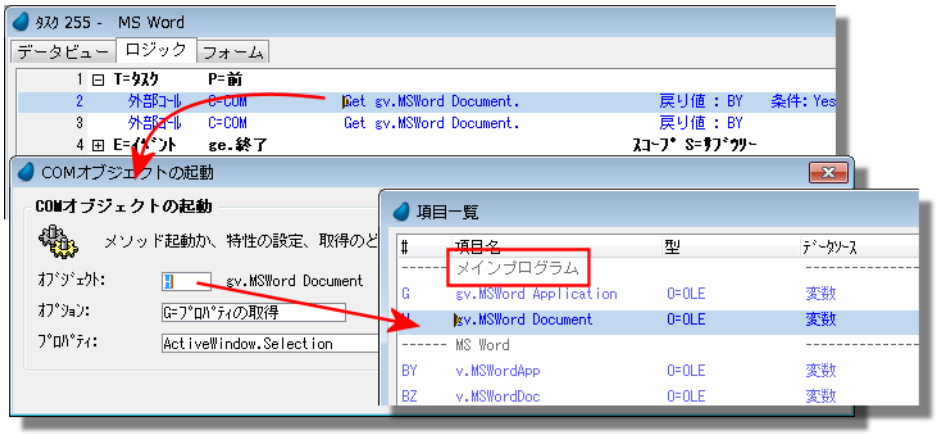
プログラム全体に渡って COM オブジェクトのインスタンスを保持するには

COM オブジェクトのインスタンスを生成する必要があり、異なるプログラムが実行中もそれをオープンしたままにする必要があるかもしれません。このためには、メインプログラム内で COM オブジェクトを定義します。これによって、プログラムリポジトリ内のどのプログラムでも利用可能になります。必要であれば、オブジェクトを初期設定するためにメインプログラムのタスク前を使用することで、プログラムが最初に実行するとインスタンスが使用できる状態になっています。

メインプログラム内で COM オブジェクトを定義する



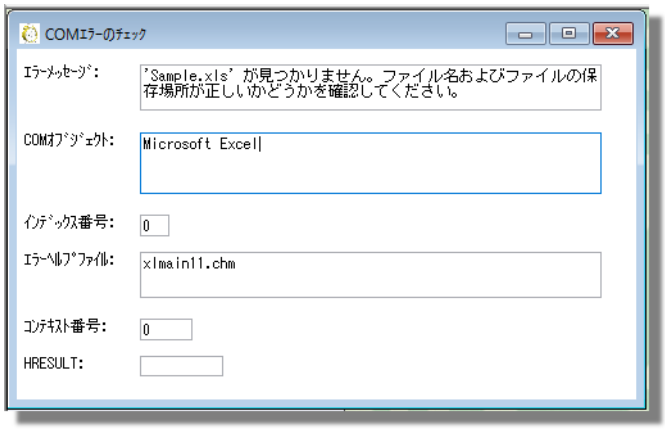
1. タスク内に COM オブジェクトを定義するように、メインプログラムのデータビューエディタでオブジェクトを定義します。



これで、COM オブジェクトを選択する際に、項目一覧の先頭のメインプログラムのヘッダの下に COM オブジェクトが一覧表示されます。

注： COM オブジェクトは、パラメータとしてプログラムの間で渡すことによって共有することもできます。

COM オブジェクトによって発生したエラーを処理するには



COMError() 関数を使用して COM オブジェクトで発生したエラー情報を受け取ることができます。**COMError()** は最後に発生した COM エラーの内容を返します。構文は以下のとおりです。

COMError(*number*)

パラメータ：

number は、返される情報の種類を指定します。戻り値は文字列で返ります。

- 1 …… エラーの内容
- 2 …… COM オブジェクトの名前
- 3 …… インデックス番号
- 4 …… エラーに対するヘルプファイル
- 5 …… コンテキスト番号
- 0 …… HRESULT コード

ヒント : *プログラムやユーザ定義関数内にこの関数を定義することでカプセル化することができます。これによって、エラーをチェックしたり、エラーが見つかったときにメッセージを表示させたり、ログを出力したりすることができます。また、COM オブジェクトをデバッグしている間、この情報が Magic xpa のアクティビティログに出力されます。*

第 15 章 : コンポーネント

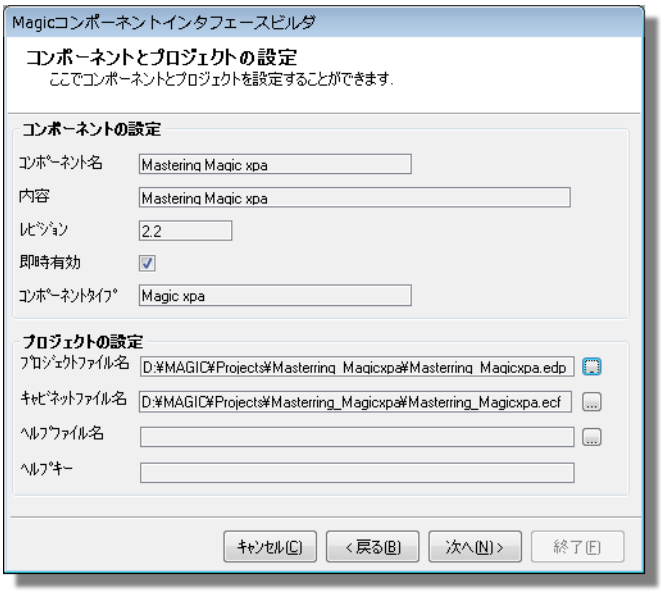
プロジェクト間で Magic xpa のオブジェクトを再利用するには

Magic xpa Studio は、1 つのプロジェクト内でオブジェクトを再利用する機能があります。リポジトリ内にモデルやデータソースを定義することにより、プログラム作成する工数を減らすことができます。しかし、複数のプロジェクト間で再使用可能なモデルやデータソース、およびプログラムのライブラリを作成することにより同じように再利用が可能になります。この場合、使用するプロジェクトを変更せずにライブラリをアップグレードすることができます。

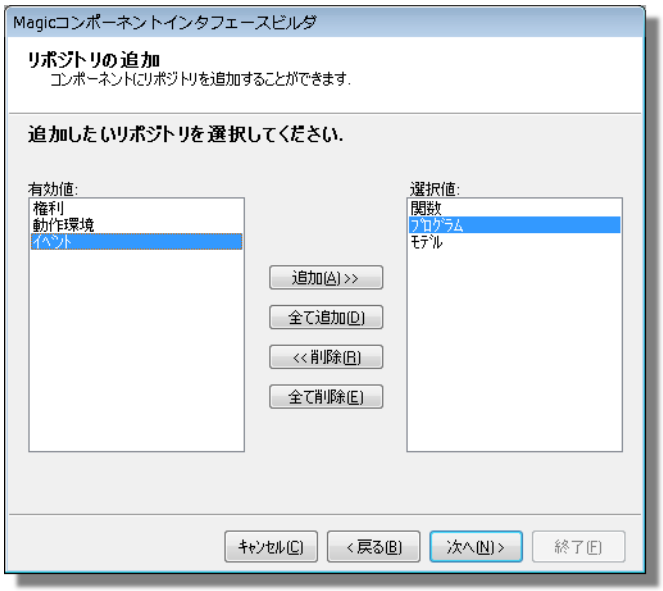
これらのライブラリは、Magic コンポーネントと呼ばれます。どの Magic プロジェクトからでもコンポーネントを作成し、***.edp*** または ***.ecf*** ファイルのどちらかでコンポーネントを利用することができます。アプリケーションのどの部分がどのコンポーネントを使用しているかということを管理することができるため、異なる機能を持った異なるコンポーネントを作成することができます。いくつかの機能は、廉価バージョン用に利用することができるため、特に商業用のアプリケーションを開発する上で有益です。

Magic コンポーネントを作成する

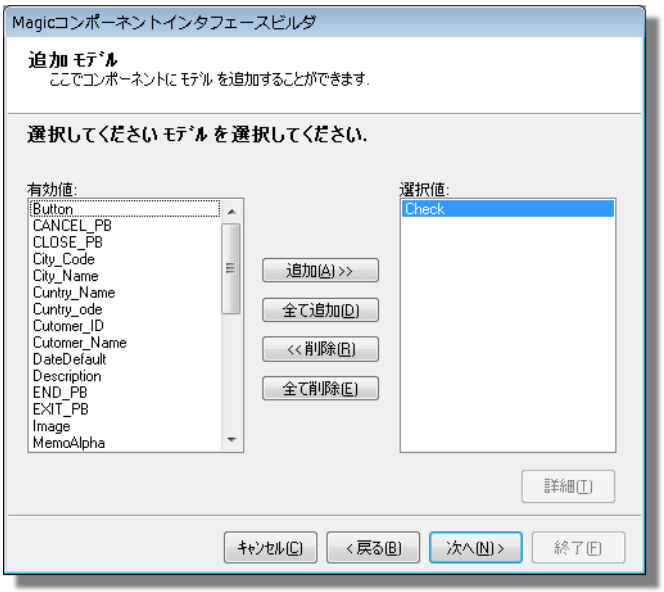
1. コンポーネントとして公開するオブジェクトを決定します。それらのオブジェクトに公開名を指定します。プログラムの場合は、更に**外部**カラムをチェックする必要があります。
2. **オプション→インタフェースビルダ→ Magic xpa** を選択します。**Magic コンポーネントインタフェースビルダ**のウィザードが起動されます。次へをクリックします。
3. 次の画面で、既存のコンポーネントを修正したり、新規作成することができます。新規コンポーネントを作成するために、新規ボタンをクリックします。



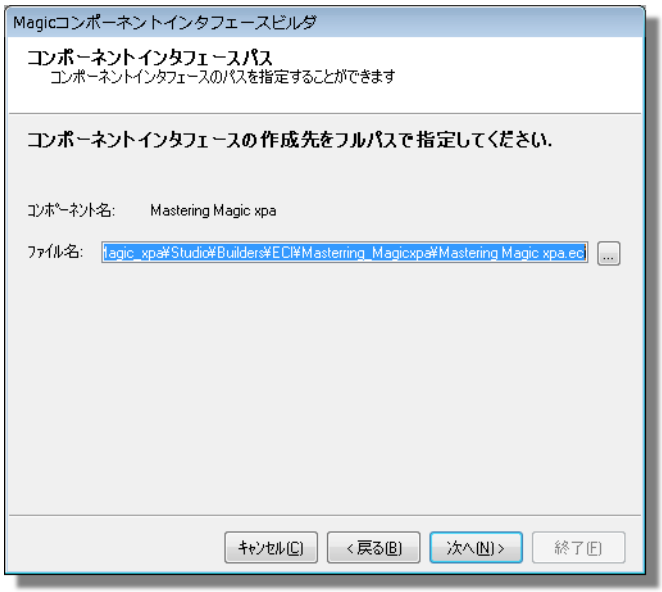
4. 次に、**コンポーネントとプロジェクトの設定**ダイアログが表示されます。ここには指定可能な多くの項目が表示されます。*F1* をクリックするとより詳細な説明が表示されます。ここでは**コンポーネント名**と**プロジェクトファイル名**を指定しなければなりません。次へをクリックします。



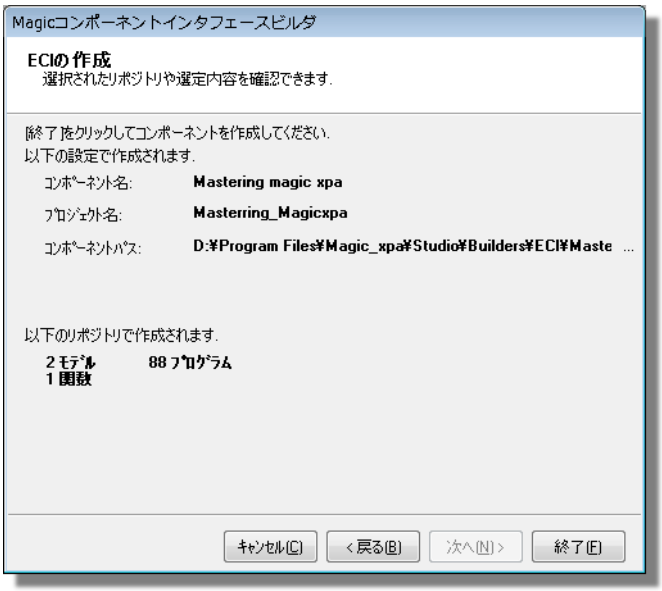
5. 次に、**リポジトリの追加**ダイアログが表示されます。ここでは、コンポーネントに表示されている中から使用したいリポジトリを選択できます。「有効」から「選択済み」へ項目を移動するために、追加 >> や全て追加のボタンをクリックします。次へをクリックします。



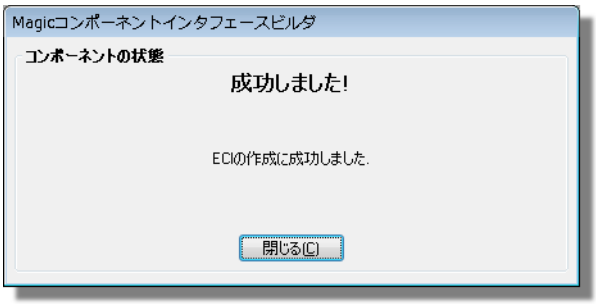
6. 次に、選択された各リポジトリ内のオブジェクトを選択することのできる一連のダイアログが表示されます。コンポーネントとして必要な項目のみを選択します。ここでは、モデルをコンポーネントに追加しています。



7. . 次に、作成する ***.eci***（eDeveloper Component Interface）ファイルのパスを入力するダイアログが表示されます。これは、コンポーネントを使用するプロジェクトがアクセスする場合に必要なファイルです。次へを**クリック**します。



8. 最後に、確認のために選択内容が表示されるダイアログが表示されます。コンポーネントを作成する場合は終了ボタンを**クリック**し、選択内容を変更するに場合は戻るボタンを**クリック**します。



9. ***.eci*** ファイルが生成されると成功しましたメッセージが表示され、コンポーネントが使用できるようになります。

注： *.eci* ファイルはテキストファイルで、Magic.ini ファイルに似た構成になっています。必要に応じて、手動で編集することができます。

これで、コンポーネントが利用できる準備が整いました。

参照 : 「プロジェクトにコンポーネントを読み込むには」（362 ページ）

オブジェクトを公開するには

コンポーネントの作成中に、コンポーネントとして何を公開し、何を公開しないかを指定できます。コンポーネントとして公開されるには、オブジェクトは以下の基準を満たす必要があります。

1. オブジェクトには、**公開名**が定義されなくてはなりません。
2. オブジェクトがプログラムの場合、**外部**カラムのボックスがチェックされていなければなりません。
3. コンポーネントの作成中に項目を選択する必要があります（「Magic コンポーネントを作成する」（357 ページ）に説明があります）。

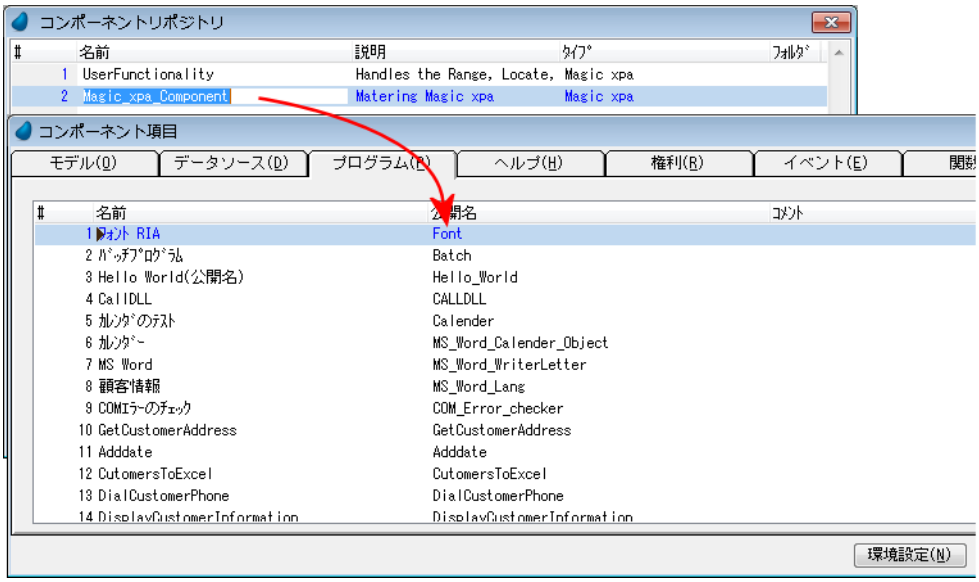
参照： 「プロジェクト間で Magic xpa のオブジェクトを再利用するには」（357 ページ）

プロジェクトにコンポーネントを読み込むには

プロジェクトにコンポーネントを読み込む前に、*.eci* ファイルを作成する必要があります。これは、プロジェクト内にコンポーネントを読み込む際に使用されるテキストファイルです。*.eci* ファイルの作成方法は、「Magic コンポーネントを作成する」（357 ページ）を参照してください。

.eci ファイルが作成されたら、プロジェクトで利用可能にするために以下の手順で読み込み処理を行います。

Magic コンポーネントを使用する



1. プロジェクト→コンポーネント（*Shift+F7*）を選択します。コンポーネントリポジトリが開きます。
2. *F4*（編集→行作成）を押下して 1 行追加します。使用するコンポーネントの名前を入力します。この名前は、コンポーネントの名前と同じにする必要はありません。これは表示用として使用されます。
3. ズーム（*F5*、またはダブルクリック）してこのコンポーネントの内容が記述された *.eci* ファイルを選択します。
4. 再度ズームすると、すべてのコンポーネント項目が一覧表示されます。コンポーネント用の動作環境を設定するボタンが下にあり、コンポーネント内の各アイテム毎のタブが表示されます。
5. これらの一覧内の各項目からズームすることでモデル特性やプログラムに対するパラメータなどの情報が表示されます。しかし、プロジェクト内から特性値を変更することはできません。使用することだけ可能です。

これで、他のオブジェクトを使用するように、プロジェクト内でコンポーネント項目を使用することができます。例えばモデルを選択すると、そのプロジェクト内にあるモデルと同じ選択一覧にこのコンポーネントが表示されます。

注： 実行時は、*.eci* ファイルは不要です。コンポーネントが同じパス上に存在していることを確認する必要がありますが、一端コンポーネントを読み込むと、*.eci* ファイルは使用されません。

ホストアプリケーションで使用している既存のコンポーネントの変更を有効にするには

コンポーネントの修正内容を反映させるには、コンポーネントを作成し直し、ホストプロジェクトでこれを再読込する必要があります。変更内容を反映させるには 2 つの方法があります。 1 つ目は、実行環境のコンポーネントを差し替える場合で、ホストアプリケーションは変更されません。 2 つ目は、開発環境でコンポーネントを使用していて、コンポーネント内の新しい項目を追加する場合です。両方の場合に対して以下で説明します。

実行環境でコンポーネントを変更する

実行環境にインストールされているコンポーネントを変更する場合は、古いコンポーネントを新しいものに交換するだけです。例えば、コンポーネントのバグを修正したり、より速く動作するような処理を組み込んだり、その他の内部処理を修正した場合、このような変更作業が発生します。オブジェクトとそれらのパラメータの名前が変わらない限り、ホストアプリケーションは修正する必要がありません。

項目をコンポーネントに追加しても、ホストアプリケーションは新しいオブジェクトを認識しません。開発時に追加されたオブジェクトを使用していないため、これは認識されておらず、何も変わりません。しかし、既存のオブジェクトの公開名を変更したり、プログラムのパラメータを変更した場合、致命的なエラーが発生する場合があります。

開発環境でコンポーネントを変更する

開発エンジンでアクセスしているコンポーネントの内容が変更された場合、変更内容を反映させるため ***.eci*** ファイルを変更する必要があります。変更しないのであれば追加されたオブジェクトを参照しないようにします。このような場合、以下の手順で作業する必要があります。

1. コンポーネントが自分で作成したものであれば、新しい ***.eci*** ファイルを作成し直します。この操作は、最初から作成するのではなく既存のコンポーネントを修正する作業になります。詳細は、「Magic コンポーネントを作成する」（357 ページ）を参照してください。もし別のサードパーティから受け取ったコンポーネントであれば、新しいコンポーネントと一緒に新しい ***.eci*** ファイルを取得します。
2. **プロジェクト→コンポーネント**（***Shift+F7***）を選択し、更新したいコンポーネント上にカーソルを置きます。
3. **オプション→コンポーネントの読込 / 再読込**を選択します。使用する ***.eci*** ファイル名を選択するダイアログが表示されます。***.eci*** ファイルを選択し、開くを**クリック**します。

これでコンポーネント一覧は再読込され、新しいコンポーネントを使用して動作することができます。

ヒント : ***古いコンポーネントを削除してから再読込を行わないでください。このような操作を行うと、コンポーネントオブジェクトへのすべての参照情報が失われます。***
読込 / 再読込の操作によって参照内容が再表示されます。

コンポーネント内のオブジェクトの名前を変更する

Magic xpa Studio 内では、オブジェクトの名前は固定ではありません。これは内部の参照システムを使用して項目を参照しているからです。しかし、オブジェクトを他のプログラムで利用可能になるように設定した場合、他のプログラミングツールと同じように、実際のテキスト名にもとづいて参照されます。

このため、コンポーネントオブジェクトの名前を変更する良い方法がありません。オブジェクトの名前を変更し、コンポーネントを再読込した場合、オブジェクトへのすべてのリンクが破損してしまい、オブジェクト名カラムに「項目は有効ではありません」が表示されます。

従って、コンポーネントを設計する際に、COM ライブラリの開発と同じような考え方で開発することを推奨します。オブジェクトの公開名を変更したり、プログラムに渡すパラメータを変更したり、オブジェクトを削除しないようにしてください。例えば、新しい改善されたカレンダオブジェクトを組み込みたい場合、新しく作成したカレンダ 2 を呼ぶようにしてから、オリジナルのカレンダを削除するようにします。

コンポーネント用のヘルプファイルを提供するには

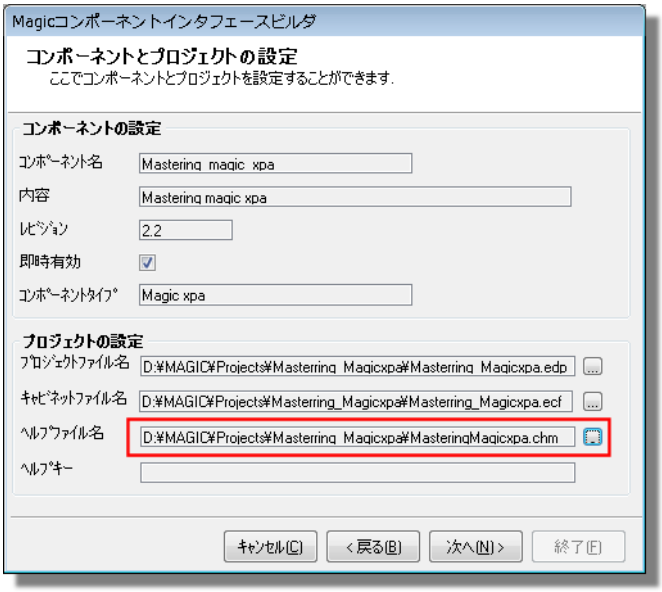
コンポーネント用のヘルプファイルを提供することで、コンポーネントの内容が理解されやすく、使用しやすいものになります。

Windows のヘルプファイルを作成するには、サードパーティ製のオーサリングツールが必要となります。すべてのヘルプ情報を入力し、コンパイルして、.chm（ヘルプ）ファイルを作成します。ヘルプファイルの内部には、各ヘルプトピックに対するユニークな数値インデックス（マップ ID）や複数のキーワードを定義することができます。

Magic xpa のコンポーネントにおいて、このインデックスやキーワードを設定することで、コンポーネントリポジトリ上で ***F1*** を押下すると関連するヘルプトピックを表示させることができます。

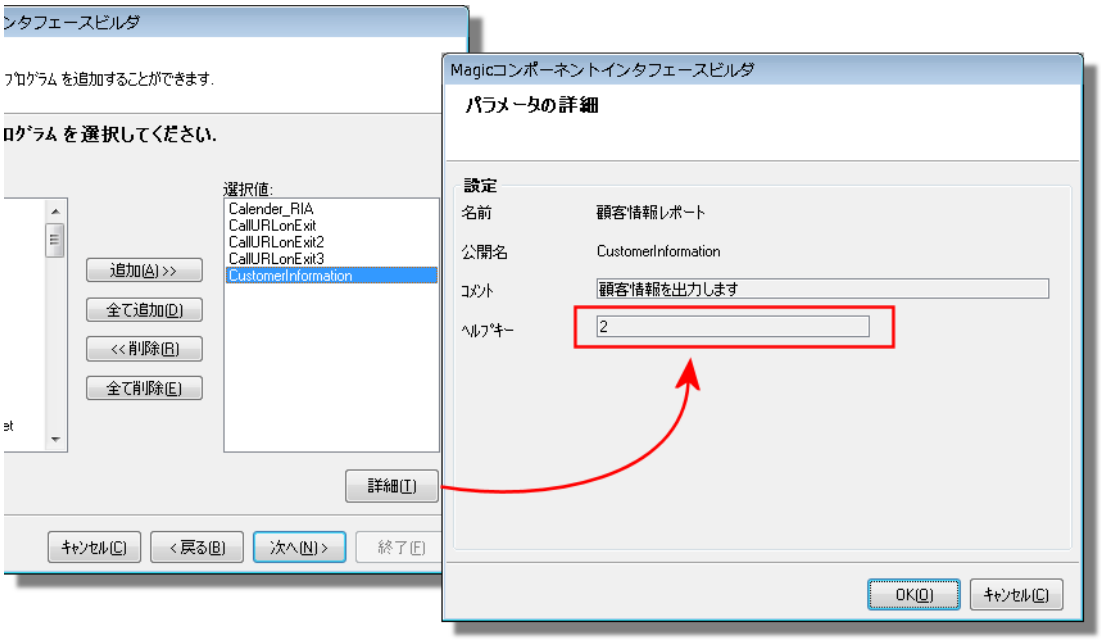
コンポーネントを作成する際に、ヘルプファイルが存在する場所の指定と、どのような状況でどのインデックスやキーワードを割り当てるかをあらかじめ検討しておく必要があります。

ヘルプファイルを組み込む



1. コンポーネントとプロジェクト設定画面で、使用したいヘルプファイルの名前とパスを指定します。パス名を直接入力するよりも、%WorkingDir% のような論理名を使用することを推奨します。
2. **ヘルプキー**には、表示させたいヘルプトピックのインデックス（マップ ID）となる数値かキーワードとなる文字列を指定します。

注：　ヘルプファイルと**ヘルプキー**の両方が設定されていない場合、Magic xpa のリファレンスヘルプが表示されます。

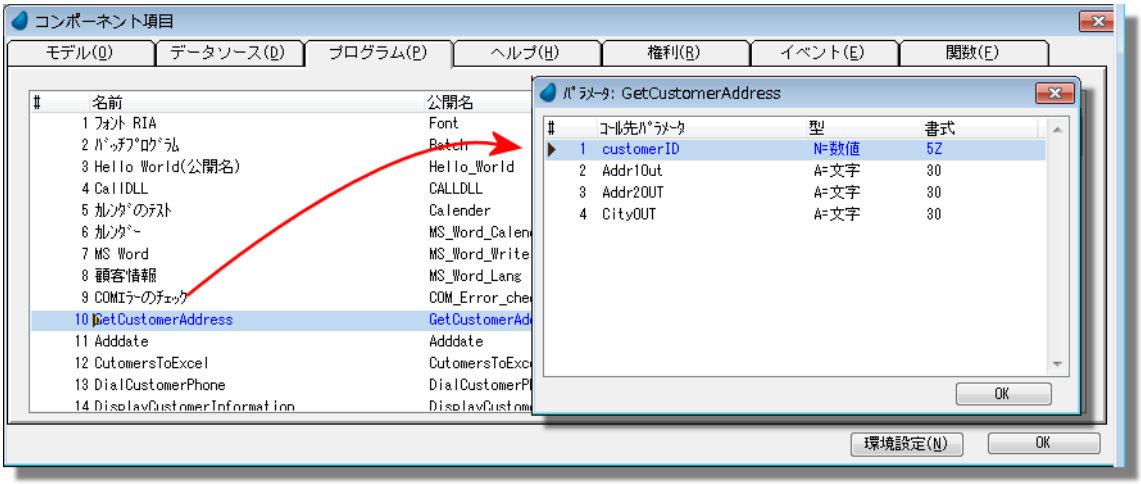


3. ヘルプを追加したいオブジェクトに到達するまでウィザードを進めます。
4. オブジェクトに対する詳細ボタンを**クリック**します。パラメータや**ヘルプキー**を追加するダイアログが表示されます。このオブジェクトに対応するヘルプトピックのマップ ID やキーワードを入力します。

これで、このコンポーネントが組み込まれると、**コンポーネント**リポジトリの**コンポーネント項目**テーブルにおいて該当する項目にカーソルが位置付けられている状態で ***F1*** を押下すると、指定されたヘルプトピックが表示されます。

また**コメント**に入力された内容も**コンポーネント項目**テーブルに表示されます。

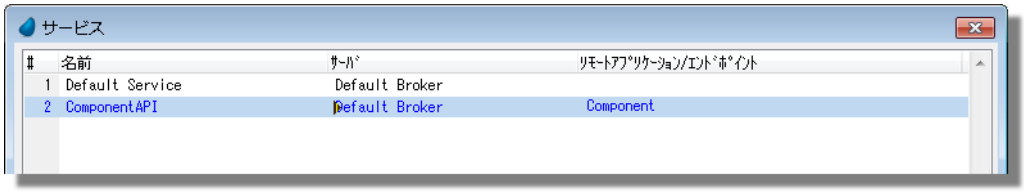
コンポーネントのオブジェクトに関する詳細情報を参照するには



プロジェクト内でコンポーネントを使用している場合に、各オブジェクトで**ズーム**する（***F5***、または**ダブルクリック**）ことでオブジェクトに関する詳細情報を参照することができます。例えば、図の例では顧客住所の取得から**ズーム**するとそのオブジェクト（プログラム）に対する**パラメーター一覧**が表示されます。

これ以外の情報は表示されません。コンポーネントは「ブラックボックス」として扱われ、実行する場合に必要な情報以外は表示されません。

コンポーネントが存在するディレクトリにアクセスするには



コンポーネントが実行している間は、**ProjectDir()** 関数を使用することでコンポーネントの存在するディレクトリにアクセスすることができます。

この関数は実際のパス名を返します（論理名ではありません）。%WorkingDir% や %TempDir% などのディレクトリ情報を表す論理名は、ロードされたすべてのコンポーネントに対して同じ値で評価されますが、**ProjectDir()** 関数は、実行中のコンポーネントに対応したパス名が返ります。

現在実行中のアプリケーションがコンポーネントかどうかを確認するには

Magic xpa のプロジェクトは、どのように組み込むかによって、ホストまたはコンポーネントとして実行させることができます。従って、1 つのプロジェクトファイルに 2 つの役目を持たせることができます。その際、どのモードで実行しているかを **IsComponent()** 関数を使用して確認することができます。関数の構文は以下の通りです。

IsComponent()

戻り値：タスクがコンポーネントとして実行している場合は、True が返ります。

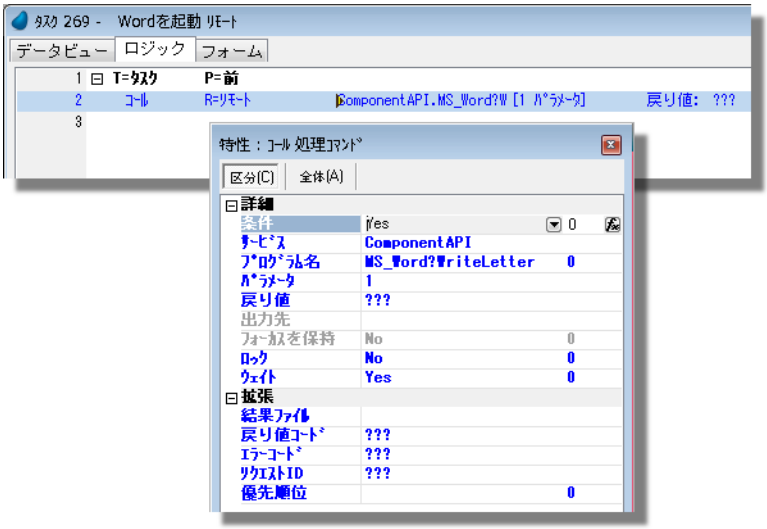
コンポーネントとして定義されない別のアプリケーション内のプログラムを呼び出すには

コンポーネントを使用しないで別のアプリケーション内のプログラムを呼び出すこともできます。COM オブジェクトや SOAP サービスとして呼び出されるプログラムを組み込む場合を含めて、いくつかの方法があります。しかし、最も直接的な方法として、**コールリモート**と**コール公開名**の２つの処理コマンドを使用する方法があります。

これらのタイプの呼び出しを使用する場合、実行モードで起動されるまで起動するオブジェクトの情報を持っていないことに注意してください。従って、名前（プロジェクトファイル名や公開名）とパラメータが正しく設定されているかどうかは、開発者の責任になります（Magic xpa ではチェックされません）。

コールリモートを使用する

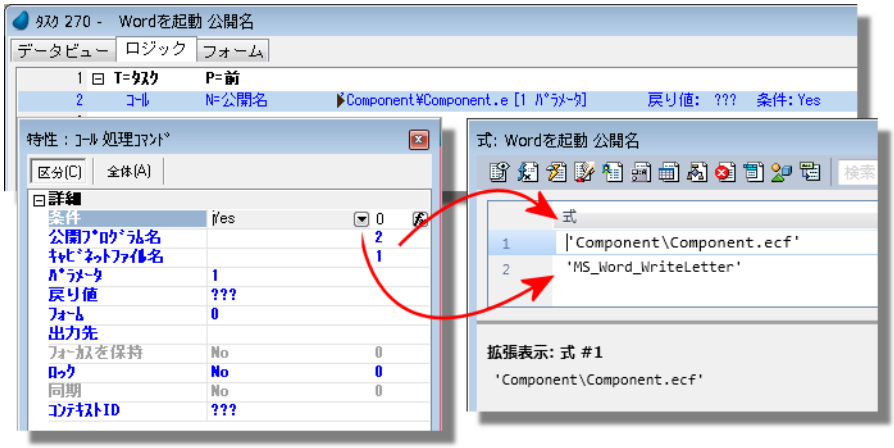
1. 最初に、**サービス**テーブル（**オプション→設定→サービス**）に他のアプリケーションを表すサービスを設定します。**リモートアプリケーション / エンドポイント**カラムから**ズーム**して、使用したいアプリケーションを選択します。
2. 次に、タスクに**コールリモート**処理コマンドを定義します。
   - 任意のヘッダ行内で ***F4*** を押下して 1 行追加します。
   - ***C*** を入力して**コール**処理コマンドを選択します。
   - ***R*** を入力して ***R= リモート***を選択します。



3. **処理コマンド**特性（***Alt+Enter***、または**特性**シートを**クリック**）を開きます。
4. **サービス**特性から**ズーム**して、実行させたいアプリケーションを表すサービスを選択します。
5. **プログラム名**特性を入力します。ここには、指定されたアプリケーションに定義されているプログラムの公開名を入力します。開発者の責任で正しく入力する必要があります。
6. **パラメータ**特性から**ズーム**してプログラムに必要なパラメータを定義します。この設定も開発者の責任で正しく行ってください。自動的に比較しません。

これで、プログラムが実行されると、他のアプリケーションのプログラムを呼び出すことができます。

コール公開名を使用する

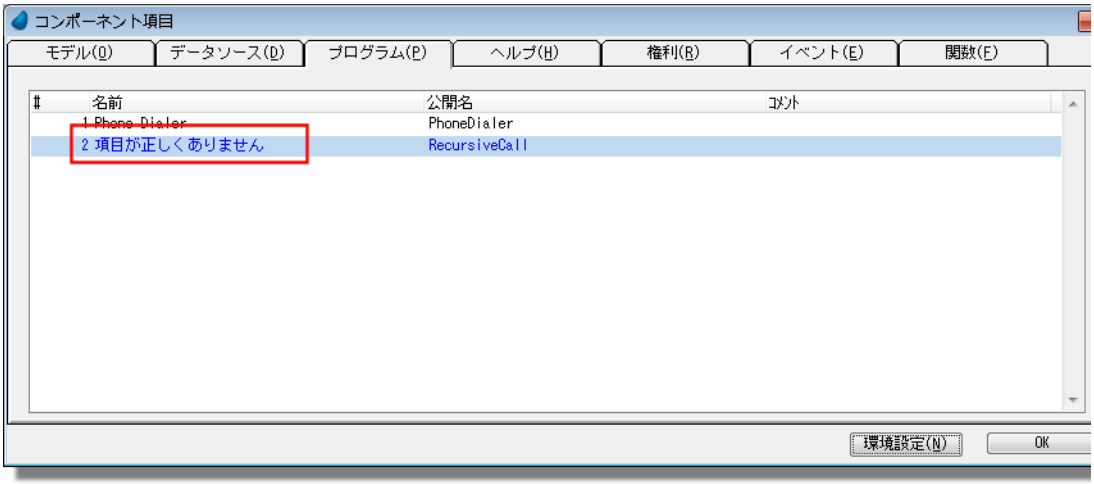


1. タスクにコール公開名処理コマンドを定義します。
   - 任意のヘッダ行内で *F4* を押下して 1 行追加します。
   - *C* を入力してコール処理コマンドを選択します。
   - *N* を入力して *N= 公開名*を選択します。
2. 処理コマンド特性（*Alt+Enter*、または特性シートをクリック）を開きます。
3. **公開プログラム名**特性から式エディタにズームします。実行させたいプログラムの公開名を定義します。
4. **キャビネットファイル名**特性でズームしてプログラムが定義されているキャビネットファイルを指定します。
5. **パラメータ**特性からズームしてプログラムに必要なパラメータを定義します。

これで、プログラムが実行されると、指定されたキャビネットファイルのプログラムを呼び出すことができます。

ヒント :*パス名を直接入力する方法は、開発時には内容が確認できますが、メインプログラム で定義された変数にパス名を格納したり、論理名を使用することを推奨します。*

アプリケーション間で再帰呼び出しを処理するには

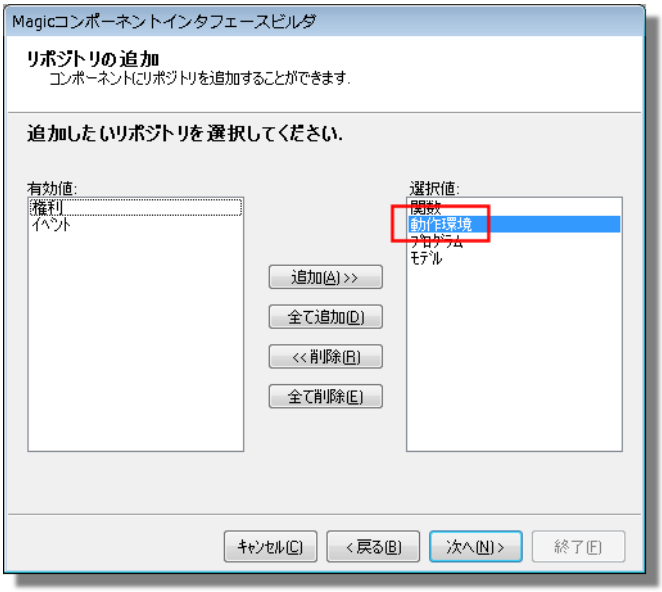


コンポーネント間の再帰呼び出しはできません。現在のアプリケーション内のオブジェクトを参照するコンポーネントオブジェクトを読み込もうとした場合、コンポーネント項目上に“項目が正しくありません”のメッセージが表示されます。

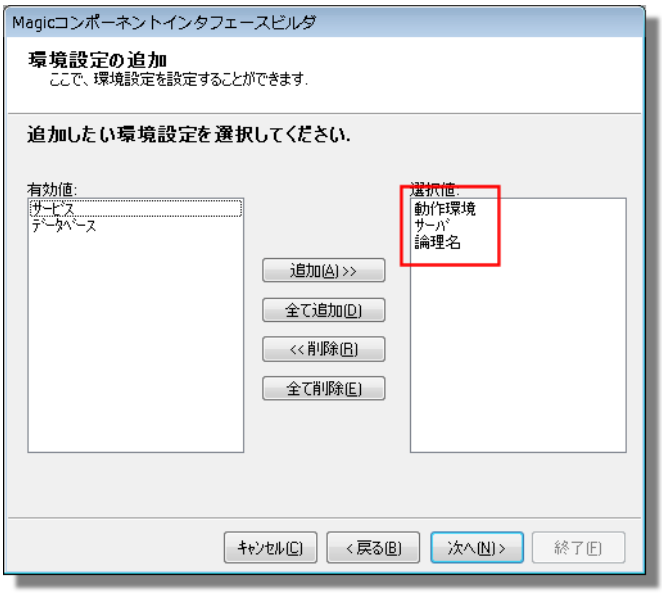
上記の例の“RecursiveCall”プログラムは、現在のアプリケーション内で定義されたプログラムを呼び出すプログラムのため、これを含めることができません。

一般的に、アプリケーションを構造化することはよい考えです。このためこのようなアプリケーションは再帰的な呼出しを必要としません。例えば、多くのアプリケーション間で使用されるユーティリティのライブラリを持つことは可能ですが、ユーティリティはそれらのアプリケーション内のプログラムを呼び出すことはしません。

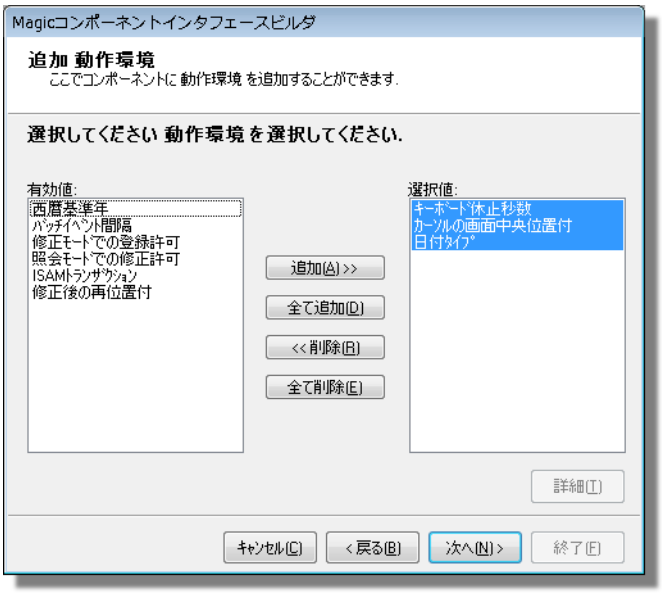
コンポーネントの動作環境を組み込むには

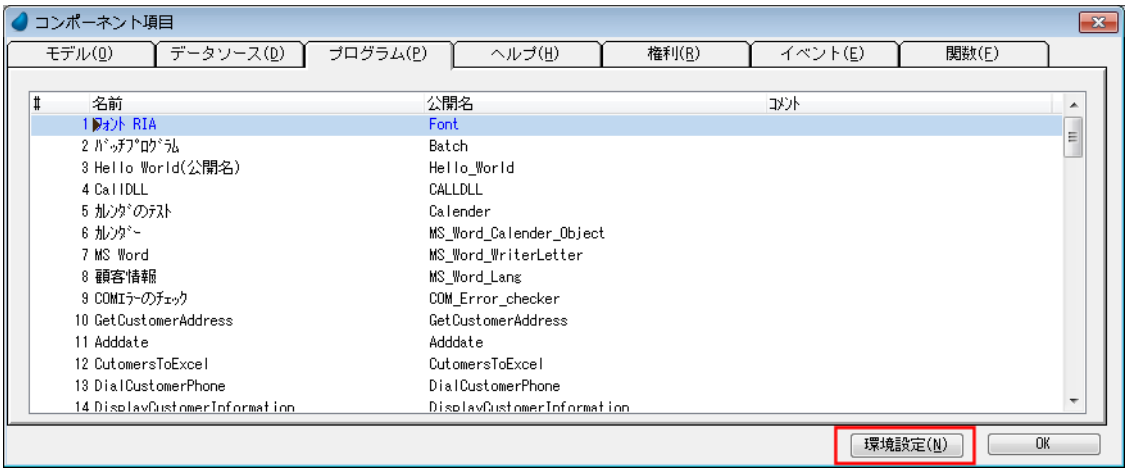


コンポーネントを作成する際に、そのコンポーネントのために必要な動作環境を設定することができます。ウィザードの**リポジトリ選択**で***動作環境***を選択すると、各環境情報を個別に指定する一連のダイアログが表示されます。



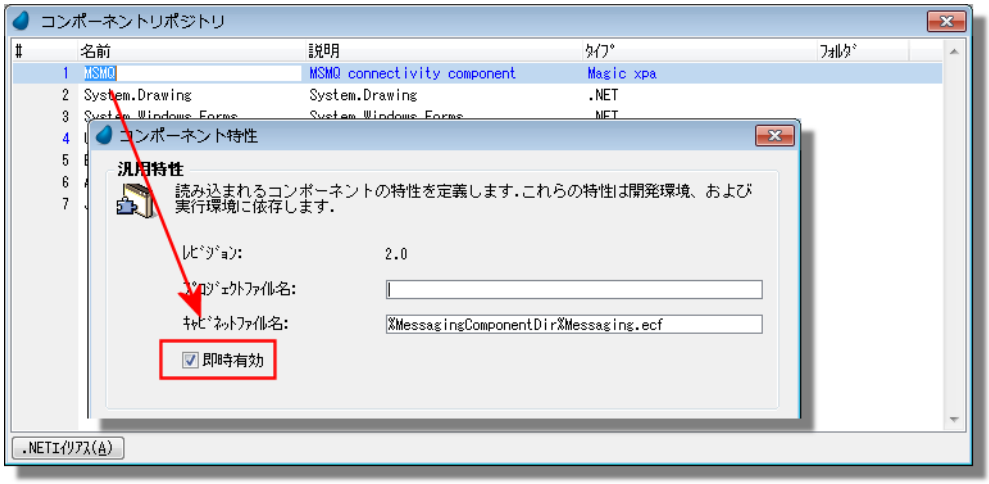
この例では、コンポーネントの動作環境、サーバ、および論理名の各情報を組み込まれるコンポーネントの一部でになるように定義しています。これらの各項目に対して、詳細な設定ダイアログが開き、必要な情報を設定することができます。





コンポーネントが使用されると、環境設定ボタンをクリックすることで設定内容が表示されます。

コンポーネントへのアクセスを最適化するには



コンポーネントは、最初に実行される前にメモリにロードされる必要があります。コンポーネントが最初に実行される場合に、遅延の無いようにするには、プロジェクトがロードされる際にコンポーネントもロードされるように指定することです。

この設定は、2 つの場所で行うことができます。コンポーネントを作成する場合に、**即時起動**のフラグを ***.eci*** に指定することができます。この設定がデフォルト設定となります。

また、**コンポーネント**リポジトリ内で、**コンポーネント特性**（***Alt+Enter***）を利用することでこのデフォルト設定を変更することができます。

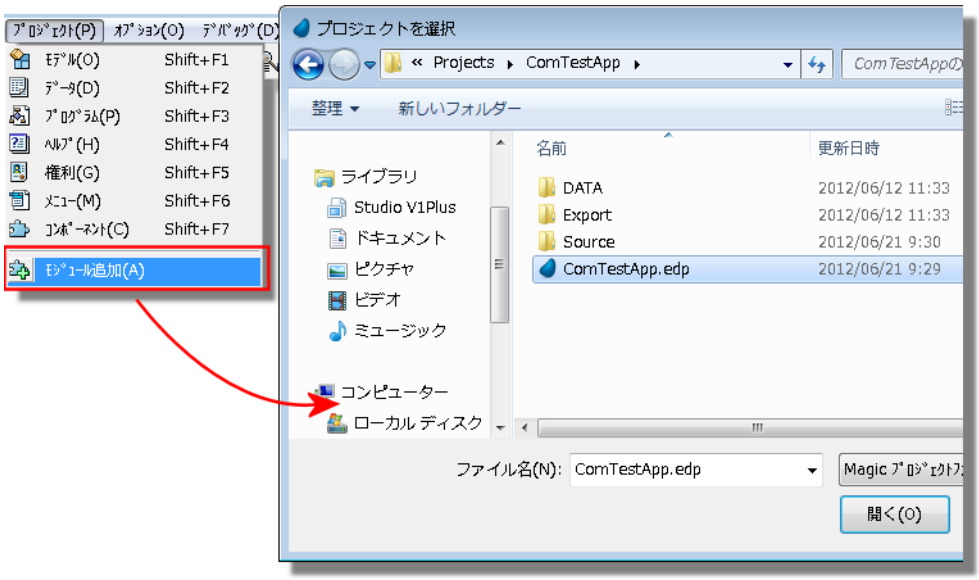
コンポーネント用プロジェクトを一括管理するには

１つのアプリケーションを開発する場合、ホストアプリケーション用のプロジェクトと複数のコンポーネント用プロジェクトを使用する場合があります。その際、関連するプロジェクトファイルを一括で管理できたほうが効率的に開発できます。

Magic xpa Studio では、コンポーネント用のプロジェクトをホスト側のプロジェクトのモジュールとして登録することで、プロジェクトを簡単に切り替えることができるようになります。

モジュールとしてプロジェクトを登録する

1. ホストアプリケーション用のプロジェクトを開いた状態で、**オブジェクトを選択**ダイアログ（**プロジェクト→モジュール追加**）を開きます。
2. コンポーネント用のプロジェクトファイルを選択します。
3. **ナビゲータペイン**にモジュール用ツリーが表示され、選択されたプロジェクトがツリーに追加されます。

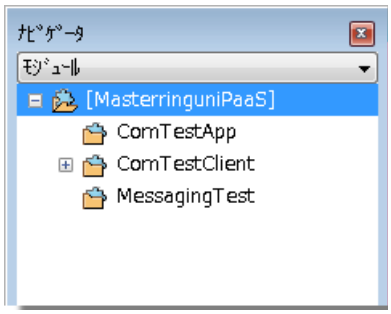


注： モジュールのプロジェクト情報がホスト側のプロジェクト（edp）ファイルに登録されるだけで、実際にプロジェクト関係のファイルがホスト側に追加されるわけではありません。

登録されるプロジェクトのパスは、ホストアプリケーションを基準に設定されます。他の PC にコピーする場合は、同じ位置関係になるようにする必要があります。

登録したモジュールを開く

1. **ナビゲータペイン**で***モジュール***を選択し、**モジュールツリー**を開きます。
2. オープンしたいモジュール名をダブルクリックします。
3. 指定されたプロジェクトが開きます。



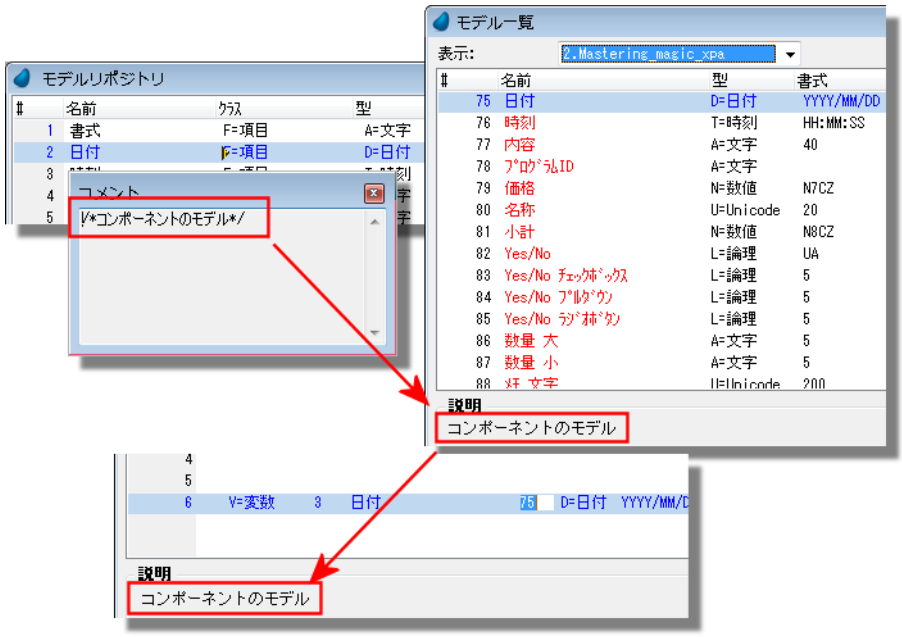
登録したモジュールを削除する

1. ナビゲータペインでモジュールを選択し、**モジュールツリー**を開きます。
2. オープンしたいモジュール名をクリックして選択します。
3. ***F3***（**コンテキストメニュー→行削除**）を押下すると削除確認のダイアログが表示されます。**はい**を**クリック**すると削除されます。

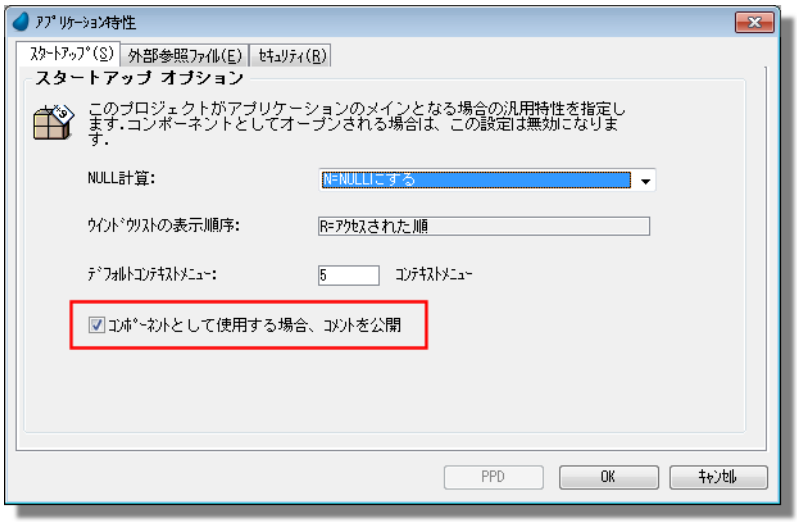
注： ホスト側のプロジェクト（edp）ファイル内のモジュール情報が削除されるだけで、実際にプロジェクト関係のファイルが削除されるわけではありません。コンポーネントとしている場合にも影響しません。

コンポーネントのオブジェクトのコメントを表示するには

オブジェクトにコメントを定義すると、そのオブジェクトの参照先でも定義されたコメントを表示させることができます。デフォルトでは、「/* ～ */」という書式でコメントを定義すると「～」の部分が表示されます。



ただし、コンポーネントアプリケーションのオブジェクトはデフォルトでは表示されません。コンポーネントアプリケーションのアプリケーション特性のコンポーネントとして使用する場合、コメントを公開をチェックした後で作成したキャビネットファイルを使用する必要があります。



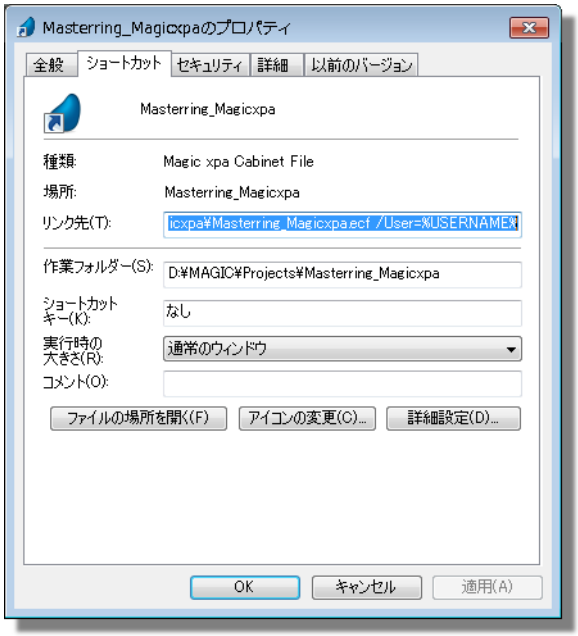
ヒント : *コメントにするための文字は、MAGIC.INI の［MAGIC_ENV］セクションで ShortDescriptionChar パラメータを設定することで変更できます。*

第 16 章 : 動作環境

Windows のログイン ID を使用して Magic xpa にログオンするには

Magic xpa には独自のログインダイアログがありますが、これを使用しないで、Windows のログイン ID を使用して自動的に Magic xpa にログオンすることができます。これは簡単で、ユーザの操作時間を短縮することができます。

ユーザが Windows にログインすると、システム変数 %USERNAME% にはユーザ ID が含まれます。この内容は、Magic xpa 起動時のショートカットに指定することで渡すことができます。



Magic xpa にログインするために Windows のユーザ ID を使用する

1. 通常の方法と同じように、Windows 上で Magic xpa を起動するショートカットを作成します。
   - **リンク先 :** ***.ecf*** **ファイルの名前とパス**
   - **作業フォルダ** : 作業フォルダとなるパスを入力します（通常は、***.ecf*** ファイルの存在する場所と同じパスを指定します）。
2. リンク先を指定した後、**/USER=%USERNAME%** を追加します。この例では、結果として以下のように指定されます。

   ```
   C:¥xpa_Projects¥Examples¥Examples.ecf /User=%USERNAME%
   ```

3. Windows のログイン ID に合わせるために、Magic xpa に同じユーザ ID を登録します。パスワードは空白にします。
4. **動作環境**ダイアログ（**オプション→設定→動作環境**）の**パスワードの入力**（**システム**タブ）選択し、***No*** に設定します。
5. **動作環境**ダイアログの**ログオン許可**（**システム**タブ）選択し、***No*** に設定します。

これで、ユーザは、ユーザ ID やパスワードを入力せずに自動的に Magic xpa にログオンすることができます。

ヒント : *Active Directory を利用して認証することで、Windows のログイン ID で Magic xpa にログオンすることもできます。詳細は、第 30 章 :「ActiveDirectory サーバを使用して認証させるには」（665 ページ）を参照してください。*

注意事項

ログオンダイアログが表示できる状態の場合、他のユーザによって簡単にログインできてしまうため、この機能を使用する場合は慎重に行う必要があります。これによって、例えば管理者用のデータを参照できてしまうような問題が発生するかもしれません。

このような問題を回避するには以下のような方法があります。

- （上記に説明されるような）アイコンを使用する場合を除いて、Magic xpa にログオンすることができないようにしてください。ユーザが管理者権限で PC にアクセスできないようにすることでこの方法が可能になります。

- ユーザ ID を指定する場合は、**パスワード入力**を *Yes* に設定してパスワードを入力しなければいけないように設定します。
- 重要なデータにアクセスするユーザに対してはパスワード入力を強制してください。この方法を使用した場合、管理者は通常のユーザとは異なるショートカットを作成し、パスワードを指定するようにします。

```
C:¥xpa_Projects¥Examples¥Examples.ECF /InputPassword=Y
```

Magic xpa Studio 起動時に自動的にプロジェクトをオープンするようにするには

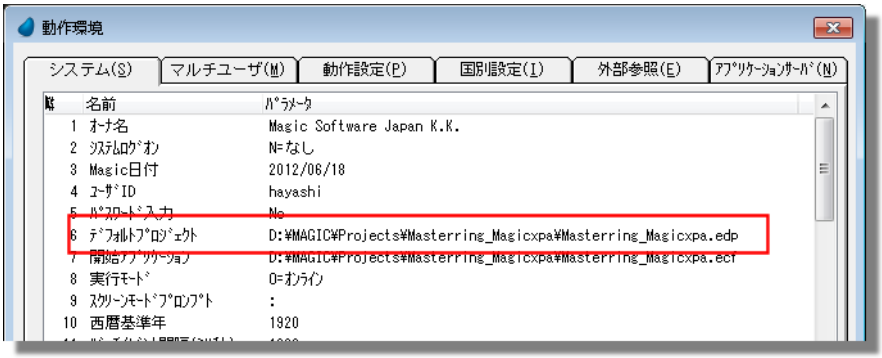
Magic xpa Studio を起動するには２つの基本的な方法があります。

- プロジェクト（***.edp***）ファイルやプロジェクトのショートカットを**クリック**して起動する。
- Windows のスタートメニューやデスクトップ上のショートカットを**クリック**して **Magic xpa Studio** を直接起動する。

直接 **Magic xpa Studio** を起動した場合、デフォルトではプロジェクトはオープンされません。しかし、**動作環境**ダイアログで**デフォルトプロジェクト**を設定することによって指定したプロジェクトをオープンさせることができます。ここには、プロジェクトファイル名とパス名を指定します。

注： この設定は、**開始アプリケーション**と同じようなものですが、実行アプリケーション（*.ecf* ファイル）の読込は行われません。

デフォルトプロジェクトを設定する



1. 上記で示されるように、**動作環境**ダイアログ（**オプション→設定→動作環境**）の**デフォルトプロジェクト**（**システム**タブ）でプロジェクトのパスとファイル名を入力することによってデフォルトプロジェクトを設定することができます。
2. **動作環境**ダイアログで指定する代わりに、Magic xpa のインストールディレクトリ内にある Magic.ini を編集することでも指定できます。この例では、以下のように指定します。

```
DefaultProject = C:¥xpa_Projects¥Examples¥Examples.edp
```

これで Magic xpa を再起動すると、指定されたプロジェクトがオープンされます。

Web 用のアプリケーションを開発する場合は、アプリケーションをテストするための Web サイトや特別なソフトウェアは必要ありません。Magic エンジンをサーバエンジンとして動作させることができます。以下に説明するような、作業が多少必要となります。Web 上でアプリケーションを実行させる方法についての基本的な知識も必要です。

Web アプリケーションをテストするには

Web 用のアプリケーションを開発した場合、アプリケーションをテストする際に Web サイトや専用のソフトウェアを用意する必要はありません。Magic は、サーバエンジンとして動作します。以下のような簡単な作業を行うことでテストを行うことができます。Web サービスをどのように動かすかを考える必要はあります。

サーバアプリケーションをテストする

必要条件： Web アプリケーションを実行させる前に、以下の準備が必要です。

1. アプリケーションを実行させる PC に IIS サービスがインストールされていることを確認します。
2. Magic xpa のインターネットリクエスタがインストールされることを確認します（IIS がインストールされていれば、Magic xpa のインストール時にデフォルトでインストールされます）。
3. **動作環境**ダイアログ（**オプション→設定→動作環境**）の**アプリケーションサーバを有効にする**（**アプリケーションサーバ**タブ）を ***Yes*** に設定します。設定内容を変更したら、プロジェクトを閉じて Magic xpa を再起動します。
4. MRB を起動します（**スタート→プログラム→ Magic xpa Studio →ブローカとリクエスタ→ MRB の起動**）。MRB をサービスとしてインストールした場合、Windows 起動時に自動的に MRB も起動されます。

これで、特定プログラムをテストしたい場合以下のような操作を実行します。

1. テストしたいプログラム上にカーソルを置きます。
2. **デバッグ→ブラウザから実行**（***Ctrl+Shift+F7***）を選択します。

Magic xpa は自動的にブラウザウィンドウを開き、そこでプログラムを実行します。特別なリンクを設定する必要はありません。それは自動的に作成されます。デバッグモードで実行していれば、**アクティビティモニタ**や**項目モニタ**などを使用することでアプリケーションの動作内容を確認することができます。

SDI アプリケーションとして実行するように指定するには

SDI（Single Document Interface ）アプリケーションは、個々のウィンドウが独自のメニューやタスクバー、ステータスバーを持っているものです。メインの MDI ウィンドウが背景に存在せず、SDI ウィンドウを閉じると、アプリケーションも終了します。

メインのコンテキストや**メインプログラム**を持っており、背景も存在していますがユーザには表示されません。

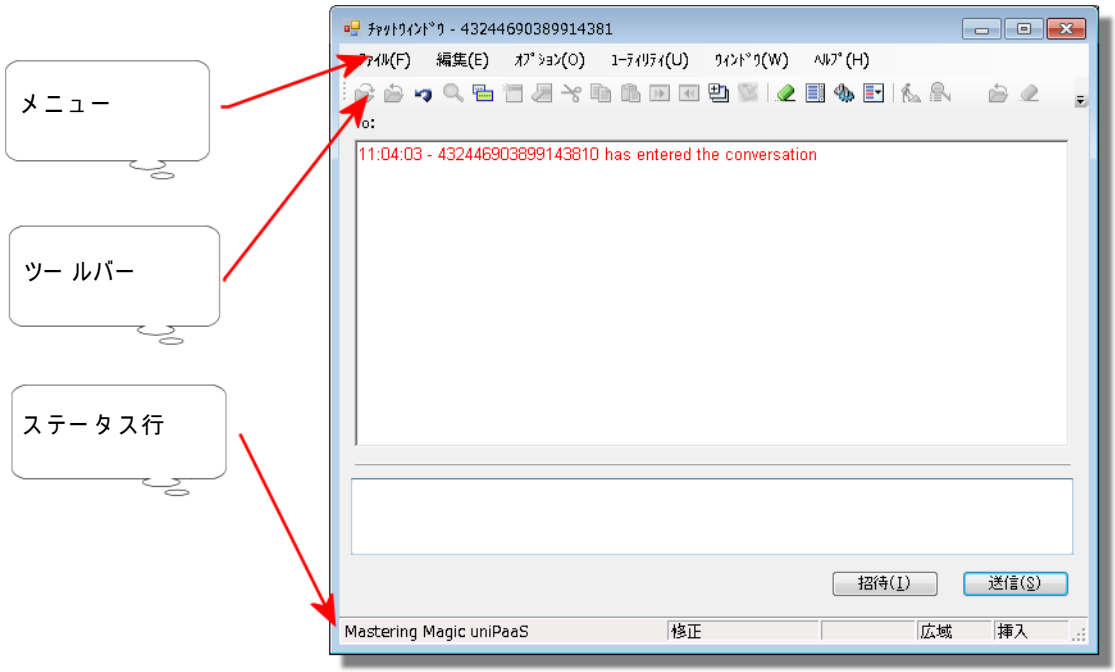
SDI アプリケーションを作成する

1. コンテキストがメインかどうかにもとづいて、**タスク前**ロジックユニットでプログラムを開始します。この設定でない場合、コンテキストが複数回読み込まれます。
2. **メインプログラム**の、**タスク特性**の**ウィンドウを開く**特性を *No* に設定します。

参照 : 「SDI プログラムとして実行させるには」（382 ページ）

SDI プログラムとして実行させるには

SDI プログラムは、独自の表示環境を備えたプログラムです。



このプログラムには、親（MDI）はありません。親はデスクトップになります。これは独自のコンテキストで実行します。

Magic xpa では簡単に SDI ウィンドウを作成することができます。以下の手順を実行します。

SDI コンテキストを定義する

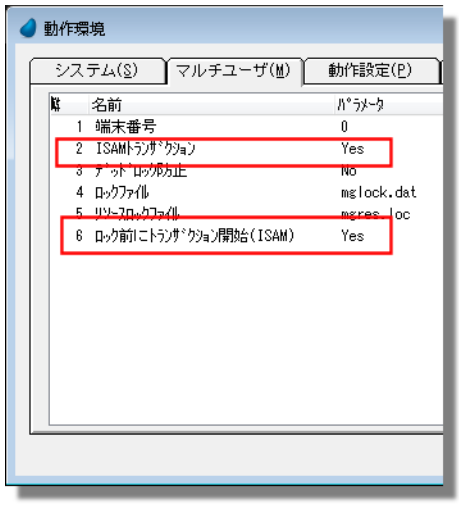
1. **フォーム特性**を開いて**ウィンドウタイプ**特性を ***S=SDI*** に設定します。
2. **タスク特性**を開き、**並行実行**特性（**拡張**タブ）をチェックします。
3. **メインプログラム**の**タスク特性**で**ウィンドウを開く**（**インタフェース**タブ）を ***No*** に設定されていることを確認します。このようにしないとエラーが発生します。

これで、SDI コンテキストの定義に必要な作業が終わりました。

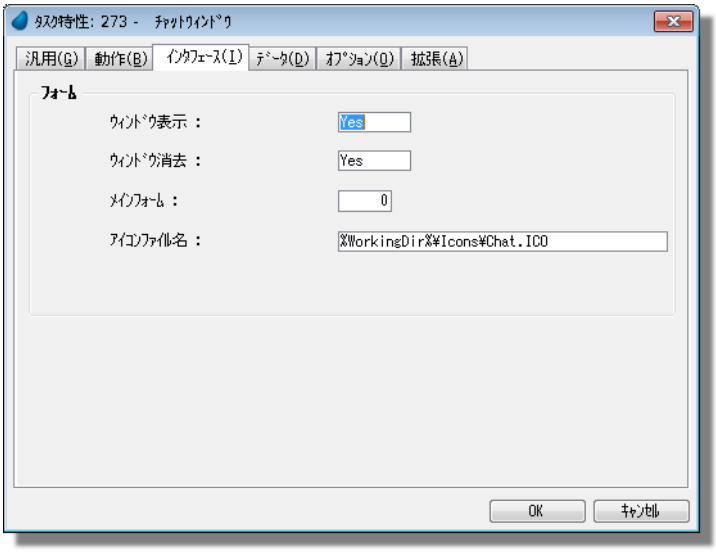
SDI コンテキストを修正する

1. **フォーム特性**：タスクを SDI と定義した場合、フォーム特性に **SDI** と呼ばれるセクションが表示されます。ここには以下の特性があります。
   - **プルダウンメニュー** と **メニュー表示** ：**メニュー**リポジトリのメニューから選択
   - **ツールバー表示**：表示するか否か
   - **ステータスバー表示**：表示するか否か
   - **開始モード**：デフォルト、最大、最小。最大が指定された場合、Magic xpa のフレームに制限されないためディスクトップ全体に表示されます。

   フォーム特性の残りの大半は有効で、他のフォームの場合と同じように動作します。**開始時の位置**特性の ***M=MDI の中央***の設定が例外です。MDI が存在しないため、この設定は意味がなく無視されます。

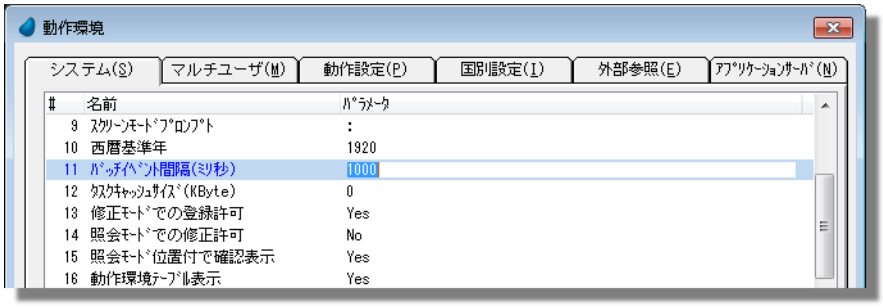


2. **タスク特性**で、フォームのアイコンを設定することができます。これはフォームの左上の隅で表示され、Windows で ***Alt+Tab*** を押下してウィンドウを切り替える際にも表示されます。ここに何も設定されない場合、**アプリケーション特性**で指定されたアイコンが表示されます。



参照 : 「SDI アプリケーションとして実行するように指定するには」（381 ページ）

バッチタスクでの画面の再表示間隔を指定するには



バッチタスクを作成する場合、ユーザに対して処理経過を示すプログレス画面を表示させることが一般的に行われています。これによって処理の状態を確認することができます。この画面は再表示させる必要があり、これによってレコードサイクルやプログレスバーの移動を表示させることができます。しかし、画面が頻繁に更新されると、バッチタスクのスピードに影響を与えることになります。理想的な再表示間隔は、実行している PC のパフォーマンスに依存します。新しくより速い PC は、古い PC より短い再表示間隔にしても処理することができます。

従って、**動作環境**ダイアログの**バッチイベント間隔**を使用することで、この機能を実行環境に合わせて設定することができます。バッチイベント間隔は 1 ミリ秒の単位で設定します。これは、1000 と設定した場合、画面が 1 秒間に 1 回の割合で再表示することを意味しています。

バッチイベント間隔を指定する

1. **動作環境**ダイアログ（**オプション→設定→動作環境**）の**バッチイベント間隔**（**システム**タブ）に移動します。使用したいミリ秒の値を入力します。
2. Magic.ini ファイルの **BatchPaintTime** のパラメータに値を設定することでも可能です。**INIPut ()** 関数を使用して一時的に **BatchPaintTime** 設定を変更することもできます。特定のバッチプログラムのみ間隔を変更し、それ以外では元に戻したい場合は、この方法を使用します。

エンジンが非同期イベントのチェックを行う間隔を指定するには

バッチタスクが実行している場合、Magic エンジンはユーザからの入力処理を待つことなく、システムが許す限り高速でレコードを処理します。

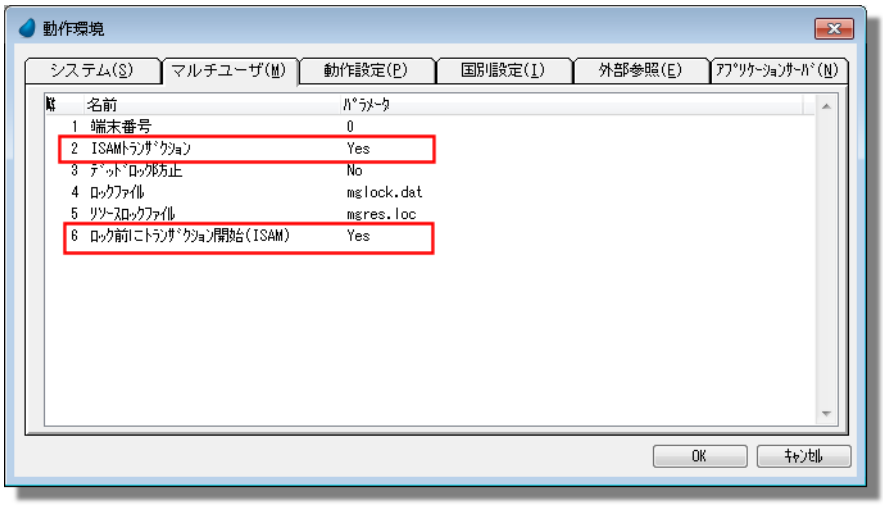
エンジンは未処理のイベントを受け取ることができ、それに対応した処理を実行させることができます。イベントのチェック間隔は、以下の３つの設定内容にもとづいて決定されます。

1. **イベント可**特性（**タスク特性→動作タブ**）：この設定が *Yes*、または **True** と評価される式が設定されている場合、エンジンはイベントをチェックします。これ以外はチェックされません。イベントには、キー操作も含まれます。このため、*No* が設定された場合、ユーザが **Esc** を押下してもバッチタスクをキャンセルすることができなくなります。
2. **バッチイベント間隔**（**オプション→動作環境→システムタブ**）：これはアプリケーション内のすべてのバッチタスクに影響しますが、必要に応じて、**INIPut**（）関数を使用することで実行中に変更することができます。
   - *0*：イベントのチェックは行われません。
   - *N*：N ミリ秒ごとにチェックされます。
3. **レコードイベント間隔**特性（**タスク特性→動作タブ**）：この設定は、現在のタスクでのみ有効で、エンジンに対し処理レコード数にもとづいたイベントのチェックを指示します。
   - *0*：レコード件数にもとづいたチェックは行われません。
   - *N*：N レコードごとにチェックされます。

これらの設定は、組み合わせて使用することができます。例えば以下のような設定が可能です。

- **イベント可**が *No* に設定した場合、他の設定内容に関わらずイベントのチェックは行われません。
- **バッチイベント間隔**を *0* に設定し、**レコードイベント間隔**を *0* に設定した場合、**イベント可**の指定内容に関わらずイベントはチェックされません。
- **バッチイベント間隔**を *300* に設定し、**レコードイベント間隔**を *20* に設定した場合、エンジンは 300 ミリ秒ごとおよび 20 レコードとにイベントをチェックします。

ISAM ファイルのトランザクションを組み込むには



ISAM 系の DBMS を使用している場合、Magic xpa は、SQL 系の DBMS と同じようにトランザクションをサポートします。**タスク特性**の**トランザクションモード**特性や**トランザクション開始**特性（**データ**タブ）を設定することで、ISAM 系 DBMS に適切なトランザクション要求が送られます。

しかし、アプリケーション全体に対して ISAM トランザクションの有効／無効を切り替えることもできます。つまり、いくつかのタスクでトランザクションを有効にすることができますが、アプリケーション全体で無効にすることで、タスクのトランザクション設定は無視されます。

ISAM トランザクションを設定する

1. グローバルに ISAM トランザクションを設定するには、**動作環境ダイアログ**（**オプション→設定→動作環境**）の **ISAM トランザクション**（**マルチユーザ**タブ）に移動して ***Yes*** を設定します。
2. Magic.ini ファイルに設定する場合は、以下のようにします。

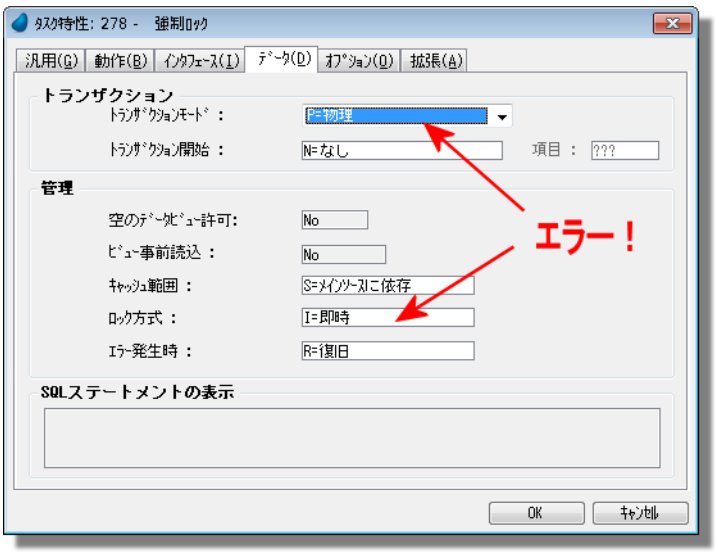
```
ISAMTransaction = Y
```

トランザクション内の ISAM の強制ロック

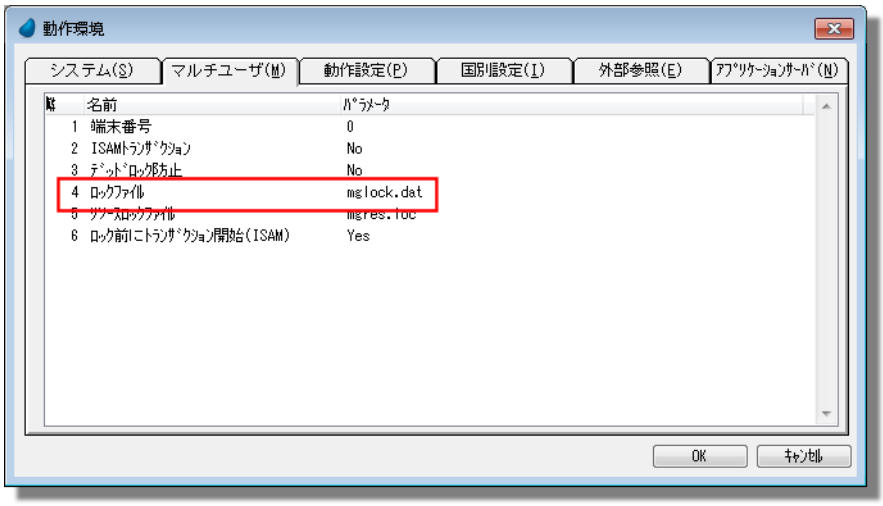
ISAM トランザクションが有効な場合、**トランザクション内の強制ロック**の設定を利用することができます。

もし**トランザクション内の強制ロック**が ***Yes*** に設定された場合、**ロック方式**特性が ***N= なし*** と設定されていなければ、**トランザクション開始**特性も ***N= なし*** に設定しなければなりません。すなわち、アクティブなトランザクションがない限り、レコードをロックすることができません。

ロック方式特性を設定した場合、トランザクションは開始されず、プログラムの**構文チェック**を実行するエラーが表示されます。



Magic xpa のロックファイルの位置を指定するには



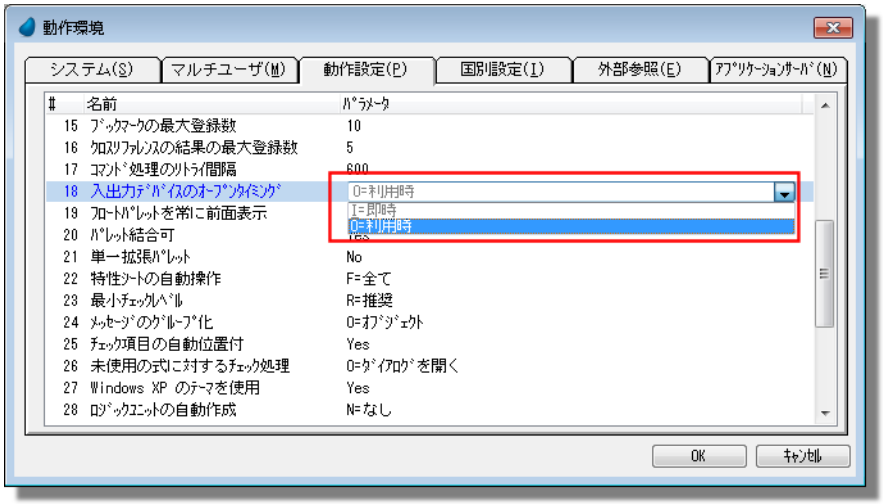
Magic xpa がレコードをロックする場合、ロックファイルを使用してロック状況を常に監視しています。

ロックファイルの名前は、Magic.ini で指定されます。ファイルを直接アクセスするか、**動作環境**ダイアログ（**オプション→設定→動作環境**）の**ロックファイル**（**マルチユーザ**タブ）で変更することができます。

ここにはロックファイルのパスは指定しません。ロックファイルは、ロックされているファイルのディレクトリに作成されます。その位置は、データベーステーブルの位置カラムや**データ**リポジトリ内のデータソース名カラムで指定されます。

参照： 第 17 章：「Magic xpa でデータベーステーブルを作成するには」（395 ページ）

入出力ファイルが実際に使用されるまで作成しないようにするには



入出力ファイルがオープンされるとすぐに、新しいファイルまたはプリントジョブが作成されます。しかし、タスクが実際に処理を開始する前にファイルがオープンされるため、何も出力していないタスクが、空の入出力ファイルを作成してしまうことを意味しています。これによって白紙が印刷されてしまうため、指定した帳票を出力したい場合に不都合が生じてしまいます。

このような問題を防止するには、**動作環境**ダイアログ（**オプション→設定→動作環境**）の**入出力デバイスのオープンタイミング**（**動作設定**タブ）を使用します。このパラメータを ***O= 利用時***に設定した場合、出力処理が実行されるまで、入出力ファイルはオープンされません。すなわち、**フォーム出力**処理コマンドが実行された時点でオープンされます。***I= 即時***に設定された場合、入出力ファイルは、定義されたタスクが起動された時点で作成されます。

ヒント : *帳票ヘッダの出力処理の定義場所によって帳票の出力処理を最適化することができます。これらの処理コマンドが**タスク前**にある場合、レコードが処理される前に、フォームが出力され、白紙の帳票を出力させる可能性があります。しかし、この処理を**グループ前**に定義したり、ページヘッダとして自動的に出力させるようにした場合、実際に出力するレコードが存在するまで出力されなくなります。*

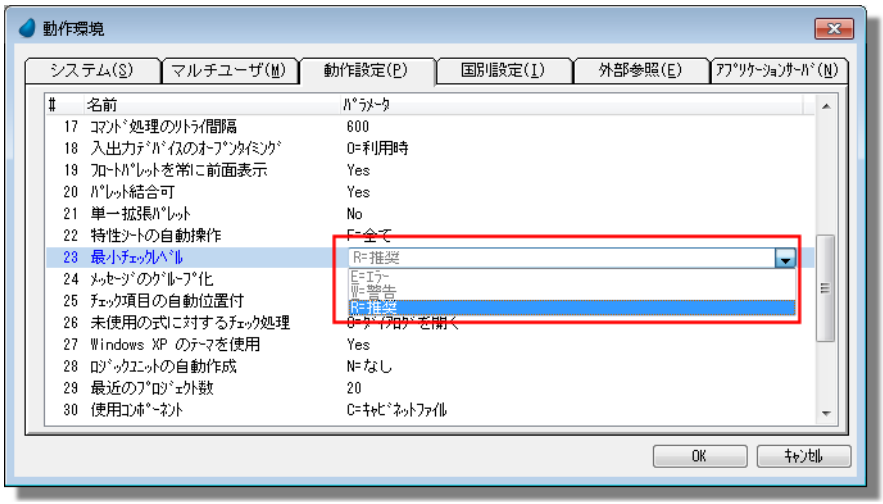
表示されたチェックメッセージを制御するには

Magic xpa Studio には**構文チェック**ユーティリティが備えられています。もしオブジェクトに問題がある場合は、このユーティリティを実行することで確認することができます。オブジェクト上にカーソルを置き、***F8***（**オプション→構文チェック**）を押下することでこのオブジェクトに対して**構文チェック**を行うことができます。***Alt+F8***（**オプション→カーソル以降をチェック**）を押下することで現在のカーソル位置移行の全てのオブジェクトをチェックすることもできます。

エラーが存在していれば、**チェック結果**ペインと呼ばれる個別のウィンドウが表示されます。表示されるメッセージの内容にもとづいて対応することができます。

最小チェックレベル

オプション→動作環境→動作設定→最小チェックレベル



この設定は、チェックメッセージ内で表示させたい内容のレベルを指定します。

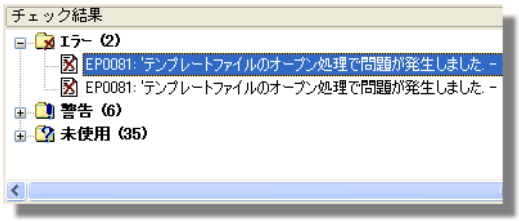
- ***R= 推奨***：推奨、警告、およびエラーを表示します。
- ***W= 警告***：警告とエラーを表示します。
- ***E= エラー***：エラーだけを表示します。

メッセージのグループ化

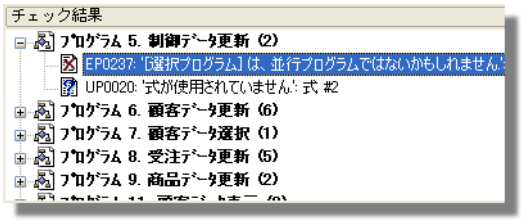
オプション→動作環境→動作設定→メッセージのグループ化

この設定は、チェックメッセージをどのようにグループ化させるかを指定します。

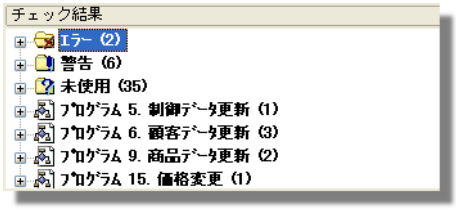
- ***T= タイプ***：リストの最上位に最も深刻な問 題を表示し、内容の深刻さの順にメッ セージを表示します。



- ***O= オブジェクト***：チェックされるオブジェ クトの順にメッセージを表示します。



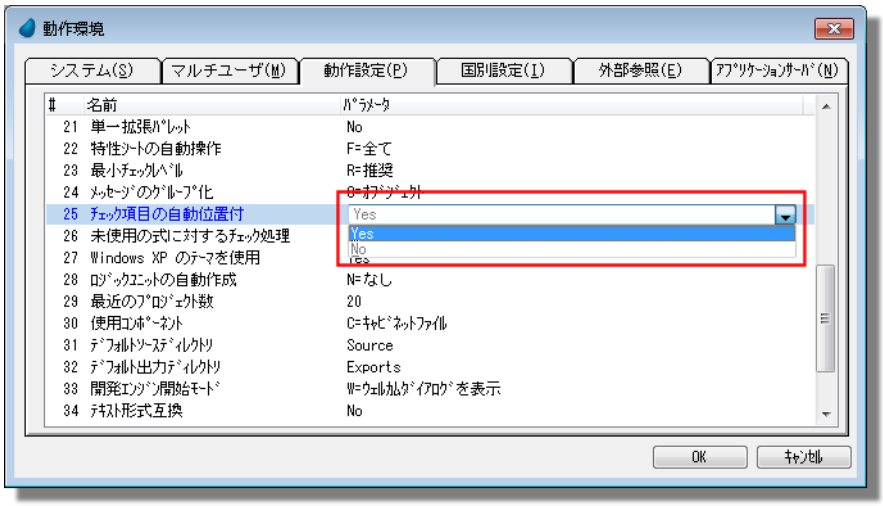
- ***B= オブジェクトとタイプ***: タイプとオブジェ クトの両方の内容をもとに順番に表示します。



これらのリストに表示されるメッセージ上で**ズーム**（***F5***、または**ダブルクリック**）することで、該当するオブジェクトにカーソルが移動します。これによってエラー原因を簡単に見つけることができます。

チェック項目の自動位置付

オプション→動作環境→動作設定→チェック項目の自動位置付



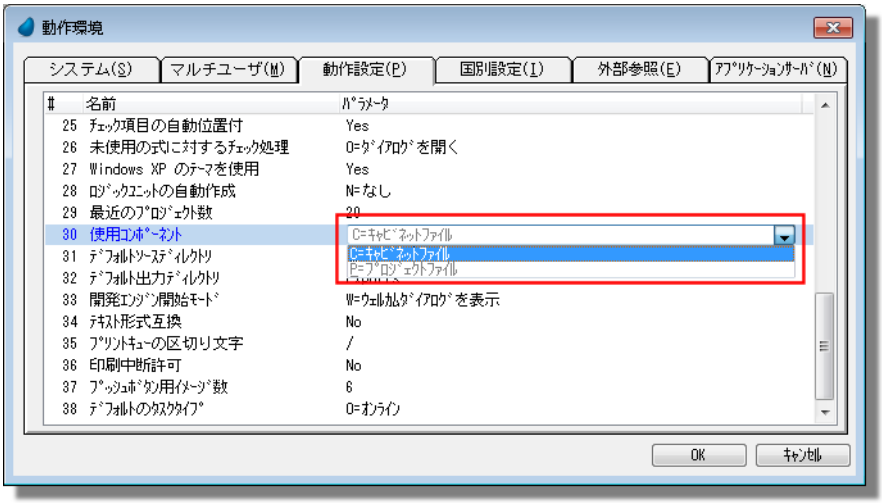
この設定を ***Yes*** にすると、**構文チェック**ユーティリティは、自動的にチェックリスト内の最初の項目に移動し、オブジェクトを開き、エラー箇所にカーソルを移動します。

チェックメッセージの表示

オプション→動作環境→システム→チェックメッセージの表示

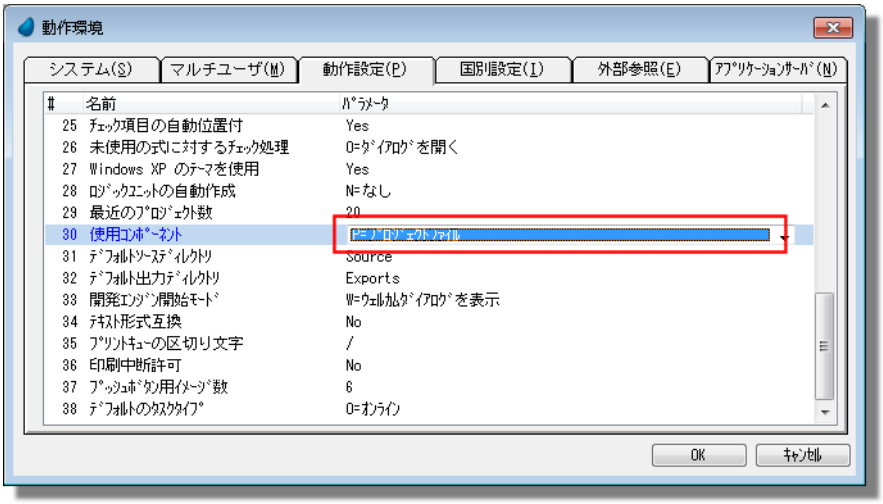
この設定を ***Yes*** にすると**チェックメッセージ**テーブルにアクセスできます。

プロジェクトファイルをコンポーネントとしてアプリケーションを開発するには

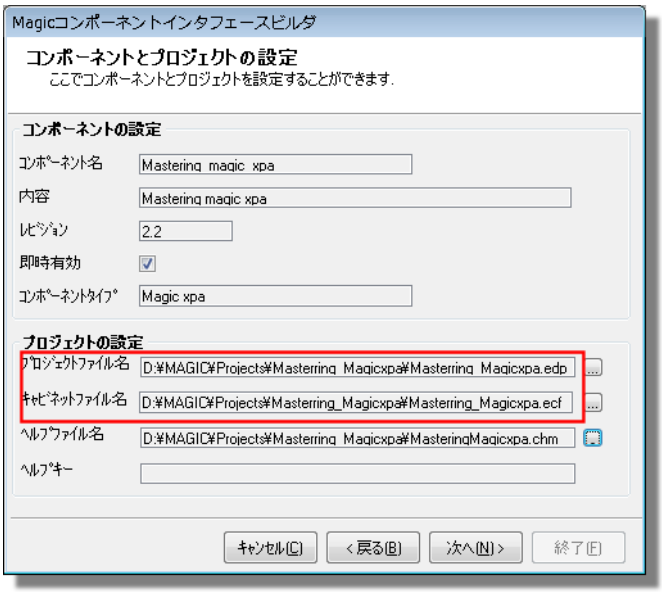


独自のコンポーネントを使用してプロジェクトを開発する場合、プロジェクト（***.edp***）ファイルまたはキャビネット（***.edf***）ファイルのどちらかをコンポーネントとして使用するように指定できます。コンポーネントを簡単に変更することができるため、プロジェクトファイルを使用した方が便利な場合があります。

プロジェクトファイルをコンポーネントとして使用する

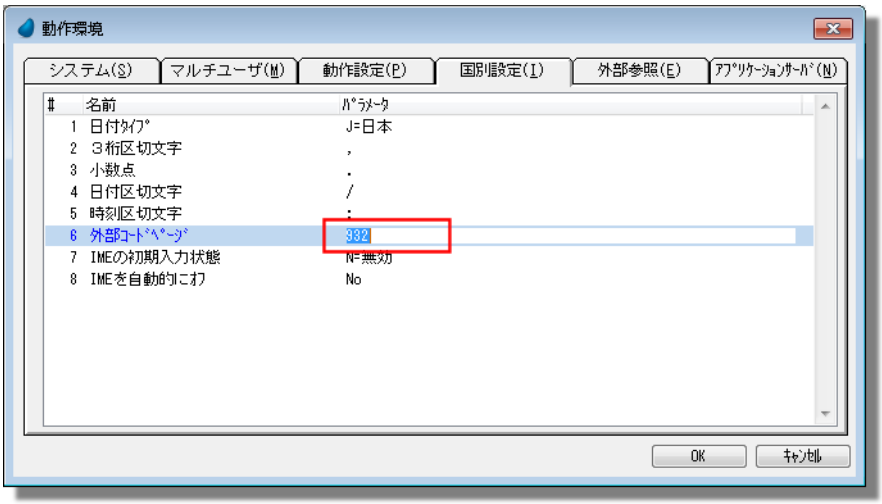


1. **動作環境**ダイアログ（**オプション→設定→動作環境**）の**使用コンポーネント**（**動作設定**タブ）を ***P= プロジェクト***に設定します。



2. この設定より、コンポーネントの作成時に指定されたプロジェクトファイルが開きます。

ANSI を Unicode に変換するためのコードページを指定するには



Unicode から ANSI（またはその逆も）へのコード変換は、コードページを使用して行われます。デフォルトでは、OS のコードページを使用します。使用するコードページは、**動作環境**ダイアログ（**オプション→設定→動作環境**）の**外部コードページ**（**国別設定**タブ）で設定することができます。

CodePage() 関数を使用することで、現在使用されているコードページを変更することができます。

［このページは意図的に空白にしています。］

第 17 章 : データソースの定義

Magic xpa でデータベーステーブルを作成するには

Magic xpa は、簡単にデータベーステーブルを作成することができます。必要な作業はテーブルのカラムを指定するだけで、Magic xpa は DBMS 内にテーブルを作成する処理を実行します。さらに、ISAM ファイルか SQL テーブル、またはメモリテーブルであるかどうかに関わらず、どのような種類のデータソースを作成する場合でも、操作内容は基本的に同じです。XML ファイルを定義する場合でも、基本的に同じ形式で行うことができます。

Magic xpa で SQL データソースを使用して作業する場合、基本的に以下の２つのケースが考えられます。

1. テーブルは存在せず、Magic xpa で新規作成する。
2. テーブルは既に存在しており、Magic xpa でアクセスできるようにする。

２番目のケースは、「既存のデータベーステーブルにアクセスするには」（400 ページ）で説明しています。ここでは、１番目のケースについて説明します。

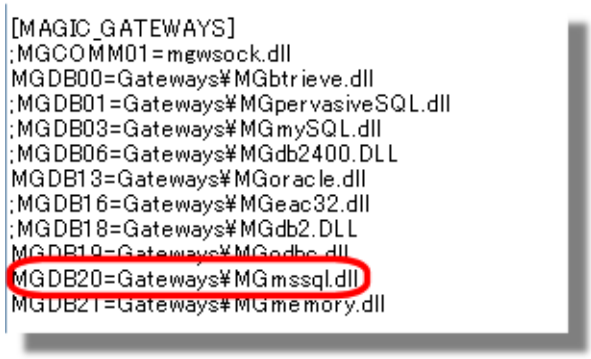
ここには、基本的な操作の概要が記述されています。以降に各詳細の説明があります。

- ゲートウェイと DBMS がロードされることを確認します。
- **データベース定義の設定**：DBMS に対応したゲートウェイをロードし、データベース定義を設定します。
- **テーブルの作成**：データソースリポジトリでテーブルを定義します。
- **カラムの作成**：テーブルに必要なカラムを定義します。
- **インデックスの作成**：テーブルに必要なインデックスを定義します。
- **テーブルの構文チェック**：テーブルの定義内容に問題がないかどうかをチェックするために、**構文チェック**ユーティリティを実行します。
- **レコードを作成してテーブルを確認する**：作成されたテーブルが動作することを確認するために、簡単な照会プログラム（***Ctrl+G***）を作成します。

データベース定義が正しく設定された場合、Magic xpa は、自動的に SQL テーブル定義を作成する処理を実行します。

注：　SQL テーブルを使用したこれらの処理は、メモリテーブルや ISAM ファイルでテーブルを作成する場合とほとんど同じになります。違いは、データベース定義が必要なことと、ISAM ファイルやメモリテーブルには命名時の制約がないということです。

1. ゲートウェイと DBMS がロードされることを確認する

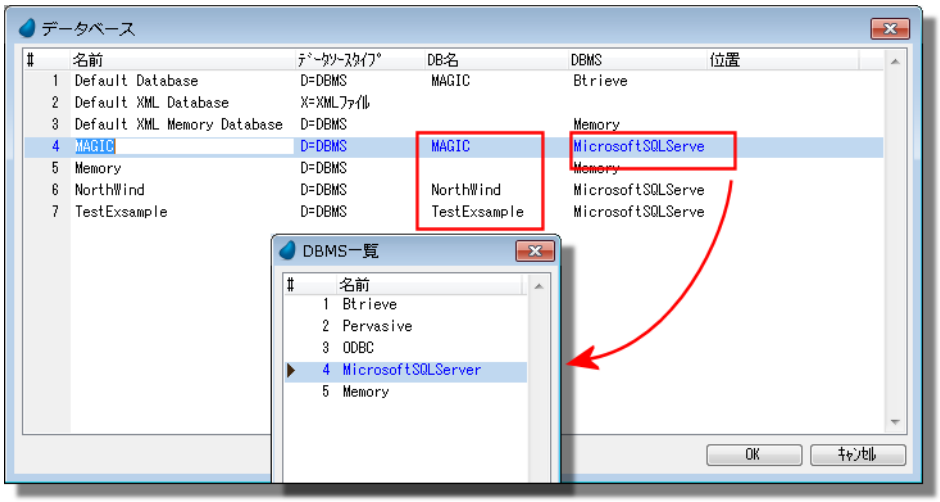


1. データベースを設定する前に、ゲートウェイがロードされていることを確認する必要があります。この設定は、Magic.ini で行います。Magic xpa のインストール処理の際に、選択されたゲートウェイの内容にもとづいて Magic.ini の MAGIC_GATEWAY セクションが自動的に設定されます。
2. 新しい DBMS タイプにアクセスする場合は、対応するゲートウェイがインストールされていることを確認するために Magic.ini とインストールディレクトリをチェックしてください。

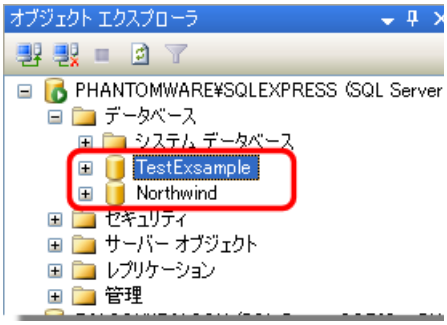
3. 当然ですが、使用する DBMS が PC にインストールされていなければなりません。この例では、Pervasive SQL の ISAM データベースと MS-SQL Server データベースにアクセスしているため、これらの製品が両方とも PC にインストールされていなければなりません。
4. ゲートウェイを追加する必要がある場合、次の手順に移る前に追加して、Magic xpa を再起動する必要があります。

2. データベース定義を設定する

データベース定義は、データソース指定がどのように使用されるか決める上での鍵となります。実際、１つのデータソース設定はいくつかのデータ定義を繰り返し使用することができ、これにより同じデータソースが、割り当てるデータ定義によってメモリテーブルとなったり、SQL テーブルや ISAM テーブルとなったりすることができます。



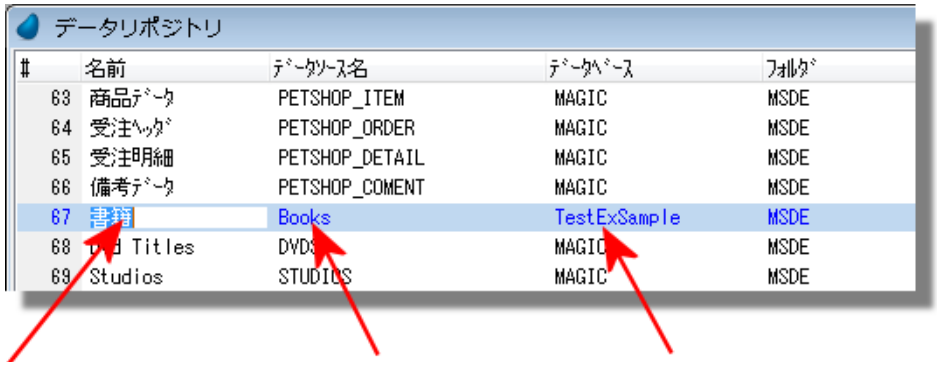
1. プロジェクトが開いていない状態で、**データベース**テーブル（**オプション→設定→データベース**）を開きます。利用可能な DBMS のみが **DBMS 一覧**に表示されます。ここから使用する DBMS を選択します。この例では、Microsoft SQL-Server と Oracle を使用しています。
2. 次に、データベース名を入力する必要があります。ここには DBMS 内のデータベース名を入力します。この例では、２つのデータベース（Northwind と TestExsample）を使用しています。
3. Magic xpa がデータベース内にテーブルを作成する前に、DBMS マネジャー（または、それに類するもの）を使用してデータベースを作成しておく必要があります。
4. 必要に応じて ***Alt+Enter***（**編集→特性**）を押下してユーザ ID とパスワードを設定します。また、Magic xpa でデータベース内にテーブルを作成する場合は、**データベース特性**の**開発でテーブル定義を変更**（**オプション**タブ）を ***Yes*** に設定する必要があります。



3. テーブルを作成する

次の手順は、実際のテーブル定義を作成することです。この作業を行う前に、使用する DBMS の制約条件を確認しておく必要があります。Magic xpa 自体はあまり制約条件がありません、このためメモリテーブルを使用する際に命名規約などの制約条件を考慮する必要がありません。しかし、SQL 系の DBMS では使用可能な文字に関する制約があります。

データリポジトリ（*Shift ＋ F2*）に移動し、一行追加（*F4*）します。以下のようにテーブルを作成します。



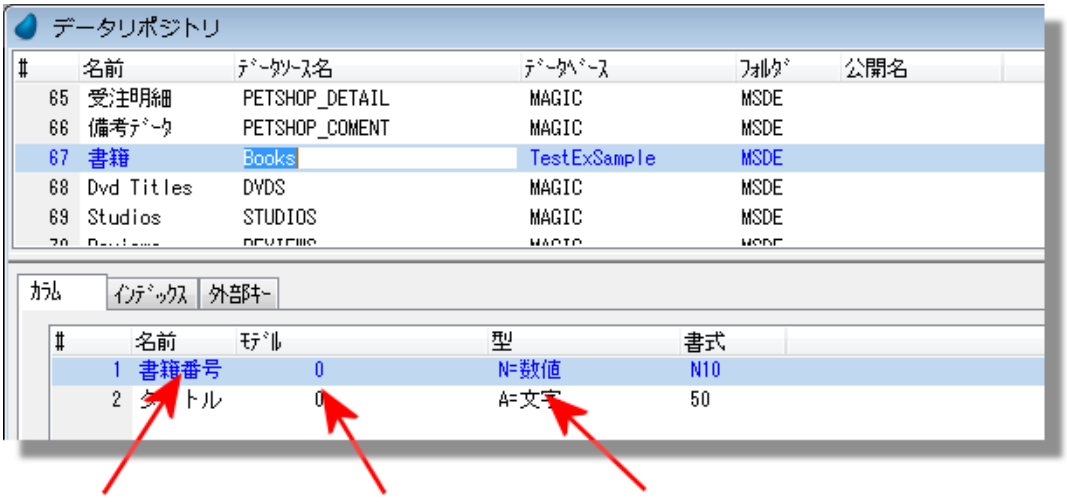
任意の**名前**を入力します。

データソース名のデフォルトは、名前カラムに入力した名前になります。この内容は変更できます。これは DBMS データベースの実際のテーブル名になります。

ここからズームして**データベース一覧**から**データベース**を選択します。

4. カラムを作成する

カラムを作成するには、以下に表示されているタブ領域を**クリック**します。各カラム毎に、*F4* を押下して行を追加し、以下のようにデータを入力します。各行には、そのカラムの基本的な特性が表示されています。より多くの特性値を設定する場合は、**特性**シート（*Alt+Enter*）を表示します。カラム定義の設定は、プログラム内で項目を定義する場合と同じような操作を行います。



各カラムの**名前**を入力します。入力された名前は、SQL テーブル内ではそのまま使用されませんが、なるべく同じ名前を指定する習慣にしてください。これは、SQL の規則に従うことを意味しています。

基本的な項目定義を行うようにしたい場合は、**モデル**を選択します。モデルの使用は、必須ではありませんが、保守作業が容易になります。

モデルを使用していない場合、カラムの**型**と**書式**を設定します。

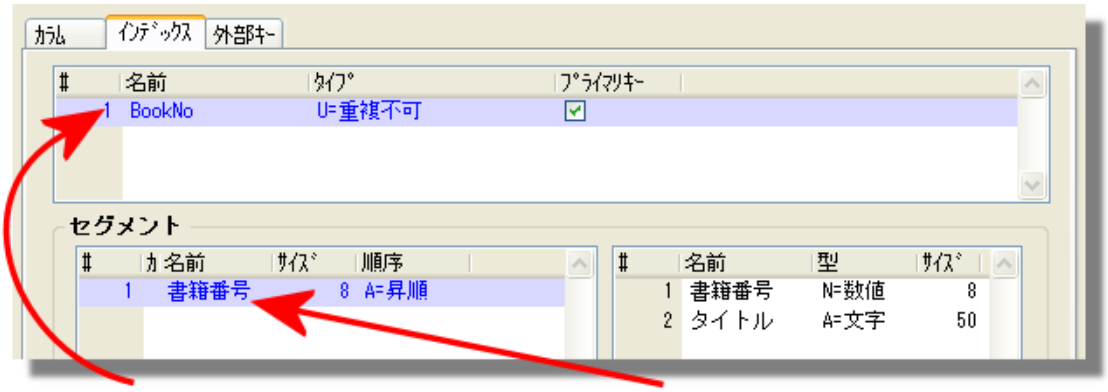
次に、**特性**シート（*Alt+Enter*）でこのカラムのより詳細な特性を設定する必要があります。ほとんどのカラムは、基本的な設定だけで使用できます。**項目モデル**に特性を設定することですべて終了する場合もあります。しかし、ここには、データを SQL データベースにどのように格納するかを正確に指定することのできるセクションがあります。

ここで重要な点は、データの各ビットに対する 2 つの内容があることです。Magic xpa のデータ型（文字型、数値型、日付型、時刻型など）は単純で一般的なものです。しかし、DBMS のデータ定義（ZStirng、LStiring、Integer、Float など）や実際の SQL 定義（Char、Integer）に対する設定もあります。Magic xpa は、デフォルトで使用頻度の高い定義内容を選択しますが、必要に応じて変更することができます。この設定内容の詳細は、「Magic xpa の項目とデータベースカラム間の割付を定義するには」（406 ページ）を参照してください。

ヒント：主要なデータ定義をカプセル化するためにモデルを使用することを推奨します。これによってSQL でのデフォルトを１カ所で定義できるようになります。

5. インデックスを作成する

次に、インデックスを作成する必要があります。画面下の領域のインデックスタブをクリックして作成します。



1. 各インデックスに対して、*F4* を押下して行を追加します。インデックスの**名前**を入力し、ユニークかどうかや、プライマリキーかどうかを指定します。**特性シート**（*Alt+Enter*）を開くことで詳細の特性を設定することができます。

2. 次に、各インデックスに対して、*F4* を押下してセグメントを作成します。ズームして右側に表示されている**カラム一覧**からセグメントの一部になるカラムを選択します。また、セグメントの並び順（昇順か降順）を指定します。ここから**特性シート**を表示して詳細な特性を設定することができます。

6. テーブルの構文チェックを実行する

カラムとインデックスの入力後、エラーがないかどうかを確認するために**構文チェック**ユーティリティ（*F8*、または**オプション→構文チェック**）を使用します。

1. チェックしたいテーブル上にカーソルを置きます。
2. *F8* を押下します。

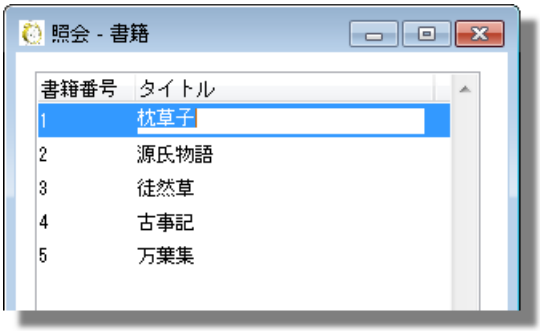
エラーがあれば**チェック**ペインに表示されます。

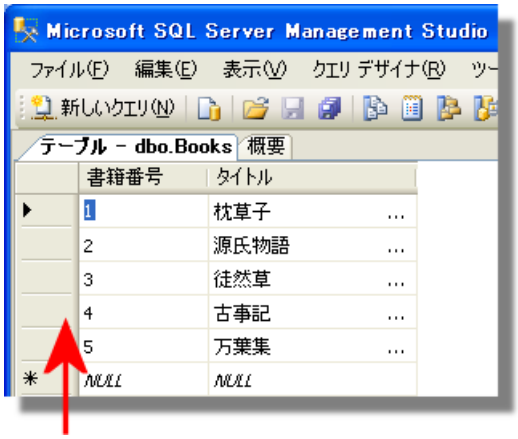
7. レコードを作成してテーブルを確認する

最後に、レコードが作成できることを確認します。簡単に確認する方法は、**APG**（Auto Program Generator）を実行することです。

1. チェックしたいテーブル上にカーソルを置きます。
2. *Ctrl+G*（**オプション→ APG**）を押下します。
3. **OK** をクリックします。

照会プログラムが正常に実行され、2、3 のレコードの追加ができれば、テーブルが正常に作成されたことをになります。DBMS ツールを使用することで、SQL DBMS に作成されたテーブルを参照することができます。





照会プログラムで作成されたテーブルが表示されます。 MS-SQL Server には同じテーブルが表示されます。

DBMS に作成されたテーブルが既にある場合、DBMS 内の定義内容と Magic xpa の定義内容を同期させる必要があります。一カ所（またはそれ以外）にある定義内容を常に維持することがとても良い方法です。Magic xpa で定義を変更した場合、**データベース特性**の**開発モードでテーブルを変更する**（オプションタブ）特性を *Yes* に設定することで、DBMS 内のテーブルの再構築が自動的に行われます。

参照： 「データベーステーブルを参照するには」（412 ページ）

既存のデータベーステーブルにアクセスするには

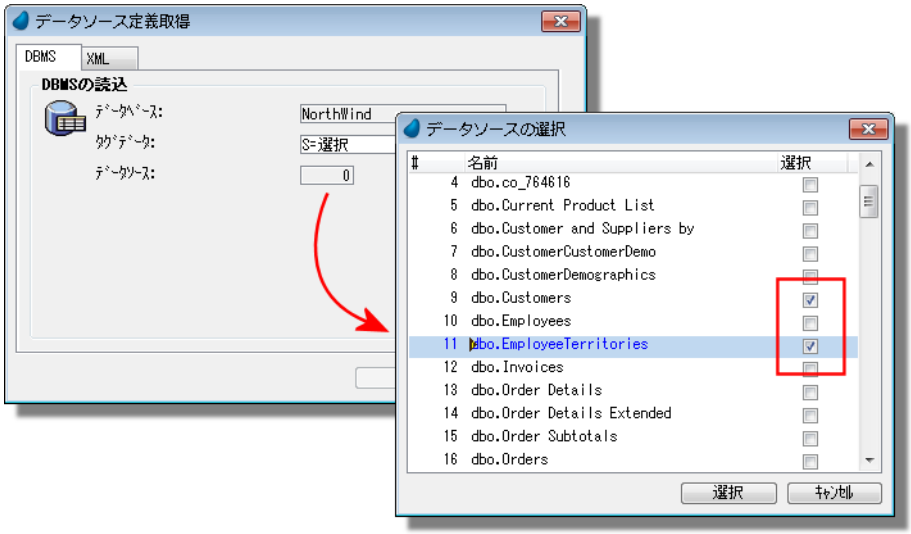
データベーステーブルがすでに DBMS 内に存在している場合、簡単に Magic アプリケーションでアクセスすることができます。

必要条件： 以下の環境設定が必要です。

- DBMS がインストールとされ、動作していること。
- インストールされたその DBMS に対応した Magic xpa のゲートウェイがインストールされていること。
- DBMS に対応したデータベース定義が行われていること。

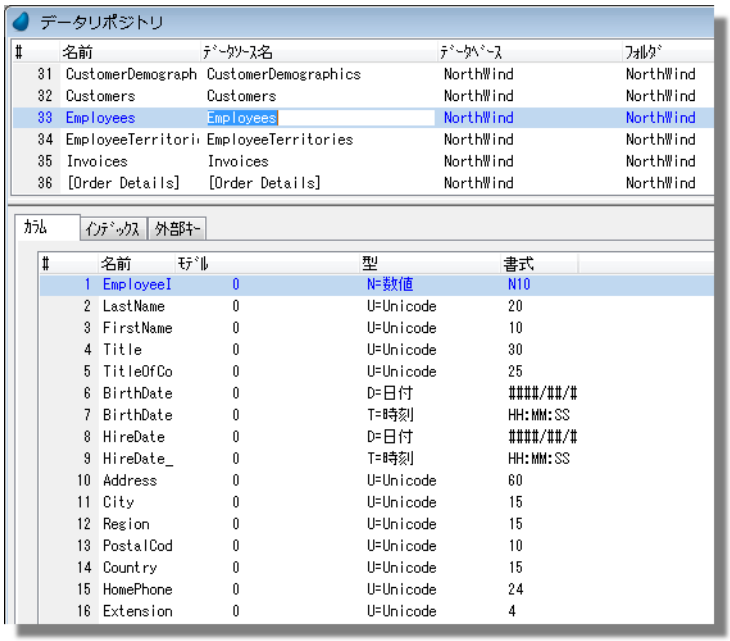
詳細は、「1. ゲートウェイと DBMS がロードされることを確認する」（395 ページ）と「2. データベース定義を設定する」（396 ページ）を参照してください。

既存のデータベーステーブルにアクセスする



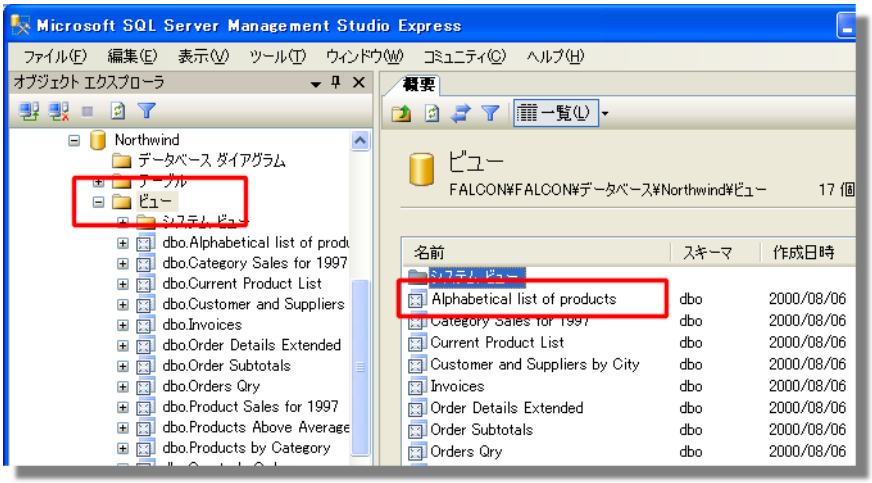
1. **オプション→定義取得**（*F9*）を選択します。**定義取得**ダイアログボックスが開きます。
2. **データベース**から**ズーム**して、データベースを選択します。
3. **タグテーブル**から、***S= 選択***または***A= 全部***のどちらかを選択します。
4. ***選択***を選択すると、**データソース**から**ズーム**してアクセスしたいデータソースを選択します。選択したいデータソースにカーソルを移動し、*Space* を押下すると選択状態になります。
5. データソースを選択したら、選択ボタンを**クリック**します。定義取得ダイアログに戻り、選択されたデータソースの数が表示されます。
6. **OK** を**クリック**します。ダイアログが閉じ、**データ**リポジトリに選択されたデータソースが追加されます。

これで、Magic アプリケーションにデータソース定義が追加されました。

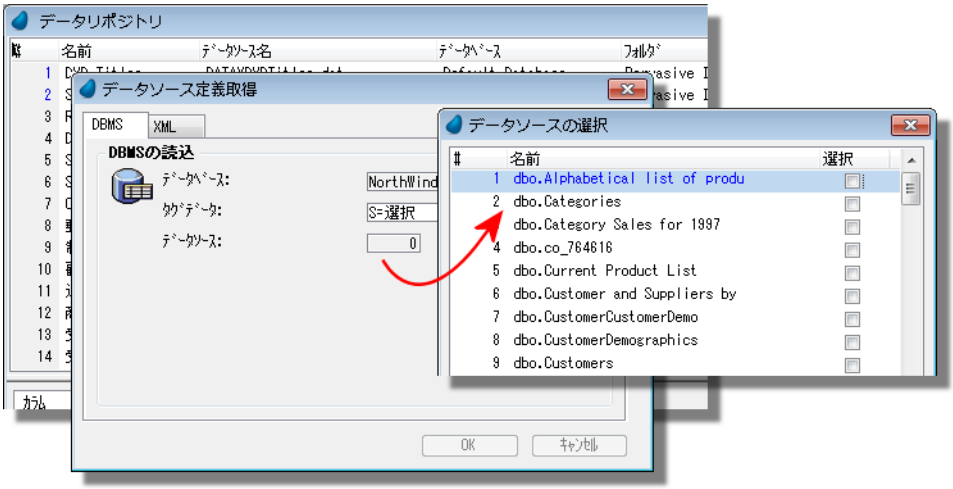


注： テーブルに日付と時刻型のカラムが存在する場合、「SQL の Date-Time のカラムを Magic xpa の日付と時刻の項目の組み合わせに変換しますか？」というメッセージが表示されます。Magic xpa での日付と時刻の組み合わせの処理に関する詳細説明は、第 11 章 :「日付／時刻データを加算するには」（250 ページ）や第 11 章 :「２つの日付／時刻データの差分を計算するには」（251 ページ）を参照してください。

データベースのビューからデータを取得するには



データベースの中には、データベーステーブル（表）とデータベースビューがあります。ビューは一種の仮想テーブルです。ビューにはデータは格納されません。むしろ、いくつかの標準にもとづいて 1 つ以上のテーブルからデータを抽出する SELECT ステートメントが格納されています。



しかし、データベーステーブルを扱うように、Magic xpa はデータベースビューを扱います。定義取得を実行すると、データベースビューがテーブルと一緒に一覧表示されます。データベースビューを選択すると、他の SQL テーブルのように動作する Magic xpa のテーブルを作成することができます。データベースビューの名前はブラケットで囲まれて表示されるため、それがデータベースビューかどうかを確認することができます。

定義取得の使用に関する詳細については、「既存のデータベーステーブルにアクセスするには」（400 ページ）を参照してください。

データベーステーブル定義を変更するには

Magic xpa で定義されたデータベーステーブルがある場合、必要に応じてこれを変更することができます。どのように変更するかは、テーブルが別のアプリケーションによって使用されているかどうかや、開発環境で利用できるデータであるかどうかに依存します。ここでは、以下のケースについて説明します。

- **Magic xpa でのみ使用**：データは Magic xpa でのみ定義されます。
- **外部アプリケーションと共有**：データは DBMS 内やサードパーティ製品によって定義されます。例えば、経理用のアプリケーションや購入したデータベースからデータにアクセスする場合などが考えられます。

Magic xpa 内でのみ定義されているデータベーステーブルを変更する

データベーステーブルが Magic xpa 内でのみ定義されている場合は、簡単に変更することができます。プログラムからの定義は、同期され可能な限り自動的に維持されます。

- カラムを追加すると、リポジトリ内のプログラムは自動的に更新されます。
- カラムを削除する場合は、どのプログラムもそのカラムにアクセスしていないことを確認する必要があります。使用されるかどうかを確認するには、カラム上で**クロスリファレンス**ユーティリティ（*Ctrl+F*）を使用します。使用していなければ、それを削除します。
- カラムを変更すると、参照情報も可能な限り変更されます。フォーム上に配置されている場合は、表示上の特性は変更されません。また、項目の型を変更した場合（例：数値型から文字型など）、データを更新するための処理コマンドの定義内容も変更する必要があります。

さて、既存のデータはどうなるでしょうか？**データベース特性**の**開発モードでのテーブル変換**特性を *Yes* に設定した場合、Magic xpa は自動的にデータの構成を変更し、万一に備えてバックアップコピーの作成を実行することもできます。

しかしこのような処理は、データを作成する運用環境で行おうとするものではありません。作成されたアプリケーションを提供する場合、既存データの書式を再定義するためのユーティリティを提供するべきです。簡単に行う方法として、**データ**リポジトリ内に新旧のテーブルを両方定義し、データを旧テーブルから新テーブルに移動するための Magic プログラムを作成することがあります。

外部テーブルと共有している Magic xpa のデータソースを変更する

さて、別の場所で定義されるデータソースを Magic xpa で利用する場合は、異なる問題が発生します。この場合、別のソースから定義を取得する必要があります。**開発モードでのテーブル変換**を *No* に設定しておく必要があります。この場合、Magic xpa は既存のデータの変更を行いません。

既存のデータソース定義上で変更内容を入力することで、手動で内容を変更することができます。

定義取得ユーティリティ（*F9*）を使用して、自動的に定義内容を取得することで変更内容を反映することもできます。ここには、定義取得を使用した変更方法についての説明があります。

1. テーブル定義を取得するために、**定義取得**ユーティリティを使用します。詳細は、「既存のデータベーステーブルにアクセスする」（400 ページ）を参照してください。
2. 定義内容を印刷するか、そのスクリーンショットを取ります。これによって次のステップで、内容を確認することができます。
3. 既存のテーブル定義を編集します。カラムが追加された場合、空白のカラムを既存の定義に追加します。カラムが削除された場合、定義から削除します。この操作は、カラムの位置や大まかな内容を外部テーブルに合わせるためです。例えば、顧客名が新しい定義では 5 番目の行にあり、20 文字の長さで、6 行目に 30 文字のカラムが追加された場合、Dummy というカラムを追加します。文字の長さは考慮する必要はありません。
4. インデックスに対しても同じようにします。
5. ２つの定義内容が位置的に整合性が取れていれば、古テーブル定義を新しい定義で上書きします。

Magic xpa は内部 ID を使用して位置でデータベースカラムを参照しているため、このような処理によって変更ができます。通常は、既存のテーブルが変更されることはあまりありません、また、項目を追加する処理には時間もかかりません。

テーブル内のデータを暗号化するには

パスワードを使用してアクセスを制限した場合でも、低レベルのツールを使用することで、ディスク内容を参照することができます。このため、非常に重要な内容が含まれているテーブルを暗号化する必要があります。

Magic xpa では簡単に暗号化することができます。以下の手順で行います。

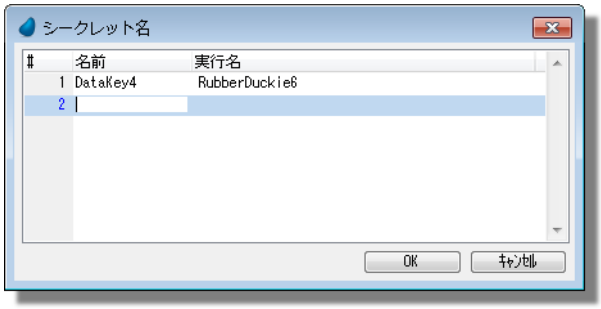
データベーステーブルを暗号化する

1. データリポジトリ内の暗号化したいテーブルに移動します。
2. データソース特性（*Alt+Enter*、または編集→特性）を開きます。
3. アクセスキー特性に、任意の文字列を入力します。シークレット名を使用することを推奨します（「シークレット名を使用するには」（405 ページ）を参照してください）。
4. テーブルの暗号化特性を *Yes* に設定します。

これで、テーブル内のデータは暗号化されます。同じキーを持っているユーザのみデータを参照することができます。

注： すべての DBMS がデータの暗号化をサポートしている訳ではありません。使用している DBMS がサポートしていない場合、アクセスキーとテーブルの暗号化の各特性は灰色で表示されます。

シークレット名を使用するには

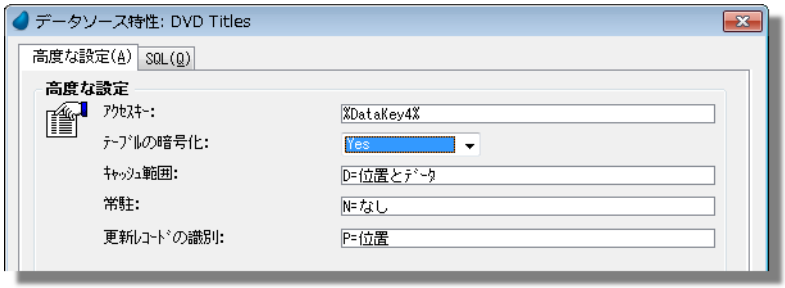


シークレット名を設定する

1. SUPERVISOR として Magic xpa にログオンします。
2. シークレット名テーブル（オプション→設定→シークレット名）を開きます。
3. シークレット名を入力します。**名前**カラムに入力された文字列は、スーパーバイザ以外ではアクセスできないことを除けば、論理名のように動作します。

シークレット名は、ユーザ ID やパスワードを暗号化して格納することができます。

シークレット名を使用する



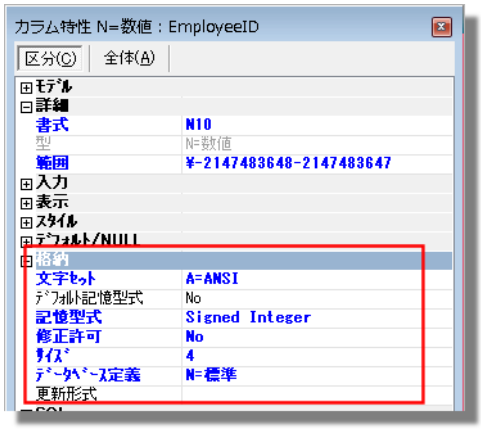
シークレット名を使用したい場合、論理名と同じ指定方法で入力します。この内容は実行時に変換されます。

このように設定されている場合、アプリケーションのインストール時にシークレット名の情報（セキュリティファイル）も一緒に提供する必要があります。シークレット名を使用せず実行名を指定することもできますが、実行環境と合っていない可能性が発生します。

Magic xpa の項目とデータベースカラム間の割付を定義するには

Magic xpa は、組み込まれた詳細内容へのアクセスを開発者に行わせないようにすることで、膨大な種類のデータタイプを容易に処理できるようにしています。Magic xpa を使用している場合、数種類のデータ型（文字型、日付型、時刻型、論理型など）を扱いさえすればよく、これらの項目が実際にメモリやデータベースにどのように格納されるかを意識する必要はありません。

しかし、実際のデータベース記憶型式を制御することができます。これらの詳細は、データリポジトリの各カラムのカラム特性（*Alt+Enter*）で指定できます。

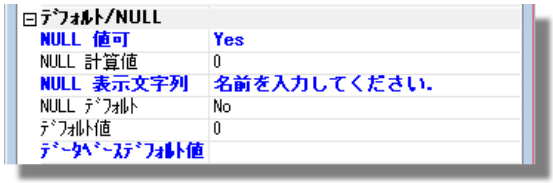


Magic xpa の項目のデータ型に関する詳細は、**詳細**セクションで指定されています。この内容は、10 桁の数値型項目を定義するとよくわかります。この場合、*2,147,483,648* から *-2,147,483,648* までの値を格納することができます。

格納セクションでは、この項目が格納される実際のデータ形式を指定することができます。この例では、4 バイトの *Siged Integer* で格納されます。

デフォルト /NULL と **SQL** のセクションは、項目に組み込まれた内容の詳細な情報を設定することができます。この内容については、以下に概要がありますが、この上にカーソルを置き、*F1* を押下することで各特性の情報が確認できます。

デフォルト /NULL セクション

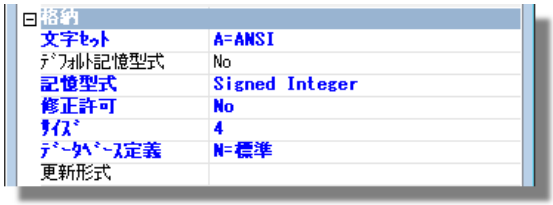


- **NULL 値可**：この特性が *No* と設定された場合、以下の 4 つの特性は灰色で表示され、この項目には反映されません。NULL は使用されなくなります。*Yes* に設定された場合、項目が最初に作成されると NULL（空白や 0 とは概念的に異なります）が設定されます。
- **NULL 計算値**：NULL が設定された場合に、実際に項目に含まれている値を指定します。この特性が指定されない場合、NULL の実際の値は Magic.ini ファイルの DBMS セクションで指定された値が設定されます。NULL 値は、通常は入力することのできない値です。しかし、プログラム内で **NULL()** や **ISNULL()** 関数を使用することができるので、実際の値を意識する必要はありません。
- **NULL 表示文字列**：NULL 値の場合にユーザに表示される値を指定します。例えば、NULL 値が 1901/01/01 などのような日付の場合に、「誕生日を入力してください。」というメッセージを表示させることができます。
- **NULL デフォルト**：この特性が *Yes* と設定された場合、項目のデフォルト値は NULL になります。*No* の場合、**デフォルト値**特性で指定された値がデフォルト値となります。
- **デフォルト値**：項目が最初に作成された場合に、Magic xpa が自動的に指定された値で項目を初期化します。手動で値を設定する必要はありません。ほとんどの項目は、*0* または*空白*がデフォルトとなります。しかし、特定の値で初期化したい場合は、ここに値を指定します。
- **データベースデフォルト値**：Magic xpa はここで指定された文字列を CREATE TABLE ステートメントを使用して DBMS に送ります。この設定によって、DBMS でデフォルト値を設定することができます。例えば、MS-SQL Server のテーブルに defvalue というデータベースデフォルト値を作成する場合、Magic xpa は以下のようにして作成します。

  ```
  CREATE TABLE owner1.table1 (Col1 CHAR(10) NOT NULL DEFAULT 'defvalue')
  ```

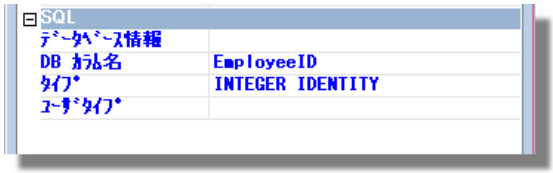
  Magic xpa は何も追加することなく、この文字列を CREATE TABLE ステートメントに追加します。

格納セクション



- **文字セット**： 格納時の文字セットを ***A=ANSI***、***O=OEM***、および ***U=Unicode*** から選択できます。
- **デフォルト記憶形式** : この特性を ***Yes*** に設定すると、DBMS によってデータ形式が決まり、**記憶形式**特性は無視されます。複数の DBMS に対して同じデータ定義を行いたい場合は、この特性を使用する方が便利です。
- **記憶形式**： ここには、実際のデータ型式が指定できます。ここからズームすると**記憶型式一覧**が表示され、選択することができます。SQL カラムに対しては、SQL セクションの**タイプ**特性でも指定することができます。
- **修正許可**： ユーザが実行時にこの項目にアクセスできるかどうかを指定します。
- **サイズ** : カラムに割り当てるバイト数を指定できます。
- **データベース定義** : ***N= 標準***、または ***S= 文字列***が指定できます。
- **更新形式** : この特性は、遅延トランザクションを使用する場合の SQL カラムにのみ有効です。差分と設定された場合、Magic xpa はメモリ内の値との差分にもとづいて値を更新します。複数のユーザが同時に同じ項目の操作をしているような場合（例えば、現在の株の総数を更新するような場合）、この特性が重要になります。

SQL セクション



- **データベース情報** : 使用する DBMS に渡すデータベース固有の情報を指定することができます。
- **DB カラム名** : DBMS 内で定義されるカラム名を指定します。
- **タイプ** : 使用するデータタイプを指定します。通常は指定する必要がありません。Magic xpa は、**記憶形式**特性で指定された値にもとづいたタイプを使用します。しかし、ここに直接指定することもできます。また、**定義取得**を使用して DBMS から定義をした場合、ここにデータのタイプが表示されます。
- **ユーザタイプ** : DBMS によっては、DBMS 内に独自の「ユーザタイプ」を定義することができます。ユーザタイプを定義した場合、ここでそのタイプを指定することができ、ユーザタイプは、テーブル作成時に Magic xpa によって使用されます。

参照 : 「データベーステーブル定義を変更するには」（403 ページ）

複数の DBMS で動作するように設定するには

Magic xpa でのハイレベルのデータ定義能力の一つは、複数のタイプのテーブルに対して同じデータソース定義を使用することができることです。例えば、MS-SQL Server テーブルによって動作し、さらに Oracle よって使用することのできる定義を持つことができます。

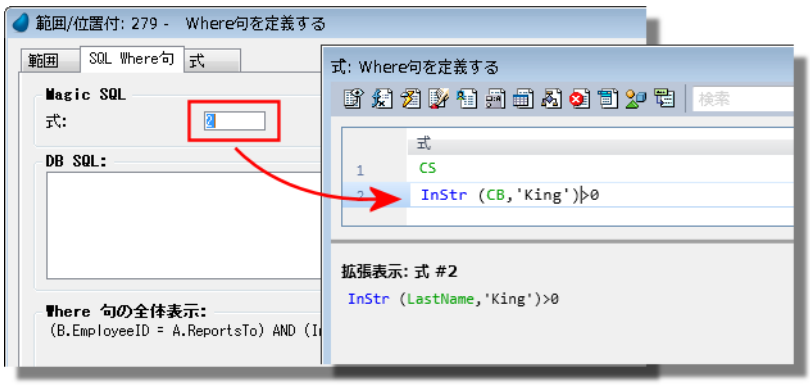
すべての DBMS は独自の処理を行うため、いくつかの制限があります。また、入力されるユーザ定義は各データベースに対して同じであることを知っておく必要があります。

しかし Magic xpa は、より高いレベルでデータを定義したり、実行時に DBMS 固有の機能を設定することを可能にするためのオプションを提供しています。

移行可能な項目を定義する

カラム特性の**デフォルト記憶形式**を ***Yes*** に設定すると、記憶形式特性は無視され、使用する DBMS 固有のデータ形式が使用されます。例えば、数値型項目がある場合、ある DBMS は ***Integer*** と定義し、別の DBMS では ***NUMBER*** や ***PACKED DECIMAL*** と定義されるかもしれません。

移行可能な WHERE 句を定義する



WHERE 句の構文は、DBMS によって異なる場合があります。例えば、特定の文字列から文字列を取得する場合、Oracle は **Substr** を使用し、MS-SQL Server は **Substring** を使用します。このため、両方で動作する WHERE 句を作成することができません。

この問題は、**範囲／位置付**ダイアログ（**タスク環境→範囲 / 位置付**）の **Magic SQL**（**SQL Where 句**タブ）に式を定義することで解決します。ここに定義する構文は Magic xpa の構文ですが、実行時に使用する DBMS 用の SQL WHERE コマンドが作成されるようになります。

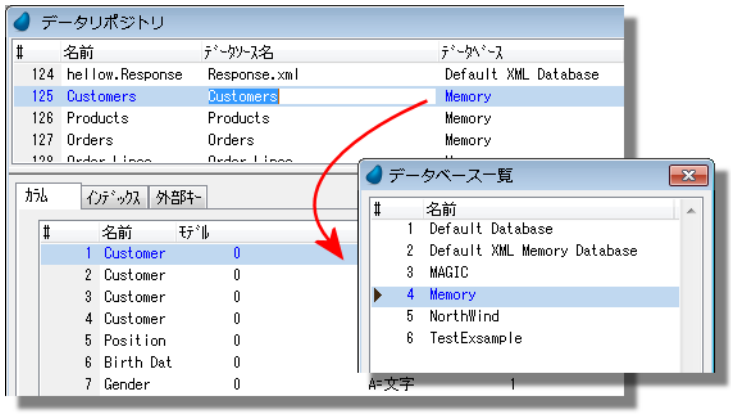
参照： 第 19 章：「タスクのデータビューに範囲を定義するには」（438 ページ）

永続性のないデータ用のテーブルを定義するには

Magic xpa には配列機能がありますが、一時的なデータを保持するための小さなテーブルを作成し、他のテーブルのように扱う方が一般的により簡単になります。この場合、項目モデルやコントロールモデルを使用することが可能で、**Blb2File()** と **File2Blb()** のような関数を使用することで素早くテーブル化したり、保存したりすることができます。テーブルを使用することは、インデックスを考慮する必要がなく、データの検索に範囲や位置付け機能が利用できることを意味しています。

Magic xpa で使用する永続性のないテーブルを***メモリテーブル***と呼んでいます。これは、他のテーブルと同じように作成することができます。

メモリテーブルを作成する



1. 他のデータベーステーブルと同じようにテーブルを作成します（詳細は、「Magic xpa でデータベーステーブルを作成するには」（395 ページ）を参照してください）。
2. **データベース**カラムから、***Memory*** を選択します。

注： もしメモリテーブルがデータベース一覧に表示されない場合は、Magic.ini またはデータベーステーブル（オプション→設定→データベース）に定義されていないためです。通常は、Magic xpa のインストール時に定義されます。メモリゲートウェイは、Magic.ini の [MAGIC_GATEWAYS] セクションの MGDB21 で定義されます。

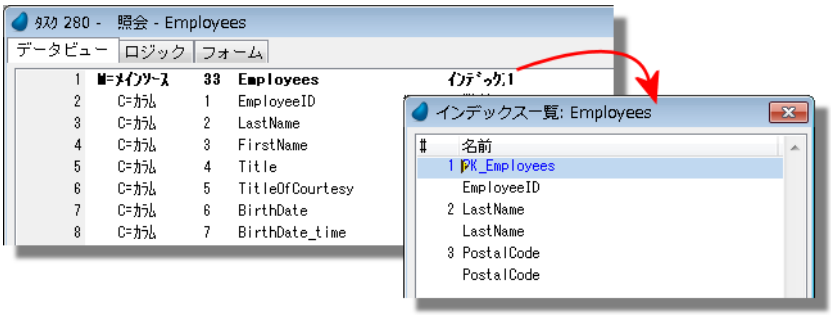
決められた順番でデータベーステーブルからレコードを取得するには

Magic xpa でデータベーステーブルを使用している場合、これらのレコードを表示する順番を指定することができます。これには３つの方法があります。

- **インデックスを指定する**：データソースを選択した場合、使用するインデックスも選択します。
- **インデックスの並び順を指定する**：昇順か降順のどちらかを指定します。
- **動的ソートを実行する**：実行時にソートされるレコードを指定することもできます。この場合、データソースにインデックスを定義する必要はありません。このオプションはデータソースに定義されたインデックスを使用する場合より処理が少し遅くなりますが、レコード数が非常に多いテーブルでない限り利用できます。

詳細の定義方法を以下で説明します。

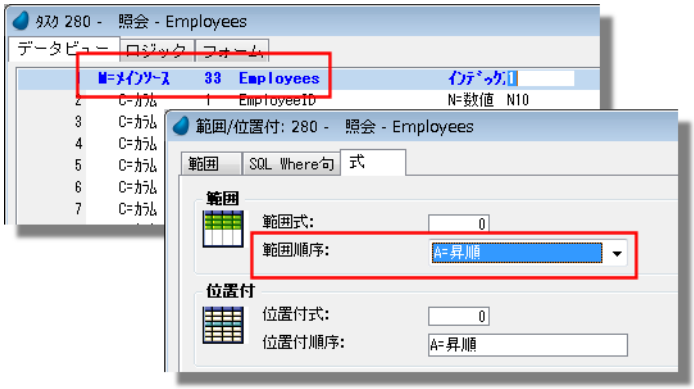
インデックスを指定する



プログラム内でデータソースを選択する場合、**インデックス**も設定することができます。ここから**ズーム**して使用するインデックス選択できます。この例では、#2 のインデックスを選択しています。この場合、従業員の姓をもとにレコードがソートとされます。

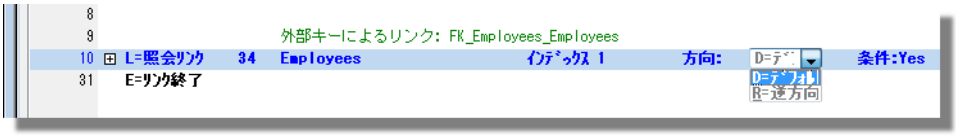
この設定は、メインソースとリンクテーブルの両方で有効です。インデックスを指定しない場合、レコードは作成順もしくは DBMS が決めた順番で表示されます。

メインソースに対する並び順を指定する



範囲／位置付ダイアログ（**タスク環境→範囲 / 位置付**）の**式**タブ（*Ctrl+R*）で**範囲順序**や、**位置付順序**を定義することでメインソースに対する並び順を指定することができます。**範囲順序**は、レコードが表示される順番を指定します。**位置付順序**は、位置付け指定されていたり、レコードの位置付けで位置付けカラムを使用している場合に使用されます。

リンクテーブルの並び順を指定する



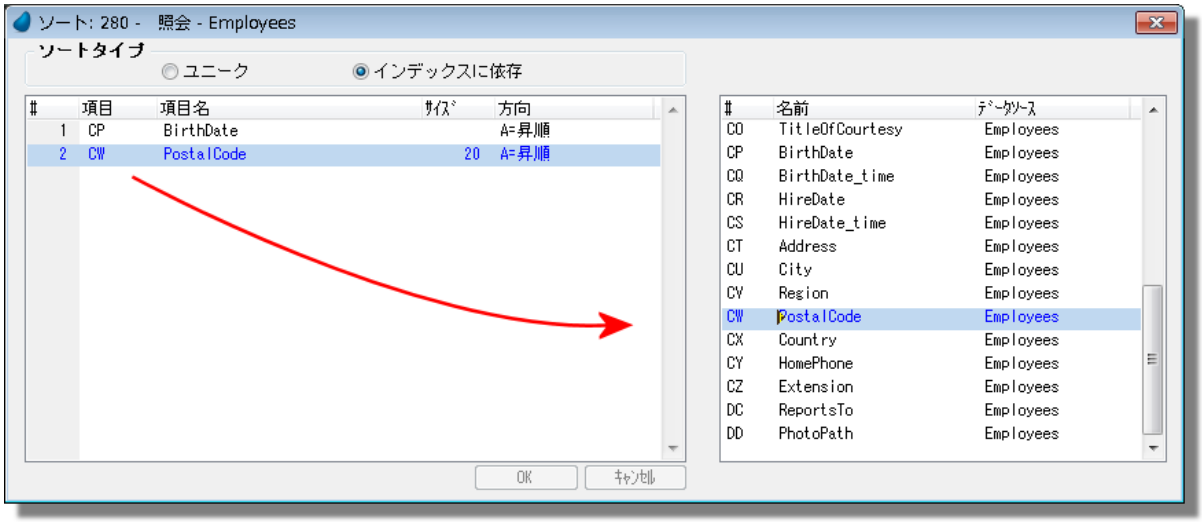
テーブルをリンクしている場合、1 つのレコードに位置付けているだけなので、方向の指定はそれ程重要ではありません。しかし、テーブル内の最初や最後のレコードを取得するような場合、指定内容はとても影響します。指定されたイ

ンデックス（例えば、郵便番号）の並び順が昇順に定義されている場合、**方向**特性を ***D= デフォルト***に設定した場合はテーブルの最小番号が返り、***R= 逆方向***を指定した場合は最大値が返ります。

定義されたインデックスを使用しないでレコードをソートするには

定義されたインデックスを使用しないでレコードをソートするには以下のようにします。

1. ソートテーブル（タスク→ソート、または *Ctrl+T*）を開きます。



2. ソートで使用する項目を以下の手順で定義します。
   - *F4* を押下して１行追加します。
   - **項目**カラムでズームして追加したい項目を選択します。
   - **方向**カラムで各項目に対して並び順（*A= 昇順*か *D= 降順*）を設定します。
3. **OK** をクリックして終了します。

データベーステーブルを参照するには

実際に格納されているデータの内容を参照することは、アプリケーションを開発する上で便利です。Magic xpa では、**APG**（**オプション→ APG**、または ***Ctrl+G***）を使用することで簡単に行うことができます。

参照プログラムを作成する

1. **データ**リポジトリで、参照したいテーブル上にカーソルを置きます。
2. ***Ctrl+G***（**オプション→ APG**）を押下します。
3. **APG** ダイアログから以下のパラメータを選択します。
   - **モード**：一時的にデータ内容を参照する場合は、***E= 実行***。**プログラム**リポジトリにプログラムを作成する場合は、***G= 作成***
   - **オプション**：***Q= 照会***
   - **カラム**から**ズーム**することで、表示するカラムを選択したり、表示する順番を変更することができます。
4. ***Enter*** を押下するか、OK ボタンを**クリック**します。

モードで***実行***を選択した場合、**データベース**テーブル内のデータビューが表示されます。デフォルトモードは照会です。データ内容を変更する場合は、***Ctrl+M***（**オプション→修正**）を押下します。

モードで***作成***を選択した場合、プログラムが**プログラム**リポジトリ内に追加されます。***F7*** を押下することでこのプログラムを実行することができます。テスト中に重要なデータを参照することができるように、プログラムを変更することもできます。

ヒント :APG を使用すると、簡単にプログラミング作業を開始することができます。APG でプログラムを作成しこれを修正することで、意図するプログラムを作成することができます。

データベーステーブルからテキストにデータを出力するには

データを表計算などのアプリケーション渡したり、データを変換したりするために、データベーステーブルのデータをテキストファイルに出力することがあります。Magic xpa は、このような処理を自動的に行うためのユーティリティとして **APG**（**オプション→ APG**、または ***Ctrl+G***）を提供しています。

テキスト出力プログラムを作成する

1. **データ**リポジトリで、出力したいテーブル上にカーソルを置きます。
2. ***Ctrl+G***（**オプション→ APG**）を押下します。
3. **APG** ダイアログから以下のパラメータを選択します。
   - **モード**：一時的にデータ内容を参照する場合は、***E= 実行***。**プログラム**リポジトリにプログラムを作成する場合は、***G= 作成***
   - **オプション**：***E= 出力***
   - **カラム**から**ズーム**することで、出力するカラムを選択したり、出力する順番を変更することができます。
4. **テキストファイル**には、出力するテキストファイル名を入力します。
5. ***Enter*** を押下するか、OK ボタンを**クリック**します。

モードで***実行***を選択した場合、指定されたデータがテキストファイルとして出力されます。

モードで***作成***を選択した場合、プログラムが**プログラム**リポジトリ内に追加されます。***F7*** を押下することでこのプログラムを実行することができます。

注： テキスト出力機能は、非常に便利ではありますが、いくつかの制限があります。特殊な問題として、CR/LF を含んだ文字型項目や BLOB 型項目が存在した場合があります。このようなデータは、XML 形式で出力するなどの別の方法が必要になります。

テキストファイルのデータをデータベーステーブルに入力するには

テキストファイルからデータを入力することは、便利な方法である場合があります。表計算アプリケーションや既に出力されたファイルからデータを入力する場合、この機能は便利です（「データベーステーブルからテキストにデータを出力するには」（413 ページ）を参照してください）。Magic xpa は、このような処理を自動的に行うためのユーティリティとして **APG**（**オプション→ APG**、または ***Ctrl+G***）を提供しています。

テキスト入力プログラムを作成する

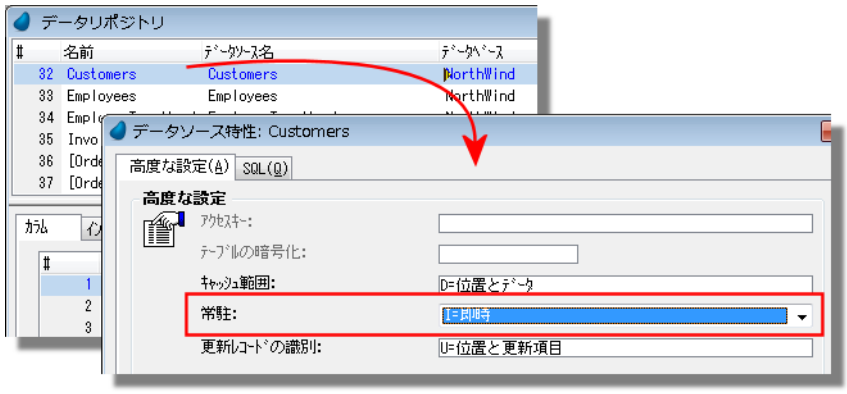
1. **データ**リポジトリで、入力したいテーブル上にカーソルを置きます。
2. ***Ctrl+G***（**オプション→ APG**）を押下します。
3. **APG** ダイアログから以下のパラメータを選択します。
   - **モード**：一時的にデータ内容を参照する場合は、***E= 実行***。**プログラム**リポジトリにプログラムを作成する場合は、***G= 作成***
   - **オプション**：***I= 入力***
   - **カラム**から**ズーム**することで、入力するカラムを選択したり、入力する順番を変更することができます。
4. **テキストファイル**には、入力するテキストファイル名を入力します。
5. ***Enter*** を押下するか、OK ボタンを**クリック**します。

モードで***実行***を選択した場合、指定されたテキストファイルからデータが入力されます。

モードで***作成***を選択した場合、プログラムが**プログラム**リポジトリ内に追加されます。***F7*** を押下することでこのプログラムを実行することができます。

注： テキスト入力機能は、非常に便利ではありますが、いくつかの制限があります。特殊な問題として、CR/LF を含んだ文字型項目や BLOB 型項目が存在した場合があります。このようなデータは、XML 形式として入力するなどの別の方法が必要になります。

データベーステーブルから一度にデータを取り込むには



プログラムで最も時間のかかる動作の１つには、データの読込処理があります。テーブルへのアクセスが頻繁に行われる場合、テーブルのデータを一度に読み込み、その内容を保持しておくことで、アプリケーションの処理が高速化されます。頻繁に更新されない読み込み専用のテーブルの場合、メモリ内にテーブル全体を読み込み、必要に応じてそれをアクセスすることは有効な処理になります。

しかし、手動でこのような処理を行う必要はありません。各データソースの**データソース特性**（*Alt+Enter*）には、**常駐**特性があります。この特性が *No* 以外の設定の場合、テーブルのデータはメモリに読み込まれアプリケーションが終了するまで常駐されます。このようなテーブルは、***常駐テーブル***と呼ばれています。

常駐テーブルは読み込み専用です。このテーブルは、一度データが読み込まれると頻繁に変わらないことを想定しています。しかし、アプリケーションの実行中に、内容を更新したい場合は、**DbReload()** 関数を使用することができます。

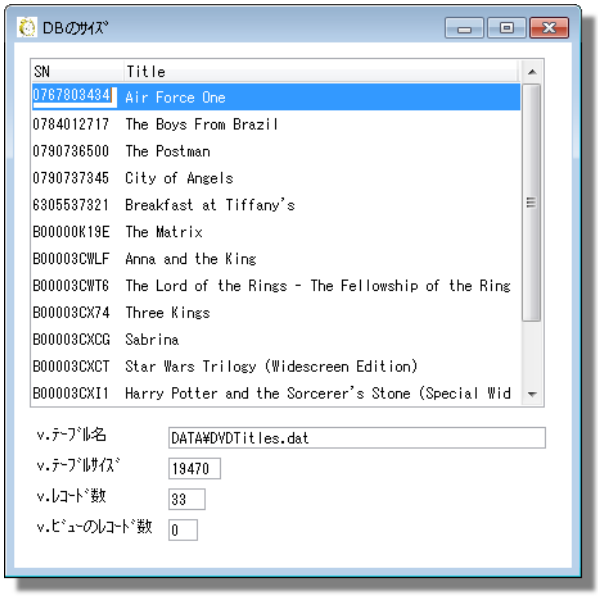
注：　この機能を利用するには、動作環境ダイアログ（設定→動作環境）の常駐テーブル読込（動作設定タブ）を *Yes* に選定する必要があります。

データがメモリ内に読み込まれる場合、**常駐**特性の設定オプションによって以下のような動作になります。

常駐特性の設定オプション

- ***N= なし***：デフォルトの設定です。テーブルは常駐テーブルにはなりません。使用しているタスクが終了するとテーブルもクローズされます。
- ***I= 即時***：アプリケーションが起動されるとこのテーブルは直ちに読み込まれ、アプリケーションが終了するまでオープンされます。
- ***O= 利用時***：プログラムが最初にこのテーブルをオープンした時点で読み込まれ、アプリケーションが終了するまでオープンされます。
- ***B= ブラウザ上で常駐***：テーブルはブラウザから読み込まれます。この設定により、テーブルのリンク定義による再計算がローカル上で行われます。

データベーステーブルからレコードを取得する際のデータサイズを指定するには



データベーステーブルのサイズを知る必要があるかもしれません。この場合、以下の３つの関数を使用することができます。

- **DbSize()**：テーブルのサイズをバイト数で返します。
- **DbRecs()**：テーブルのレコード数を返します。
- **DbViewSize()**：現在のデータビューのレコード数を返します。

以下に説明するように、これらの関数は多少異なった動作をします。

DbSize() 関数を使用する

バイト数でテーブルのサイズを求めたい場合、**DbSize()** 関数を使用します。構文は以下の通りです。

DbSize(*dsource#*, *tablespec*)

パラメータ：

- *dsource#*：データリポジトリでのテーブル番号です。
- *tablespec*：データリポジトリに定義されているテーブル名以外のテーブルにアクセスする場合は、ここにテーブル名を指定します。

戻り値：バイト数を表す数値が返ります。

例：DbSize('1'DSOURCE,'')

DbRecs() 関数を使用する

テーブルのレコード数を求めたい場合、**DbRecs()** 関数を使用します。構文は以下の通りです。

DbRecs(*dsource#*, *tablespec*)

パラメータ：

- *dsource#* ：データリポジトリでのテーブル番号です。
- *tablespec* ：データリポジトリに定義されているテーブル名以外のテーブルにアクセスする場合は、ここにテーブル名を指定します。

戻り値：レコード数を表す数値が返ります。

例：DbRecs('1'DSOURCE,'')

DbViewSize() 関数を使用する

現在のデータビュー内のレコード数を求める場合、DbViewSize() を使用します。構文は以下の通りです。

DbViewSize (*generation*)

パラメータ：

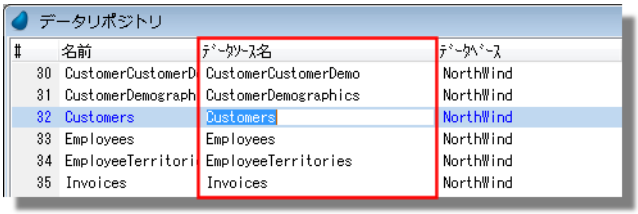
- *generation*：タスクの世代番号です。(0 は現在のタスク、1 は親タスクになります)

戻り値：レコード数を表す数値が返ります。テーブル内のレコード数ではなく、タスクで範囲指定をした場合にビューとして展開されたレコード数のみ返されます。

例： DbViewSize(0)

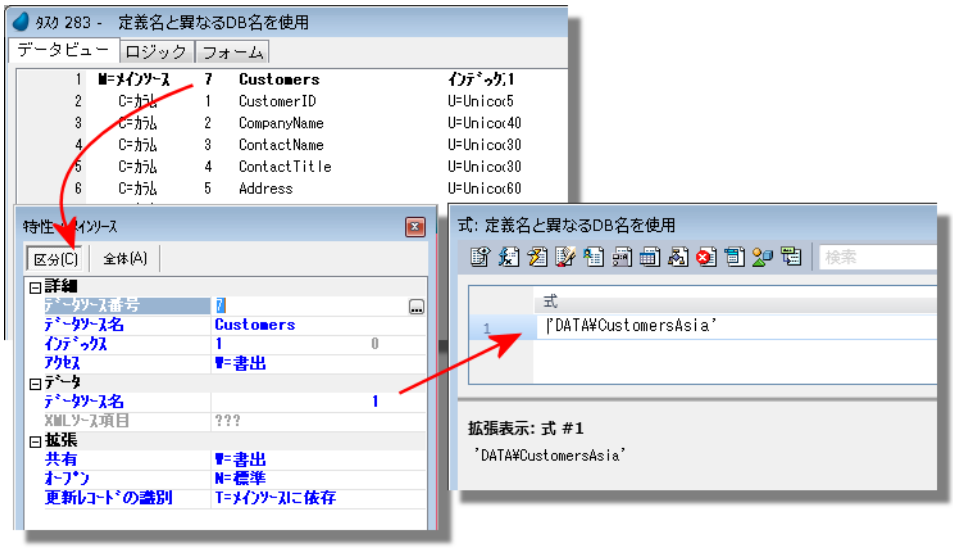
注： デフォルトでは、データビュー内のレコードは必要に応じて取り込まれたものだけのため、**DbViewSize()** は、範囲条件に合う実際のレコード数のみ返ります。この関数で正確なレコード数を取得したい場合は、タスク特性の**ビューの事前読込**特性を *Yes* に設定する必要があります。タスクが起動される前に、ビュー内のすべてのレコードが読み込まれます。これは **DbViewSize()** で正確な値を返すだけでなく、期待値どおりにスクロールバーを表示させるようにできます。

動的にデータソース名を設定するには



通常、データソース名はデータリポジトリの**データソース名**カラムに設定された名前が使用されます。しかし、データソースの定義の際や、そのデータソースにアクセスする関数にて指定することで、プログラムが実際にアクセスするデータソースを変更することができます。さらに論理名を使用することで、データソース名を実行環境にもとづいた設定ができます。以下で、これらのオプションについいて説明します。

定義された名前と異なるデータソース名を使用する



メインソースやリンクテーブルを定義する際に、データソース名を変更することができます。この設定を行うには、**項目特性**の**データソース名**特性で式を使用してデータソース名となる文字列を指定します。この例では、Cutomers というデータソースを CustomersAsia という名前ので使用するように設定しています。

関数でデータソース名を指定する

例えば、CustomersAsia テーブルのレコード数を取得する場合、以下のようなパラメータでは指定できません。

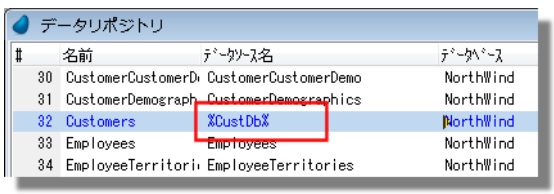
DbRecs(*'4'DSOURCE,''*)

以下のように指定する必要があります。

DbRecs(*'4'DSOURCE,'CustomersAsia'*)

２番目のパラメータを使用することで、実際にアクセスするデータソース名を変更しています。DB 関数のいくつかは、このように２番目のパラメータでデータソース名を指定することができます。

データソース名に論理名を使用する



データソース名をプログラムで指定する場合、直接名前を指定することは推奨できません。**論理名**を使用するようにしてください。この方が、保守がしやすくなります。例えば、CustomersAsia の代わりに、CustomersEurope や CustomersAustralia などの別のテーブルにアクセスすることも簡単にできるようになります。また、異なる保存データ用に名前を変更したテーブルを使用することもできます。

顧客データベース用に論理名 %CustDb% を定義し、データソース名を %CustDb% と定義した場合、**DbRecs()** 関数では以下のように指定します。

DbRecs(*'4'DSOURCE、%CustDb%*)

1 つのテーブルを定義することで、複数の実テーブルにアクセスするため、データリポジトリで論理名を使用することもできます。

タスクに定義されているデータソースを一元管理するには

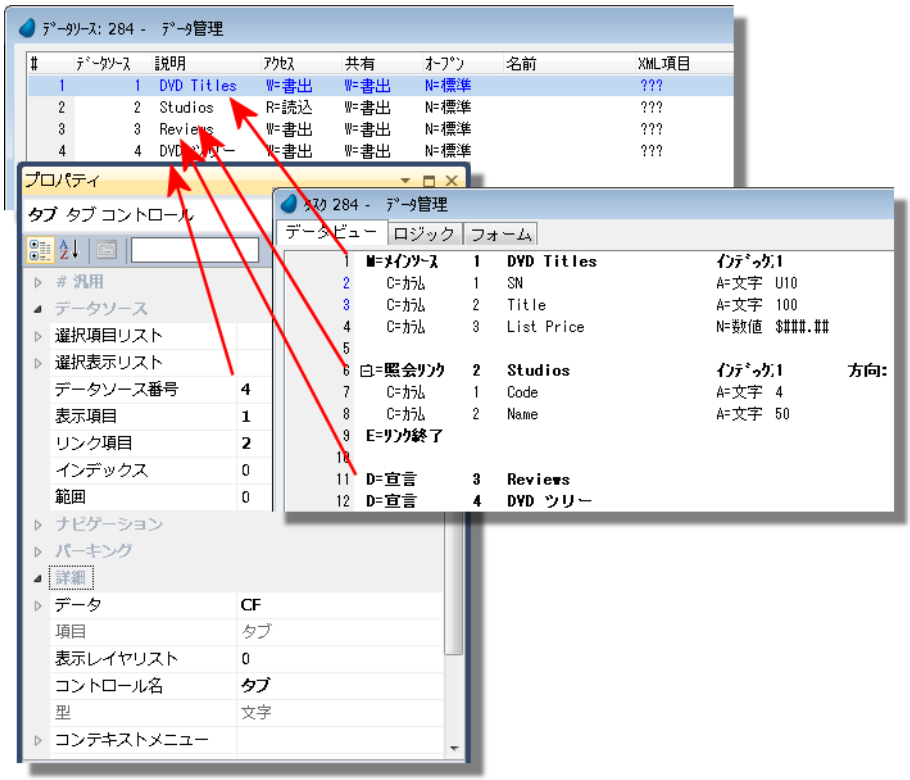
タスクには、以下の場所でデータソースを定義することができます。

- メインソース
- リンクテーブル
- 宣言テーブル
- データコントロール

定義された各データソースに対して特性シート上で特性値の設定／参照を行うことができますが、これらをまとめてアクセスした方が便利な場合があります。データソーステーブルを表示させることでこれらを一元管理することができます。

データソーステーブルにアクセスする

1. タスクを開いた状態で ***Ctrl+D***（*タスク環境→データソース*）を押下します。
2. データソーステーブルが表示されます。データコントロールで使用しているデータソース以外は、ここで特性値を修正することができます。



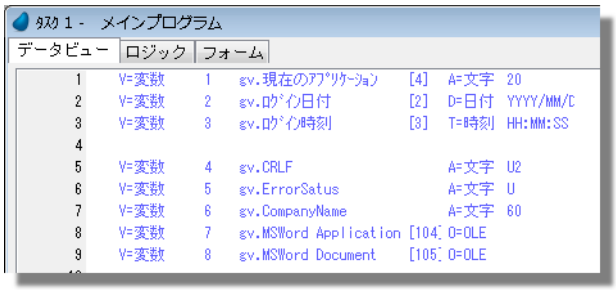
第 18 章 : メインプログラム

アプリケーションを初期設定するには

アプリケーションを設定する際に、アプリケーション内の全てのプログラムが実行される前にグローバルに設定する必要のある項目があるかもしれません。Magic xpa では、これらの項目を**メインプログラム**で定義します。これは**プログラム**リポジトリの最上位にある特別なプログラムです。

メインプログラムは、プロジェクト内の全てのプログラムの「親」タスクと考えることができます。(デバッグモードでテストする場合も) プログラムが起動される前に、**メインプログラム**の**タスク前**が実行されます。実行中のプログラムから、**メインプログラム**内の**ユーザ定義関数**や**変数項目**にアクセスすることができます。そして、プロジェクトがクローズされると (デバッグモードの場合は、プログラムが終了すると)、**メインプログラム**の**タスク後**が実行されます。

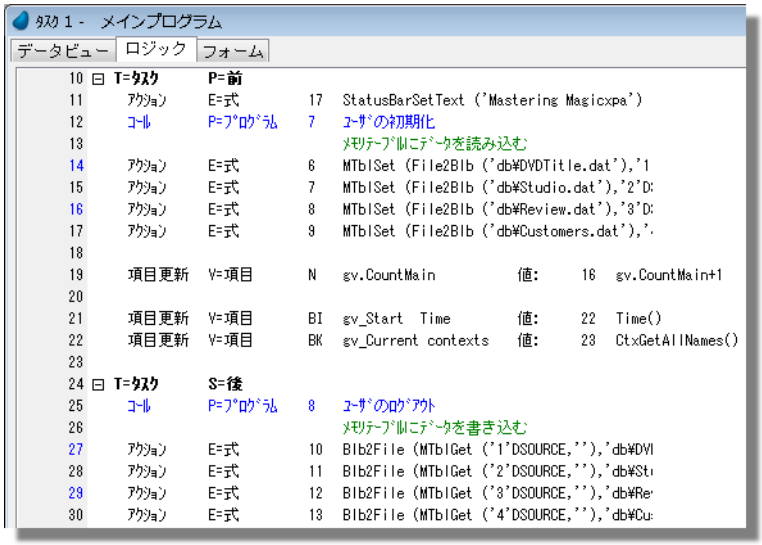
メインプログラムのデータビューエディタ



メインプログラムの**データビューエディタ**で、アプリケーションで使用する (OLE オブジェクトを含めて、) 変数項目を定義することができます。**タスク前**でこれらの項目を初期設定することができます。

SOAP サービスで使用される様々なタイプの変数項目 (CRLF データや BLOB など) が必要な場合、**データビューエディタ**は、これらを格納しておく便利な場所となります。

メインプログラムのロジックエディタ



1. **タスク前** : プログラムが起動される前に実行させたい処理コマンドを定義します。この例では、ファイルを元に BLOB 化された Magic テーブルを (メモリ) テーブル化する処理が定義されています。ログオンしたユーザの記録をとるためのプログラムをここで呼び出すこともできます。

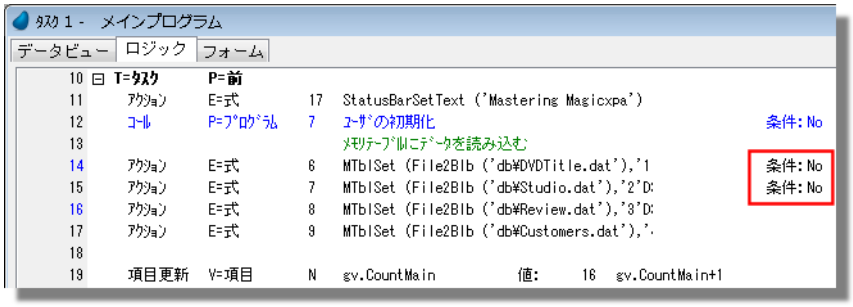
2. **タスク後**：ユーザがプロジェクトを終了する際に、実行させたい処理コマンドを定義します。この例では、メモリテーブルの内容を BLOB 化およびファイル化する処理が定義されています。ここにも、様々な処理を実行するためのプログラムを呼び出すことができます。
3. **関数：関数**ロジックユニットを定義することもできます。ここに定義された**ユーザ定義関数**は、グローバルにアクセスすることができ、Magic xpa の**関数一覧**に表示されます。
4. **イベント**：ここに定義された**イベント**ロジックユニットはグローバルにアクセスできます。この例では、***Ctrl+S*** を押下すると、どのプログラム上であってもスペルチェックプログラムが実行されるように定義されています。**メインプログラム**内でこの種のロジックを定義することで、個々のプログラム内で同じ処理を何回も定義する必要がなくなります。
   イベントロジックユニットを定義することで、**エラー**タイプのイベントを使用してグローバルにエラーを処理したり、**ActiveX** イベントを処理したりすることもできます。
5. **項目**：**項目変更**ロジックユニットは、メッセージを表示させたり、ログを出力する目的で使用することができます。

メインプログラムのフォームエディタ

フォームエディタでは、アプリケーション全体で利用可能なフォームを定義することができます。例えば、ここに一般的な帳票ヘッダを定義することで、多くの異なる帳票用に使用することができます。

アプリケーションの背景を定義するために使用することもできます（「」（423 ページ）を参照してください）。ここに定義されるフォームは、アプリケーションのグローバルなコンテキストメニューを設定する場所でもあります。

初期設定のプログラムを省略するには



メインプログラム内に定義されているあらゆる初期化処理は、**条件**カラムに条件式を設定することで簡単に処理を制御することができます。**条件**カラムを ***No*** に設定すると、初期化処理は実行されなくなります。

しかし、一定の条件（テスト環境など）でのみ初期設定をスキップさせたい場合があるかもしれません。プログラムをテストしている場合や **APG** を使用してテーブルを参照する場合も、**メインプログラム**は自動的に実行されます。この場合、初期設定処理よって起動が多少遅くなります。**タスク前**でユーザ操作が必要なプログラム（時刻の入力など）があった場合、これによって起動時間はさらに遅くなります。

従って、これらの状況で条件付きで処理を実行させないようにするために **RunMode()** 関数を使用することができます。この関数を実行すると以下の値が返ります。

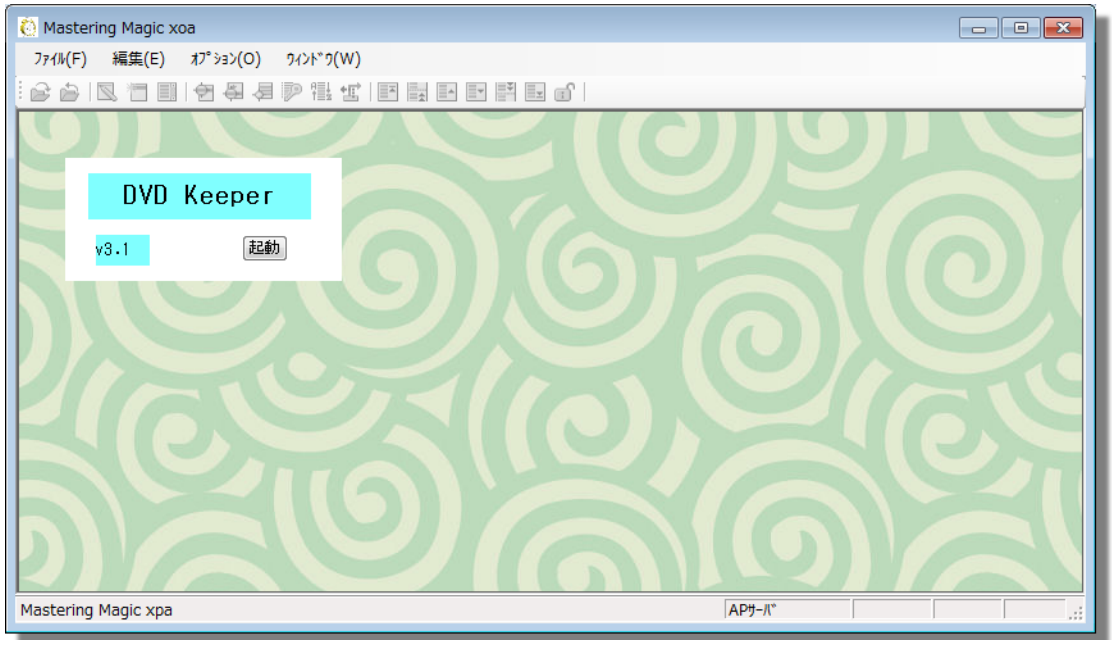
- -1：アプリケーションの**メインプログラム**がアプリケーションサーバで最初に実行された
  アプリケーションが閉じ、別のアプリケーションが最初のコンテキストでオープンされた
  バックグラウンドエンジンによるマルチスレッドでアプリケーションが最初にオープンされた
- 0：アプリケーションが実行エンジンで（フォアグランド／バックグラウンド／開発環境でのバックグラウンド）動作している
  関数が、呼び出されたプログラム／サブタスクで評価された場合
- 2：プログラムが開発エンジンで実行していて、実行モードに切り替わった（**デバッグ→実行**（*F7*）または **オプション→ APG**（*Ctrl+G*））
- 3：デバッグモードでプロジェクトが実行している（**デバッグ→プロジェクトを実行**（*Ctrl+F7*））

例えば、開発モードの場合のみ初期設定処理を無効にするには、以下の条件式を設定します。

Runmode()<2

これで実運用の場合のみ処理が実行されるようになります。

アプリケーションの背景 / 壁紙を設定するには



メインプログラム で背景や壁紙を設定することができます。

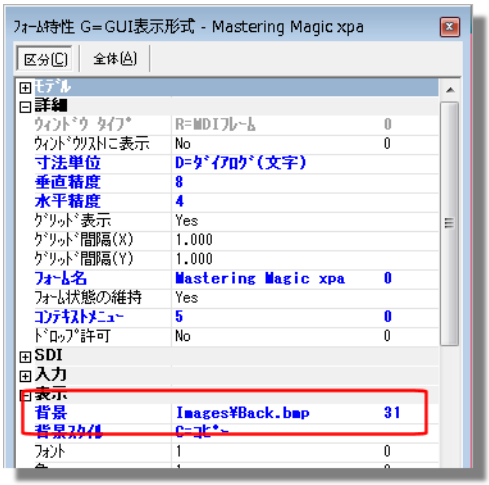
プロジェクトが実行中に、他のタスクフォームの背後にメインプログラムの GUI 表示フォームを表示させることができます。

メインプログラムのフォームではデータを入力することはできませんが、プッシュボタンのようなインタラクティブなコントロールを配置することができます。

上記の例では、背景で壁紙を表示させ、アプリケーション名とバージョン番号のテキストをタイトルとして表示させ、起動ボタンを配置しています。

メインプログラムに背景を定義する

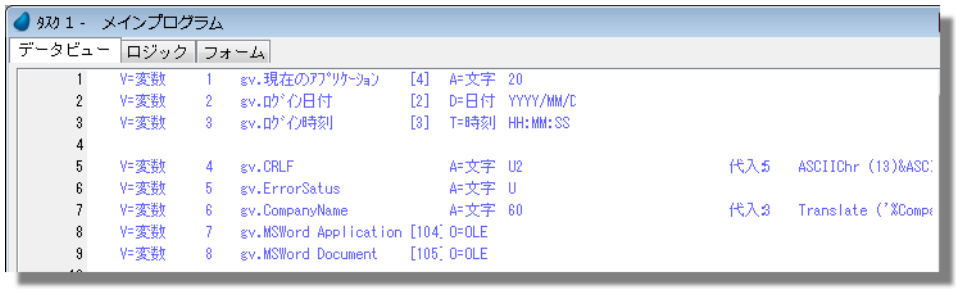
1. タスク特性 の**フォーム表示**特性を *Yes* に設定します。
2. フォーム特性の**背景**特性に式で表示させたい壁紙のファイルを定義します。



［テキスト］または［ボタン］などのコントロールは、通常の GUI 表示フォームと同じように追加することができます。

*ヒント : ファイルを式で指定した、**メインフォーム** 特性に式を使用することで実行時に背景を動的にカスタマイズすることができます。*

グローバル変数の設定と使用を行うには（コンテキスト単位）

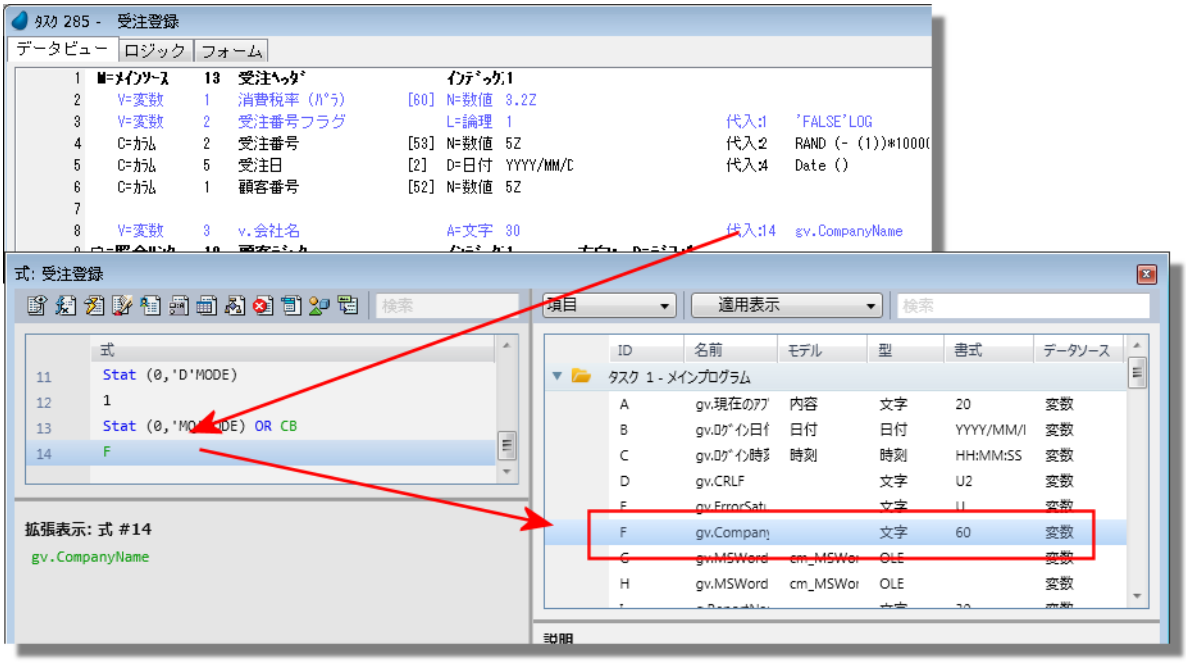


メインプログラムで定義された変数項目は、プロジェクト内のどこからでも使用することができます。この例では、Company Name という変数項目を定義しているため、この変数を様々な処理で使用することができます。

代入特性を使用して変数項目の初期設定を行うことができます。**Translate(*'%Company%'*)** は、Magic.ini に定義されている論理名をもとに会社名に変換します。別の方法として、データベーステーブルから会社名を取り出すためにタスク前でプログラムを呼び出したり、テーブルをリンクしてデータを取り出すこともできます。

注： メインプログラムでテーブルにリンクする場合、データは読み込み専用となります。

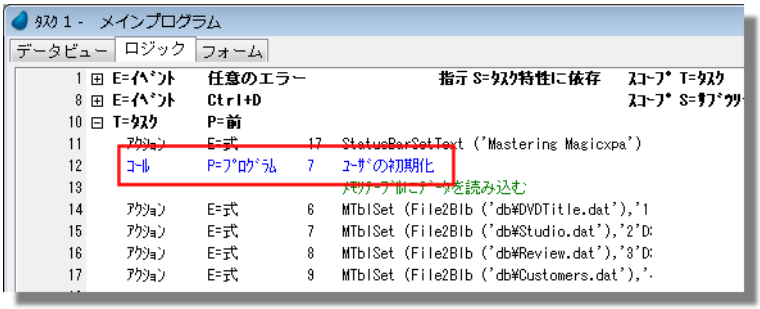
グローバル変数を使用する



メインプログラムに変数項目を定義すると、他の項目と同じようにアクセスすることができます。また、どのタスクからでも項目一覧の最上位に表示されます。式エディタでこの変数項目から値を取り出したり、項目更新処理コマンドを使用して、内容を更新することができます。

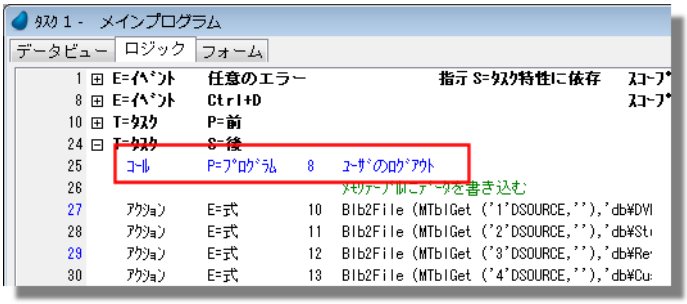
ヒント：*項目一覧では、変数項目とパラメータ項目、カラム項目によって表示色が異なります。これによって項目の内容がわかりやすくなっています。*

アプリケーション起動時の手続き処理を実行するには



アプリケーションの起動時に実行させたいプログラムは、**メインプログラム**の**タスク前**で起動しなければなりません。コンポーネントプログラムを含む Magic xpa 内のあらゆるプログラムをここで呼び出すことができます。呼び出すプログラムが同じプロジェクト内に存在する場合、呼び出されたプログラムは**メインプログラム**の変数項目にアクセスできるため、パラメータを渡す必要がありません。

アプリケーション終了時の手続き処理を実行するには



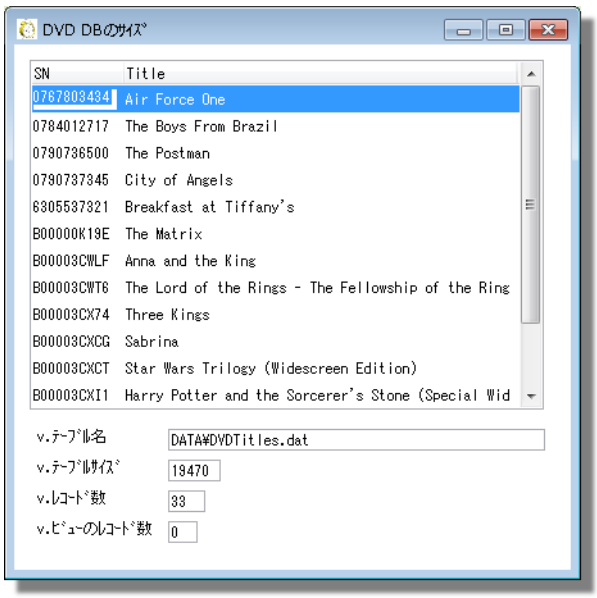
アプリケーションの終了時に実行させたいプログラムは、**メインプログラム**の**タスク後**で起動しなければなりません。コンポーネントプログラムを含む Magic xpa 内のあらゆるプログラムをここで呼び出すことができます。呼び出すプログラムが同じプロジェクト内に存在する場合、呼び出されたプログラムは**メインプログラム**の変数項目にアクセスできるため、パラメータを渡す必要がありません。

［このページは意図的に空白にしています。］

第 19 章 : データビュー

タスクのデータビュー内のレコード数を取得するには

データベーステーブルのサイズを知る必要があるかもしれません。この場合、以下の３つの関数を使用することができます。



- **DbSize()** : テーブルのサイズをバイト数で返します。
- **DbRecs()** : テーブルのレコード数を返します。
- **DbViewSize()** : 現在のデータビューのレコード数を返します。

DbViewSize() 関数のついては以下で説明します。

DbViewSize() 関数を使用する

現在のデータビュー内のレコード数を求める場合、**DbViewSize()** 関数を使用します。構文は以下の通りです。

DbViewSize (*generation*)

パラメータ :

- ***generation*** : タスクの世代番号です。(0 は現在のタスク、1 は親タスクになります)

戻り値 : レコード数を表す数値が返ります。テーブル内のレコード数ではなく、タスクで範囲指定をした場合にビューとして展開されたレコード数のみ返されます。

例 : **DbViewSize(*0*)**

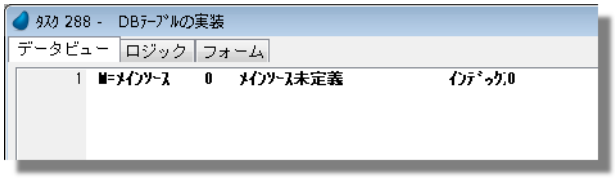
注 : デフォルトでは、データビュー内のレコードは必要に応じて取り込まれたものだけのため、**DbViewSize()** は、範囲条件に合う実際のレコード数のみ返します。この関数で正確なレコード数を取得したい場合は、タスク特性の**ビューの事前読込**特性を *Yes* に設定する必要があります。タスクが起動される前に、ビュー内のすべてのレコードが読み込まれます。これは **DbViewSize()** で正確な値を返すだけでなく、期待値どおりにスクロールバーを表示させるようにできます。

タスクのメインソースを定義するには

タスクのメインソースは、データビューエディタの最初のヘッダ行として常に表示されています。メインソースの番号と名前およびレコード表示で使用されるインデックスが同じ行に表示されています。これ以外の特性は、特性シート（*Alt+Enter*）で指定できます。

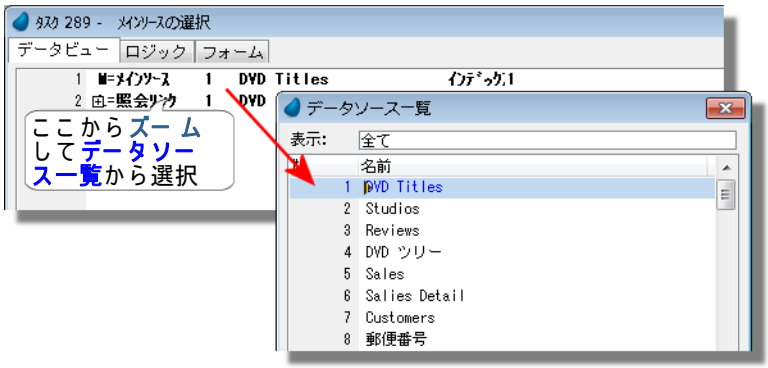
タスクが複数のレコードを検索することができる唯一の方法は、メインソースを定義することです。メインソースが定義されたら、これをもとにタスクは実行されます。表示される（バッチタスクの場合は、繰り返し処理が実行される）レコード数は、メインソースの範囲条件を満たしているレコードによって決定されます。

メインソースは必須ではありません。タスクにメインソースが定義されていない場合、最初のヘッダ行は以下のように表示されます。



メインソースのないオンラインタスクはレコードを表示しません。この場合、1 度に 1 つの（リンク）レコードを表示することができるだけになります。メインソースのないバッチタスクは、デフォルトで永久にループします。このため、タスク特性でタスク終了条件特性を指定する必要があります。

メインソースを入力する



1. **メインソース**の次のカラムにカーソルを移動します。
2. ズームして、**データソース一覧**を表示します。
3. 使用したいデータソース上にカーソルを置き、*Enter* を押下します。

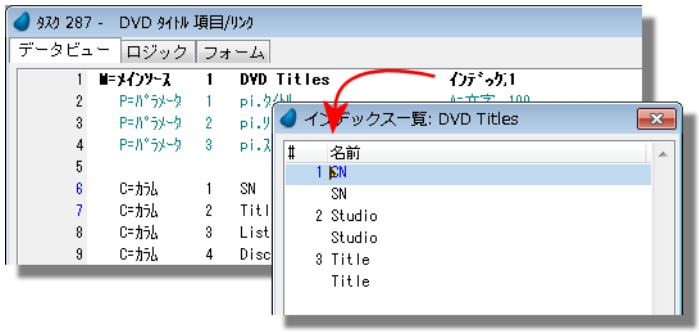
タスクで処理されるレコードの順番を動的に設定するには

メインソースが指定されると、タスクはデフォルトで、データベースの全てのレコードを検索します。レコードが処理される順番は、開発者によって設定されます。処理の順番を指定するには３つの方法があります。

- インデックス番号の指定
- インデックスを式で指定
- タスクのソートテーブルで指定

これらの方法について以下で説明します。

インデックス番号を指定する



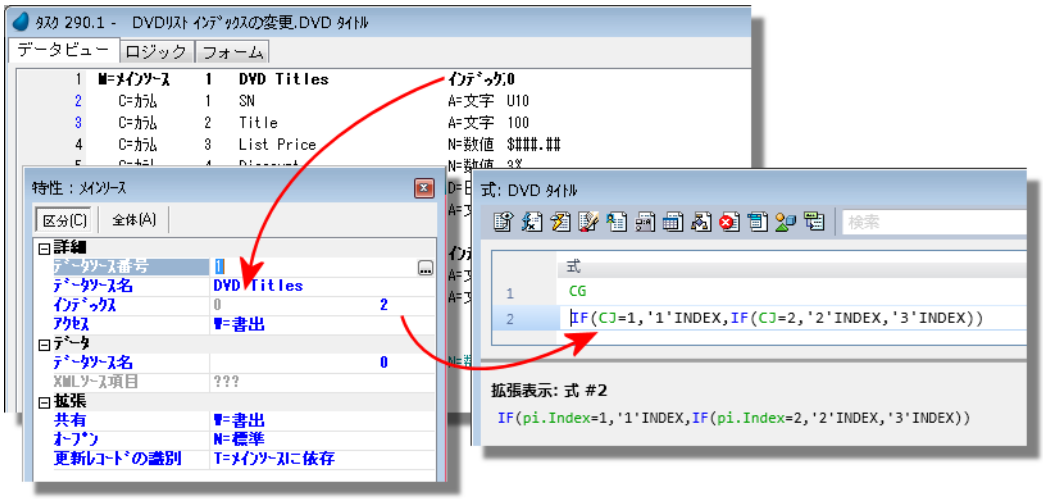
インデックスを最も簡単に指定する方法は、メインソースの**インデックス**カラムにインデックス番号を指定することです。この例では、タイトル毎に DVD をソートしています。インデックスは、**#1** が指定されています。

1. データビューエディタの**メインソース**に移動します。
2. **インデックス**カラムかズーム（*F5* またはダブルクリック）して**インデックス一覧**を開きます。
3. *Enter* を押下して使用するインデックスを選択します。

インデックスが追加／削除された場合、ここに表示されているインデックス番号も変更されます。

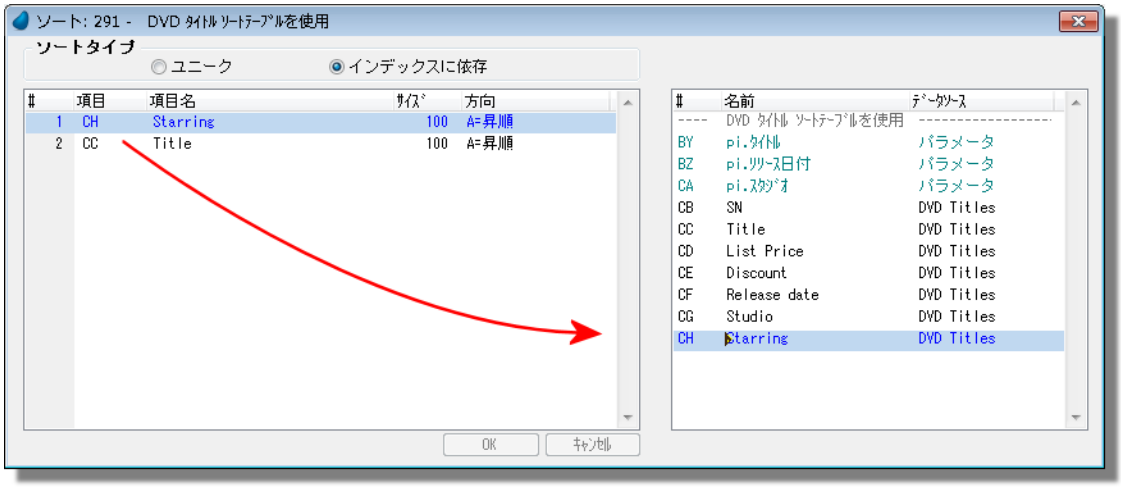
インデックス式を使用する

実行時に動的にインデックスを選択する場合は、式を使用してインデックスを指定します。この場合、ユーザがソート順序を選択することができます。また、ユーザによって選択された選択条件にもとづいた適切なソート順序が使用されるようにロジックを組み込むこともできます。



1. **メインソース**から *Alt+Enter* を押下して**メインソース特性**を開きます。
2. **詳細**セクションの**インデックス**特性に移動します。右側の**式**特性に移動します。ここからズームして、実行時にインデックス番号として評価される式を指定します。
   この例では、ユーザが入力した選択条件をもとにインデックスを選択しています。ユーザが選択した値（1、2、または 3）を使用するのではなく、対応する **INDEX** リテラルを指定するようにしていることに注意してください。インデックスが今後変更されても、**INDEX** リテラルは自動的に変更されるためこの指定方法は重要です。

タスクのソートテーブルを使用する



使用したいインデックスが定義されておらず、表示する順番を変更することだけが問題の場合は、Magic xpa に内蔵されているソート機能を使用します。このソート機能ではどのような項目でも指定することができ、Magic xpa は実行時にテーブルに一時的なインデックスを作成します。

1. ソート処理を実行させたいタスクを開きます。
2. **ソート**テーブル（**タスク環境→ソート**、または ***Ctrl+T***）を開きます。
3. ウィンドウの左側は、何も設定されたいないか（まだソート定義されていない場合）、項目が一覧表示（既にソートが定義されている場合）されています。この一覧によってテーブルのソート順序が決まります。この例では、Starring と Title の２つの項目ソートされるように定義されています。
4. 一覧に項目を加えるには以下のようにします。
   - ***F4***（**編集→行作成**）を押下して１行追加します。
   - **項目**カラムで、項目のシンボル名を入力するか、**ズーム**して一覧から選択します。
   - 項目の長さが長すぎる場合、ソート処理が効率的に実行されない場合があります。このような場合、**サイズ**カラムで文字数を入力することによって、ソートで使用される文字数を短くすることができます。この例では、Starring と Title の２つの項目の長さは 100 桁ですが、最初の 20 桁のみ使用するように指定しています。
   - 必要に応じて**方向**カラムを降順に変更することもできます。日付項目に対してこの指定を行うことで、最新の情報から表示されるようになります。
5. 項目をリストから削除する場合は、***F3***（**編集→行削除**）を押下します。
6. 設定が終了したら、OK を**クリック**します。

ヒント: *ソート処理は範囲処理の後に実行されます。レコードの範囲を指定してレコード数を制限した後にソートを行うことで、テーブルのソート処理が早く実行されることになります。*

プログラムに受け渡しするパラメータを定義するには

様々なプログラム言語では、パラメータとしてデータの受け渡しができるようになっています。パラメータは、データを渡したり、受け取ったりすることができます。

相手から受け取るデータをパラメータとして定義するだけでなく、戻り値として定義することもできます。戻り値は、一般的に関数として動作するプログラムのために使用されます。戻り値を使用することで、**CallProg()** 関数を使用して式でプログラムを呼び出すことができます。戻り値は、コールプログラム処理コマンドの**戻り値**カラムで受け取ることもできます。

以下で、パラメータと戻り値の定義方法について説明します。

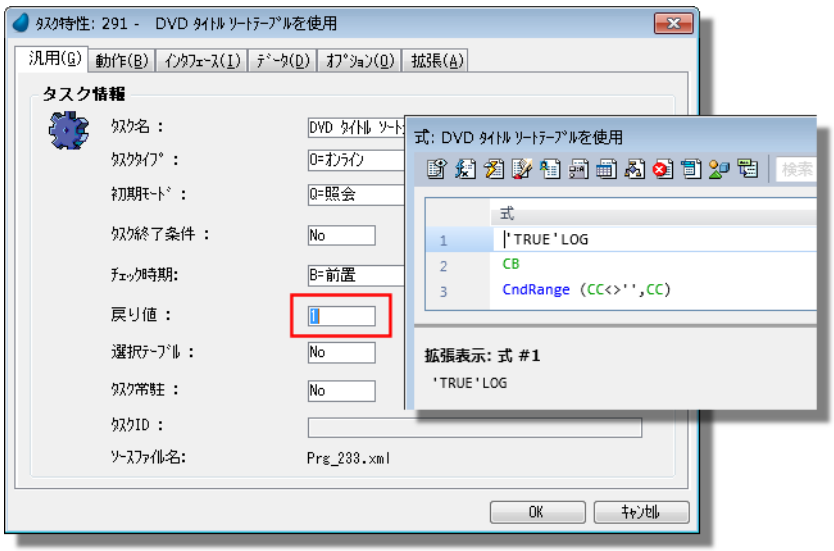
パラメータ項目を定義する

1. パラメータ項目は、他の変数項目と同じ方法で定義します。唯一の違いは、***カラム***または***変数***ではなく***パラメータ***を選択することだけです。
2. パラメータの**名前**を入力します。Magic xpa から見て、入力された名前はあまり重要ではありませんが、このプログラムを利用する際、定義されたパラメータが入力用か出力用または共用のものかどうかが確認できるようにするために、何らかの命名規約を決めておくことを推奨します。例えば、入力パラメータは先頭に pi を付加し、出力用は po、共用は pio を付加するようにします。
3. 他の項目と同じように**型**や**書式**およびその他の**項目特性**を設定します。**項目モデル**を指定することで、パラメータの項目長を合わせ易くなります。
4. パラメータ項目の定義場所はあまり重要ではありません。**データビューエディタ**の先頭や下部に定義されていたり、他の項目の間に分散されて定義されているかもしれません。しかし、定義場所を（先頭に定義するとか）あらかじめ決めておいた方がメンテナンスが容易になります。

定義されたパラメータ項目は、**コール**処理コマンドの**パラメータ**テーブルに表示されます。また、コンポーネントや COM、または SOAP オブジェクトを作成する際にも使用されます。このプログラムを呼び出す処理を定義する際、**パラメータ**テーブルでパラメータ名やデータ型を確認することができ、**構文チェック**を実行するとパラメータの数とデータ型の整合性をチェックします。

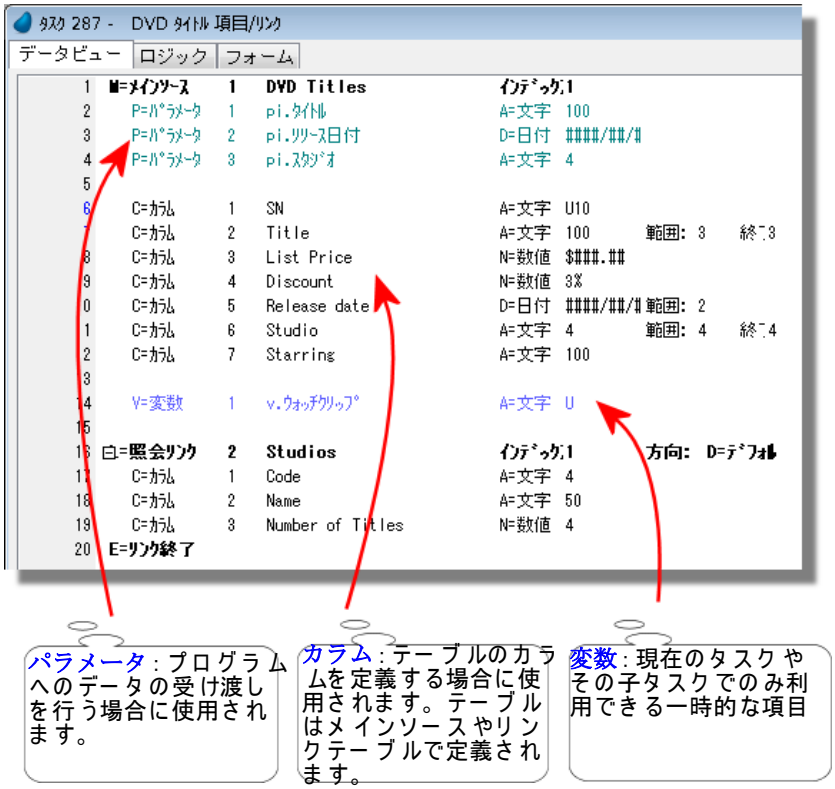
注：　パラメータを式で指定した場合、値を更新することができません。従って、項目 G が定義されており、G を項目カラムに入力した場合、更新することができます。項目 G を式で指定した場合、G の値はプログラムに渡りますが、項目の値は更新されません。値が変更される必要がない場合のみ、式でパラメータを指定することができます。

戻り値を定義する



1. タスク特性の**戻り値**（汎用タブ）に移動します。
2. ズームして式エディタを開きます。
3. 返したい値として評価される式を指定します。

タスクのデータ項目を定義するには



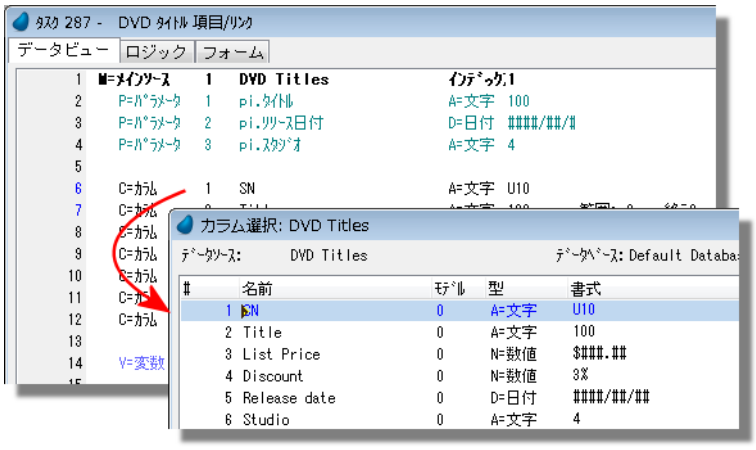
Magic xpa には 3 種類の項目があります。

- パラメータは、プログラムへのデータの受け渡しを行う場合に使用されます。変数項目と同じ方法で定義します。この例では、スタジオ番号（SN）に値を渡します。
- カラムは、データソースから選択される項目です。Magic xpa では、データソースは SQL テーブルやメモリテーブル、または XML ファイルが使用されますが、全て同じ方法で扱われます。この例では、メインソース（DVD テーブル）とリンクテーブル（スタジオファイル）からカラムを選択しています。
- 変数は、現在のタスクやその子タスクでのみ利用できる一時的な項目です。この例では、パーク可能なプッシュボタンを定義するために使用されています。

これらの項目が定義されると、各種類に対応した動作を実行します。主な違いは、タスクの終了時のデータの扱いです。

パラメータ項目に関する詳細は、「プログラムに受け渡しするパラメータを定義するには」（434 ページ）を参照してください。以下でカラムと変数項目の定義方法について説明しています。

カラムを定義する

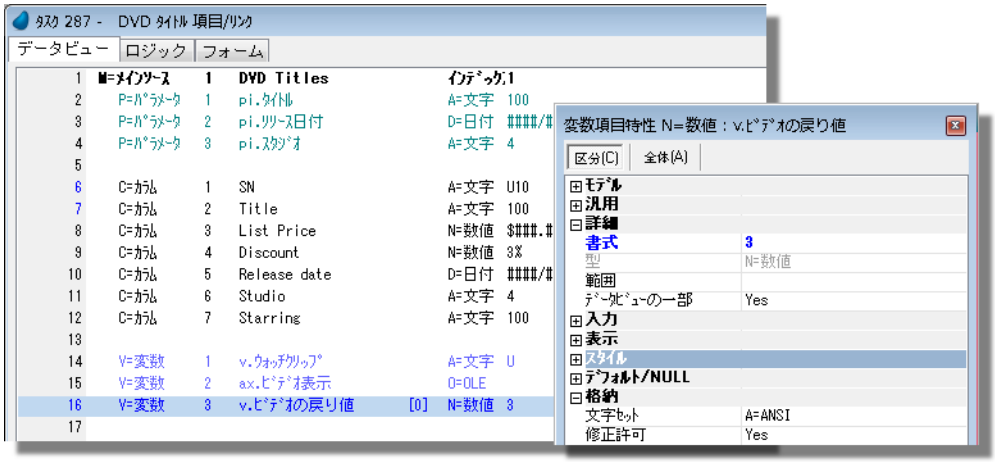


1. リンクテーブルのカラムを選択する場合、リンクとリンク終了の間にカーソルを置きます。これ以外の場所にカーソルがある場合、メインソースのカラムとして扱われます。

2. *F4*（編集→行作成）を押下して１行追加します。
3. *C* を入力するかプルダウンメニューから ***C= カラム***を選択します。*Tab* 移動します。
4. ズーム（*F5*、またはダブルクリック）してカラムを選択するか、カラム番号を入力します。*Tab* 移動します。
5. 名前カラムに入力することで、必要に応じて変更することができます。同じデータソースをリンクしているなどで、同じ名前のカラムが複数定義されている場合、名前を変更する方がわかりやすくなります。データビューで名前を変更しても、データリポジトリには影響しません。

これで定義できました。カラムの特性はデータリポジトリで設定されるため、ここで設定する必要はありません。

変数項目を定義する



1. 変数項目を作成したい行にカーソルを置きます。変数はどこにでも作成できます。
2. *F4*（編集→行作成）を押下して、１行追加します。
3. *V* を入力するかプルダウンリストから ***V= 変数***を選択します。*Tab* 移動します。
4. 変数名を入力し、*Tab* 移動します。
5. **モデル**カラムには、必要に応じて変数に定義する項目モデル番号を設定します。この例では、変数 b.WatchClip はモデル **#40** を使用していますが、変数 v.VideoReturnCode にはモデルが使用されていません。

   モデルを使用することで、開発工数が削減され、プログラムの保守が楽になります。

   モデルを選択するには、ここからズーム（*F5*、またはダブルクリック）します。

   モデルを使用している場合、変数項目の特性はすでに設定された状態になっています。必要に応じて、デフォルトの特性値を変更することができますが、モデルの使用方法についてよく理解した上で行う必要があります。モデルを使用していない場合、*Tab* 移動して次のステップを行ってください。
6. モデルの次のカラムは**型**カラムです。ここでは、項目の型（文字型、数値型日付型、時刻型、OLE 型など）をプルダウンメニューから選択することができます。*Tab* 移動します。
7. 次に**書式**カラムに移ります。変数の書式を入力します。入力する上での補助が必要であれば、ここからズームして、**書式**ダイアログを開きます。

OLE オブジェクトのように更に特性を設定する必要なものもありますが、ほとんどの変数はこの時点で、利用することができます。**特性**シート（*Alt+Enter*）を開いて、必要な特性値を設定することができます。

タスクのデータビューに範囲を定義するには

デフォルトでは、Magic xpa はメインソースのすべてのレコードを表示します。例えば、APG を使用してプログラムを作成した場合、テーブルのすべてのレコードが表示されます。

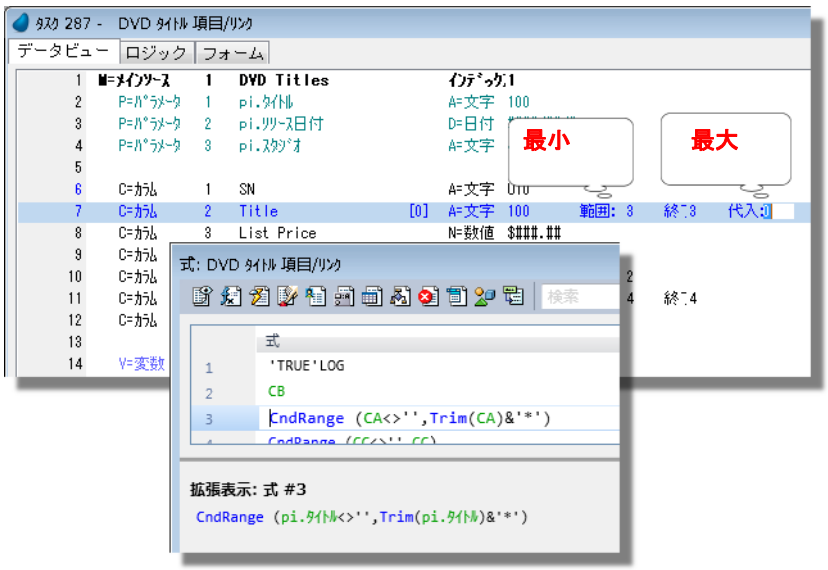
しかし、通常はすべてのレコードを表示することはありません。例えば、削除フラグが設定されたレコードを表示しないようにしたり、代理人による販売内容のみ表示させるようにすることができます。また、1 つのユニークなレコードを検索する場合もあります。

データビューでレコード数を制限するには、いくつかの方法があります。範囲カラムに直接範囲条件を設定する便利な方法もありますが、このようなオプションのほとんどは、範囲ウィンドウ（タスク→範囲 / 位置付、または *Ctrl+R*）にあります。

- 範囲：データ項目で範囲を定義します。
- 式：論理式で範囲を定義します。
- SQL Where 句 ：SQL Where 句を指定して範囲を定義します。

範囲設定についての詳細な方法を以下で説明します。

範囲の最大 / 最小を使用する



範囲指定の 1 番目の方法は、最大 / 最小の指定です。これは、最も簡単でよく利用される方法です。この場合、上限と下限のフィルタを指定し、指定された範囲内のレコードのみに絞ることを可能にします。文字型のカラムの場合、いくつかのマスク文字が指定できます。例えば、検索文字列の後に「*」が付加されている場合、移行の文字列を無視するように検索できます。

このような範囲指定方法は、一般的に、1 種類（同じ状態、同じ国、同じ親レコード ID）のレコードで絞り込んだり、最小と最大を同じ値に指定することで 1 つの特定のレコードを絞り込むために使用されます。

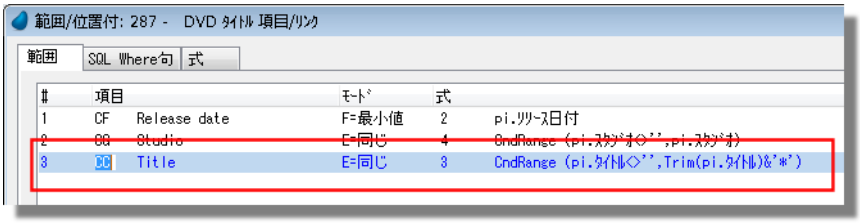
この例では、プログラムに渡された文字列で始まっているすべてのレコードを検索しています。The L という文字列で始まるレコードが存在する場合、結果として表示されるレコードには、映画「The Load of the Rings」の全てのレコードが含まれています。

しかし、この方法では、文字列の中に存在するテキストを検索したり、より複雑な条件で検索することはできません。このような場合は、範囲式を使用する必要があります。

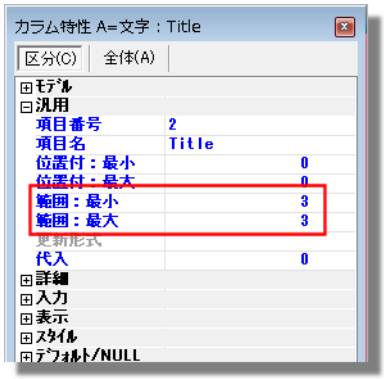
範囲指定を異なる場所から確認する

便宜上、範囲の最小 / 最大指定は、いくつか異なる場所で参照することができます。ここには、上記で設定された範囲条件をそれぞれの場所で表示させています。

データビュー内に表示される範囲カラム



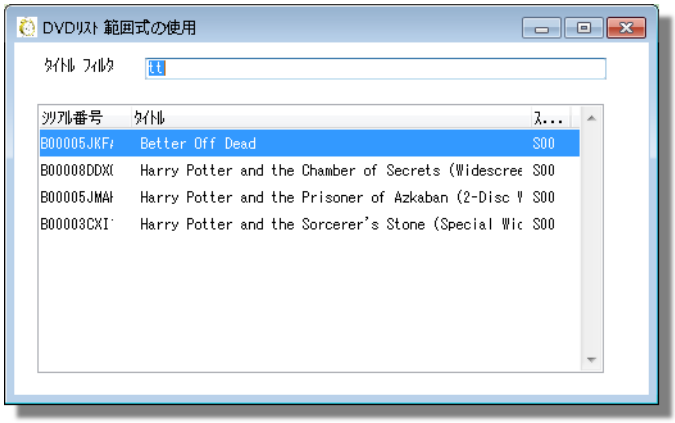
範囲ウィンドウ（*Ctrl+R*）で同じ範囲条件が指定されています。



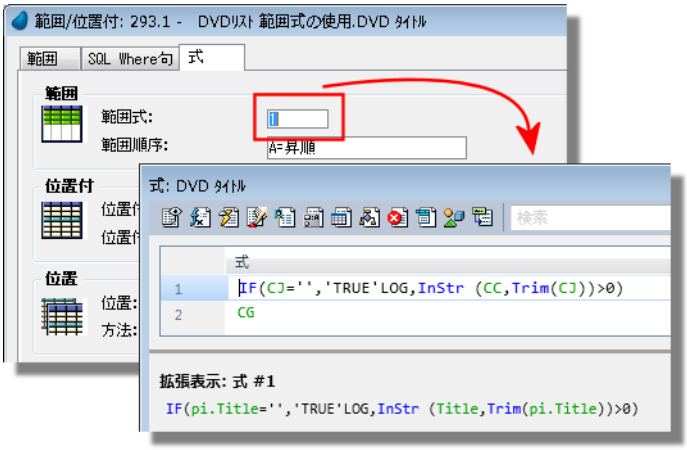
カラム特性（*Alt+Enter*）での範囲指定

どのオプションを使用しても構いません。どこで指定しても同じ動作になります。

範囲式を使用する

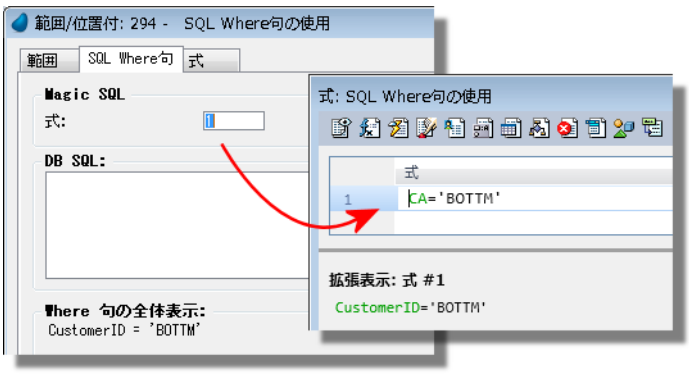


範囲式オプションを使用することで、より柔軟な範囲設定を行うことができます。この方法では、任意の式を入力することができ、式が True に評価された場合、レコードは選択されます。この例では、指定された値が文字列のどこに存在していても、True が返るような式を指定しています。



上記の例には、**範囲式**特性（**範囲 / 位置付→式タブ**）に入力された式が表示されています。パラメータが空白の場合、式は常に True を返すため、すべてのレコードが表示されます。空白でない場合、指定された文字列が現在のレコードの文字列内に存在するかどうかをチェックするために、**InStr()** 関数を使用しています。存在している場合、**InStr()** 関数は正の数値（0 より大きい数値）を返すため、該当するレコードはビューに含められます。

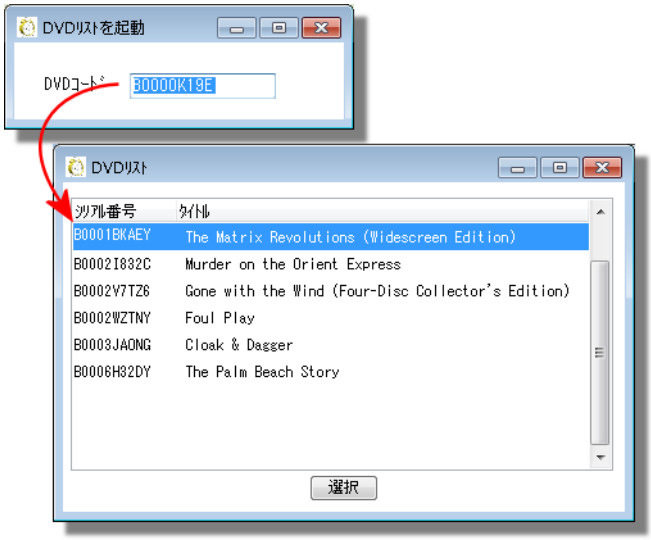
SQL Where 句を使用する



SQL テーブルを使用している場合、通常の範囲指定（最大 / 最小）オプションは実行時に SQL ステートメントに変換されます。しかし、SQL コードを **SQL Where** タブの **DB SQL** 特性に入力することもできます。この方法の不利な点は、SQL コードが使用する DBMS に依存する場合があるということです。このため、DBMS を切り換えて使用する場合、移植性がなくなる可能性が発生します。

その代わりに **Magic SQL 式**を入力することができます。この場合、実行時に使用する DBMS 用の SQL コードに変換されます。

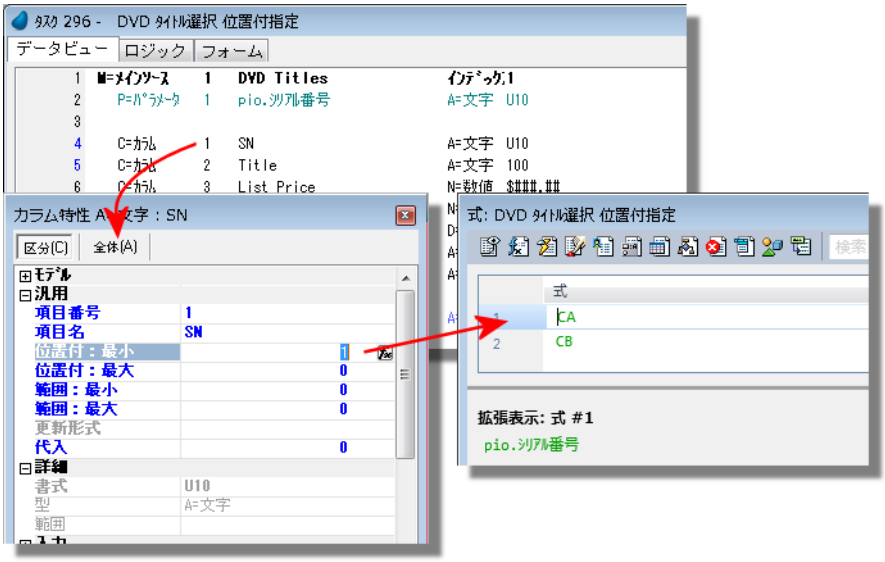
タスク起動時に特定のレコードに位置付けるには



テーブルを表示する際に、任意のレコードに位置付けたい場合があります。この例では、DVD タイトルを入力する項目で**ダブルクリック**すると、カラムがそのそのタイトルに移動します。

これは、**位置付**特性を使用することで実行されます。**範囲**特性が、表示されるレコード数を限定する処理であるのに対し、**位置付**特性は現在のレコード位置が変わるだけです。

位置付特性を使用する



1. **データビューエディタ**の位置付け条件を設定したい**カラム**にカーソルを移動します。この例では、シリアル番号（SN）カラムを表示し、パラメータとして渡されたシリアル番号と同じ番号の最初の DVD にレコードが位置付くようにしています。
2. *Alt+Enter* を押下して、**カラム特性**を開き、**位置付 最小**特性に移動します。
3. **位置付 最小**特性で**ズーム**して**式エディタ**を開きます。ここに、位置付け条件として評価される式を入力します。

これにより、プログラムが起動されると**位置付**特性で指定された値と同じ値を持つ最初のレコード上にカーソルが位置付きます。該当するレコードが存在しない場合、その次に相当するレコードに位置付きます。

位置付順序の効果

位置付特性のデフォルト設定は *A= 昇順*です。従って、この例では、検索処理はレコードの先頭から最後まで実行されます。一致した最初のレコードに位置付くように、**位置付 最小**特性を使用しています。

しかし、**位置付順序**特性（**範囲 / 位置付→式タブ**）で降順を設定した場合、検索処理はレコードの最後から先頭方向に実行されます。この場合、**位置付 最小**特性に位置付条件を設定する必要があります。

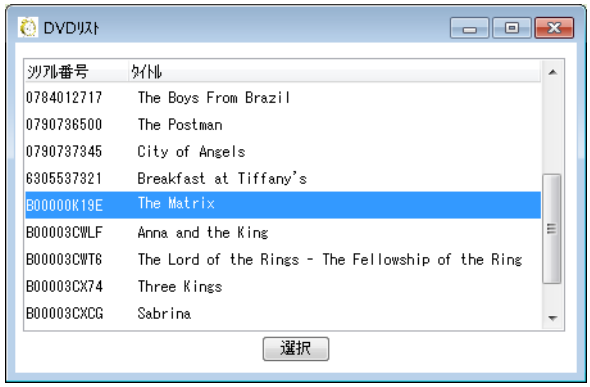
位置付の最小 / 最大の両方を使用する

位置付 最小と**位置付 最大**の各特性に位置付け条件を設定した場合、レコードが見つからないと、エラーメッセージとして「レコードが見つかりません - 最初に位置付けます」が表示されます。

位置付式を使用する

レコードを検索するために論理式を使用する場合、**位置付式**特性（**範囲 / 位置付→式タブ**）に入力することができます。この場合、範囲式（「範囲式を使用する」（440 ページ）を参照してください）と同じように動作します。

テーブルの中央に位置付ける



デフォルトで、位置付けされたレコードはテーブルの先頭行として表示されます。この場合、上の行にレコードが存在しないものと受け取られる場合があります。しかし、**動作環境**ダイアログ（**オプション→設定→動作環境**）の**カーソルの画面中央位置付**（**動作設定**タブ）を *Yes* に設定することで、位置付けレコードがテーブルの中央に表示されるようになります。

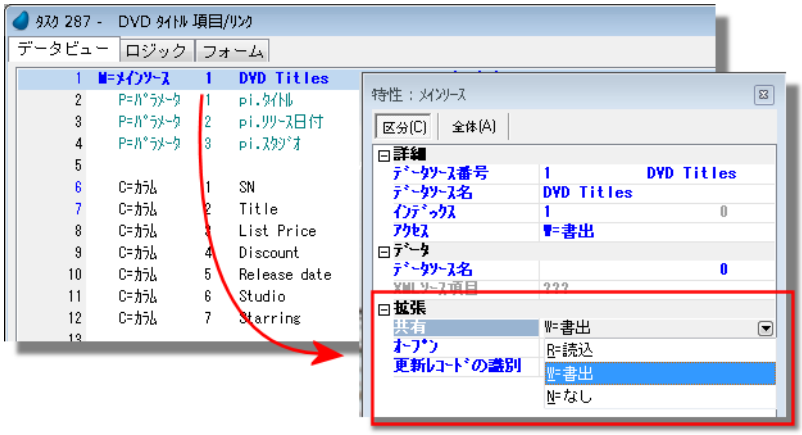
読み込み専用のテーブルにアクセスするには

テーブルを使用している時間のほとんどは、更新することがなく、他のテーブルの更新や表示、印刷を行うためにデータを読み込んでいるだけの場合が多いはずです。テーブルを更新する必要がない処理であれば、読み込み専用でテーブルをオープンすることを推奨します。これによって、レコードが競合する可能性を減らし、プログラムの処理が早くなります。

データソースを読み込み専用でオープンする

1. データビューエディタのメインソースまたはリンクテーブルのヘッダ行に移動します。
2. *Alt+Enter* を押下して、特性シートを開きます。
3. アクセス特性を、*R= 読込*に設定します。

データソースを複数のユーザ間でアクセスさせるには



複数のユーザがテーブルにアクセスしている時に、DBMS は Magic xpa と連携して、通常は適切にレコードレベルでの処理を行います。すなわち、２人のユーザが、同時に同じレコードにアクセスしようとした場合、どちらか一方はロックアウトされ、エラーメッセージが表示されますが、テーブル内のデータが破損するようことはありません。

しかし、更新中に誰もテーブルにアクセスして欲しくない場合は、多少設定が必要です。例えば、経理用のテーブルを使用して月末調整を行っている場合や、古いレコードの保管処理を行っているような場合などが、このようなケースに該当します。テーブルの**共有**特性を設定することによって、これらの処理を行うことができます。

共有特性には以下のオプションがあります。

- ***W= 書込***：このタスクの実行中は、他のタスクでこのテーブルを書込モードでオープンすることができます。
- ***R= 読込***：このタスクの実行中は、他のタスクでこのテーブルを読込モードでオープンすることができます。ただし、書込モードではオープンできません。
- ***N= なし***：このタスクの実行中は、他のタスクでこのテーブルをオープンすることができません。

この特性は DBMS レベルで実行されるため、DBMS が Magic xpa 以外のプログラムによってアクセスされる場合、これらのプログラムに対しても影響します。

タスクのデータ項目の値を設定するには

Magic xpa のプログラムでデータ項目の内容を更新するには、いくつかの方法があります。これらの方法は、以下の種類で分けることができます。

- レコード作成時に自動的に設定される値。この場合は、**デフォルト値**特性や**代入**特性を使用します。
- プログラムの処理コマンドを使用して設定される値。この場合は、**項目更新**処理コマンドや **VarSet()** 関数、およびパラメータとして値を渡す方法があります。
- ユーザが入力したり、コントロールを操作することによって更新される値。

以下では、各更新方法についての詳細が説明されています。

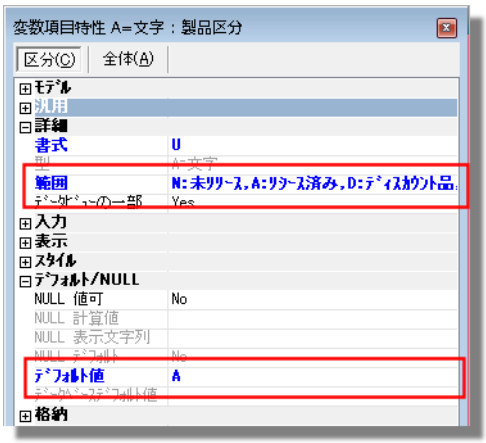
自動的に値を設定する

デフォルト値特性を使用する

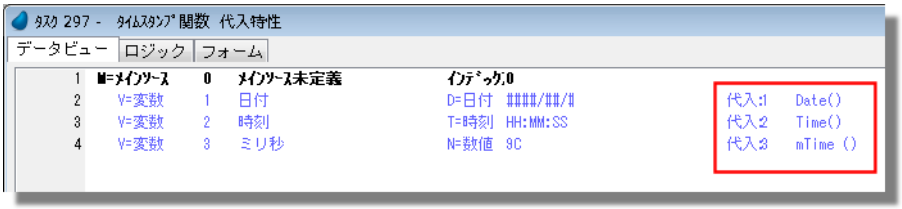
どのデータ項目に対しても**デフォルト値**特性を設定することができます。この特性は、どのレベル（モデル、データソース、タスク）で指定できます。項目にデフォルト値が設定されている場合、ユーザがレコードを参照する前に、自動的にこの値が設定されます。

範囲特性は、この項目に入力できる値を制限する場合に指定します。この例では、有効な値は *'N'*、*'A'*、*'D'*、および *'X'* になります。

デフォルト値特性は、最初にオープンされた場合に初期設定する値を指定します。



代入特性を使用する



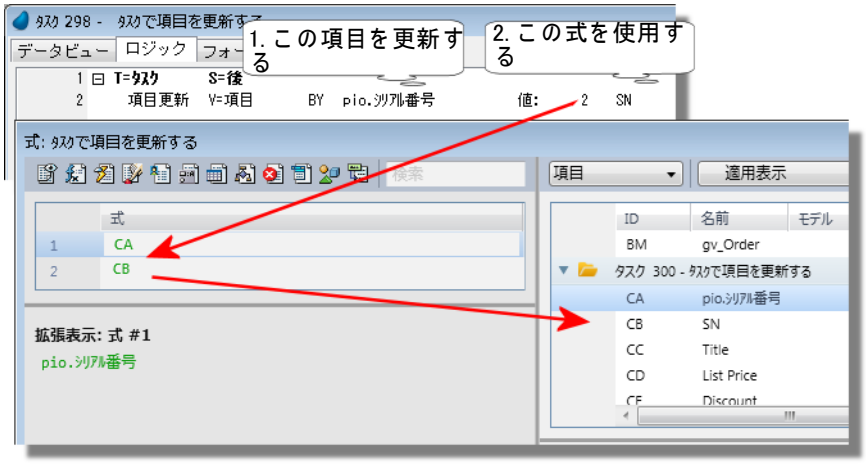
項目の**デフォルト値**特性に加え、**データビューエディタ**の項目定義の行の**代入**カラムを使用することもできます。

代入は**デフォルト値**とは少し異なった動作になります。

代入	デフォルト値
項目の初期設定	項目の初期設定
式で指定	値を直接入力
代入式の評価結果が変更されると、再計算される。	指定された値のみ有効

タスクで項目を更新する

項目処理コマンドを使用する



タスク内で項目を更新する最も一般的な方法は、**項目更新**処理コマンドを使用することです。この処理コマンドは、どのような**ロジックユニット**内でも使用できます。

項目更新処理コマンドの構文は上の図に表示されているとおりです。また、以下のようにも表すこともできます。

項目更新 〈項目〉 値 〈更新式〉 条件 〈条件式〉

ブラケット内の入力カラムでは、ズームすることで各々、項目一覧や**式エディタ**にアクセスすることができます。

VarSet() 関数を使用する

VarSet() 関数は、式の中で直接項目の値を更新することのできる関数です。様々な目的で使用することができます。関数の構文は以下の通りです。

VarSet(*variable*, *value*)

パラメータ：

- *variable*：更新対象の項目を **VAR** リテラルで指定します。**VAR** リテラルは、項目番号が P であれば、'P'VAR というように指定します。項目の定義位置が変更されても（'P' が 'Q' に変更されても）式の中で自動的に変更されます。
- *value* ：更新する値か更新値として評価される式を指定します。値を直接指定したり、項目を指定したり、関数を使用することもできます。

例：**VarSet** (*'P'VAR, X+6*)

項目 'P' の値を、項目 X の値 +6 の結果で更新します。

VarSet() 関数を使用すると、複数の項目を配列のように扱うことができます。例えば以下のような式を設定します。

VarSet(*'P'VAR+1, X+6*)

項目 'Q'（'P' の次）の値を、項目 X の値 +6 の結果で更新します。

ユーザによって項目を更新する

ユーザによって項目の値を変更することもできます。ユーザによって変更可能な項目は、**変更可**特性や**パーク可**特性が *Yes* に設定されているコントロールとしてフォームに配置されている必要があります。

1 つのタスク内で複数のテーブルからデータを検索するには

Magic xpa でデータソースを利用する場合、以下の 2 つの方法でデータを取得することができます。

- メインソースから取得
- リンクテーブルから取得

メインソースは、タスク内で繰り返しアクセスすることのできるデータソースです。メインソースから取得されたデータはフォーム上のテーブルコントロールに表示させることができ、1 つのレコードやテーブル全体を表示させることができます。1 つのタスクに対して 1 つのメインソースのみが定義できます。

リンクテーブルは、1 度 1 レコードのみ表示することができます。リンクテーブルは、一般に、現在表示されているレコードに関連する情報を表示させるために使用されます。1 つのタスクには、必要に応じて複数のリンクテーブルを定義することができます。

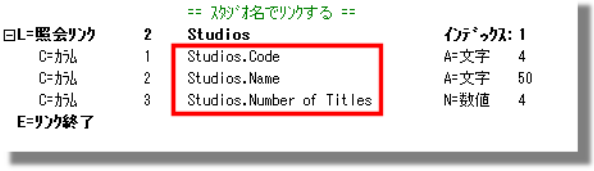
注： ユーザに対して表示されるフォームに 1 つのテーブルしか表示できないという意味ではありません。1 つのタスクに対して 1 つのメインソースしか定義できませんが、サブフォームを使用することでフォーム上に複数のテーブルを表示させることができます。サブフォームは、親タスクの一部のように表示されますが、実際はサブタスクであり、これ自身でメインソースを定義することができます。

リンクコマンドを使用してリンクテーブルからデータを取得することができます。以下はこの方法について説明しています。

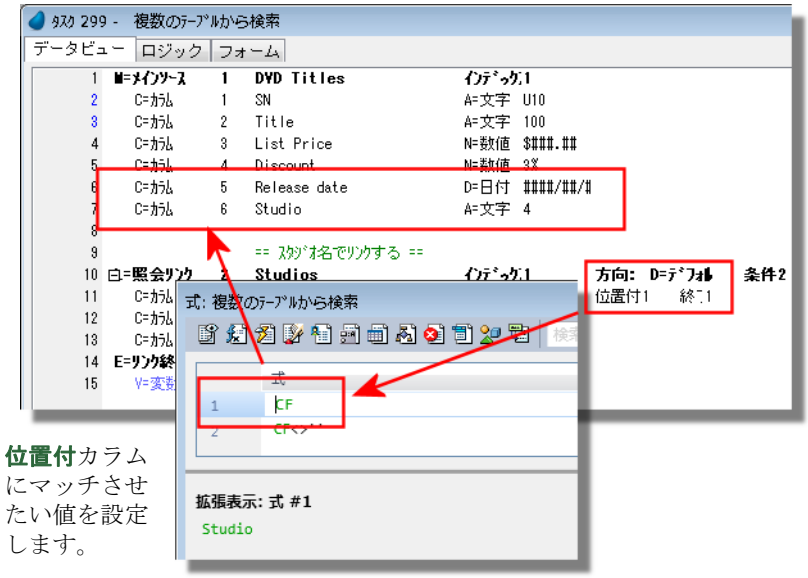
リンクコマンドを使用する

1. 最初に、リンク / リンク終了のペアの行を作成します。*F4* を押下して 1 行追加します。*L* を入力します。***照会リンク*** と ***リンク終了***の 2 つの行が作成されます。リンクさせたいデータソースを選択します。ここでは、#2 のデータソース（Studio テーブル）を選択しています。データソース番号を入力するか、ズームして一覧から選択できます。
2. 次に、**インデックス**カラムに *Tab* 移動します。インデックスは、テーブルの検索順序を決定します。指定するインデックスが、レコードの検索条件に合っているかどうかは重要な要因になります。例えば、この例では、インデックスがスタジオコードのため、スタジオコードを使用して位置付けしなければなりません。
   インデックスを選択すると、インデックスを構成するセグメントカラムが自動的に追加されます。
3. 位置付カラムを設定します（設定方法は、次のセクションを参照してください）。
4. 最後に、リンクカラムとして含めたいカラムを選択します。選択するには以下のようにします。
   - *F4* を押下して 1 行追加します。項目のタイプは自動的に***カラム***になります。次のカラムに *Tab* 移動します。
   - ズームしてカラムを選択するか、カラム番号を直接入力します。
   - 必要に応じてカラム名を変更することができます。

ヒント：*カラム名を変更することで、どの項目がどのデータソースからリンクされたものなのかをわかりやすくすることができます。*



リンクの位置付カラムを設定する

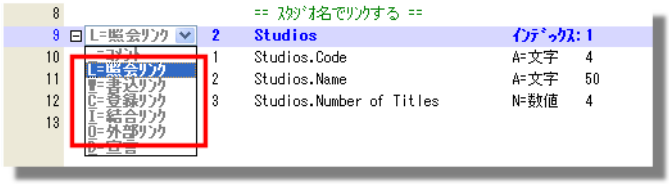


位置付カラムにマッチさせたい値を設定します。

リンクの位置付カラムは、SQL 系データベースの Select Where 句のように動作します。**位置付**と表示されているカラムは、位置付け条件の最小値を指定し、**終了**と表示されているカラムには、最大値を指定します。この例のように両方とも同じ値が設定されている場合、指定された条件値のレコードのみを取得しようとします。

従って、この例では、DVD タイトルに定義されているスタジオコードと合ったレコードが取得されます。

リンクの種類を設定する

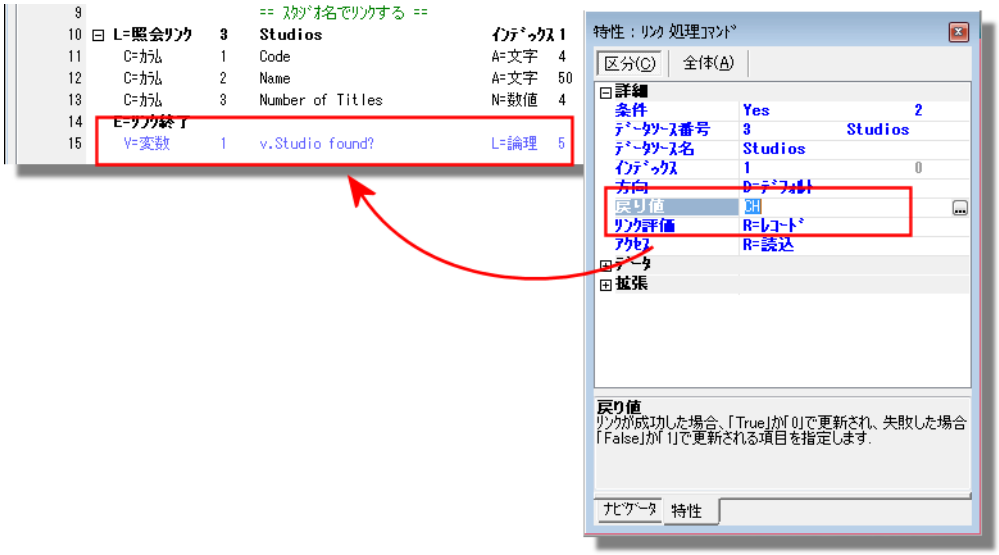


リンクにはいくつかの異なるタイプがあり、それぞれ異なる動作をします。これらはすべて、現在のタスクにカラム項目を定義するものですが、書込リンクと登録リンクはレコードの作成も行います。以下は、各リンクのタイプとその特徴です。

リンクの種類	説明	利用目的
照会リンク	既存のレコードを取得します。	既存のレコードを取得したり、レコードが存在しているかどうかをチェックします。 例えば、顧客コードが顧客テーブル内に存在していない場合に、エラーメッセージを表示させたい場合、照会リンクを使用します。
書込リンク	既存のレコードを取得しようとします。レコードが存在しない場合は、そのレコードを作成します。	レコードの存在が不確定で、存在しなければ作成したい場合に使用します。 例えば、電話番号を作成したい場合に、入力された電話番号がすでに存在していなければ、自動的に登録するようにできます。
登録リンク	レコードを作成しようとします。そのレコードが存在しているかどうかは確認しません。	レコードが存在していない場合は、書込リンクより処理が早くなります。 例えば、ユーザが一定の画面を開いている場合に、常にログレコードを作成したいのであれば、ユーザ ID や日付、およびタイムスタンプを保存するためにリンク作成が使用できます。

リンクの種類	説明	利用目的
結合リンク	データソースが両方とも SQL テーブルの場合、SQL の内部結合を実行します。	SQL テーブルを使用して内部結合を使用する場合は、SQL に対してより早くリンク処理が実行されます。
外部結合リンク	データソースが両方とも SQL テーブルの場合、SQL 外部結合を実行します。	SQL テーブルを使用して外部結合を使用する場合は、SQL に対してより早くリンク処理が実行されます。

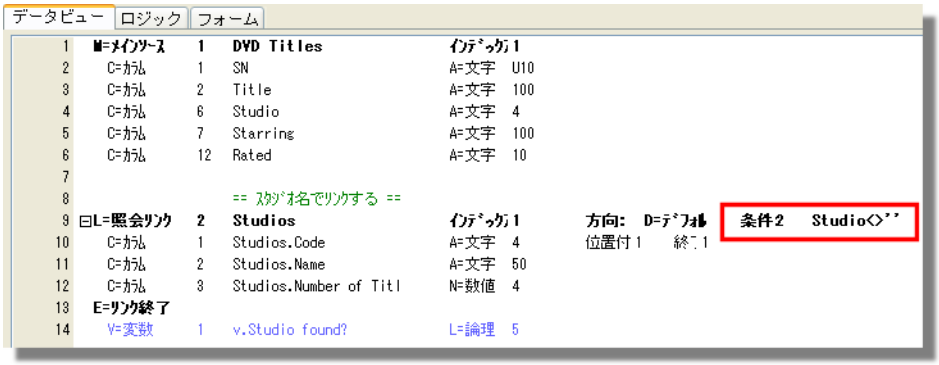
リンクの戻り値特性を設定する



リンクコマンドによって適合したレコードが存在したいるか否かは、戻り値のコードで確認できます。この値は、**戻り値**特性に設定されます。**戻り値**特性は、一般的にユーザによって入力されたデータの有効性を評価するために使用されます。

1. 論理型の変数項目を定義します（数値型も使用できますが、論理型の方が保守が容易です）。
2. 戻り値特性でズームしてこの変数項目を選択します。

リンク条件を設定する



リンクコマンドの動作を条件特性で制御することができます。**リンク**特性には、*Yes* または *No* を設定するか、式で設定できます。*No*、または実行時に式が False と評価された場合、リンクは実行されません。この例では、スタジオコードが空白の場合リンクが実行されない条件が設定されています。

リンクカラムの範囲を使用する

リンクカラムには、メインソースのカラムと同じように範囲条件を指定することができます。メインソースのカラムを使用することでデータビューとして展開するレコード数を制限することができます。例えば、メインソースのカラムで以下のような範囲設定を行った場合

- **範囲最小**：*'2001/01/01'DATE*
- **範囲最大**：*'2001/12/31'DATE*

これにより、2001/01/01 から 2001/12/31 までの日付で範囲が設定され、この範囲内のレコードのみ表示されます。

リンクテーブルに対して範囲指定を行った場合も、同じように動作します。すなわち、リンクテーブルで以下のように日付の範囲設定を行った場合。

- **範囲最小**：*'2001/01/01'DATE*
- **範囲最大**：*'2001/12/31'DATE*

リンクレコードにこの範囲の日付データが存在する場合のみ、メインソースのレコードが表示されるようになります。

これは便利な機能ではありますが、**範囲**特性に位置付用の値を設定しないように注意する必要があります。この 2 つの特性は、異なった動作をします。

テーブルの最初または最後のレコードを取得するには

タスク内で 1 つのレコード（テーブルの最初または最後）のみを取得する必要があるかもしれません。例えば特定のユーザの最後のログイン時間を更新したり、テーブル内の最大のレコード番号を取得したりする必要がある場合などです。

これらのことを行うには、以下のような手順でロジックを組み込みます。対象がメインソースの場合とリンクテーブルの場合では、方法が異なります。

メインソースの最初または最後のレコードの取得

1. インデックスを定義する

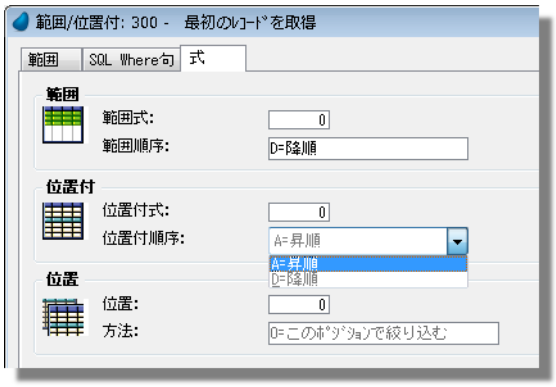
最初または最後のレコードを取得する場合に最初に考えることは、何をもとに最初と最後にするかということです。これは、テーブルには複数のインデックスが定義されている場合があるからです。例えば、顧客テーブルにおいて最初の顧客は、顧客 ID が最も小さい場合であったり、顧客名がアイウエオ順で先頭であったり、最も小さな郵便番号を持つ顧客であるなど色々考えられます。

Magic xpa では、メインソース定義でインデックスを選択することでどのインデックスを使用しているかを決めることができます。

しかし、要求される検索順に対応したインデックスが定義されていない場合は、タスクのソートテーブルを使用することで実行時にレコードのソート処理が実行されます。

最初や最後のレコードを取得する場合は、データビューに範囲や位置付を指定する必要がありません。

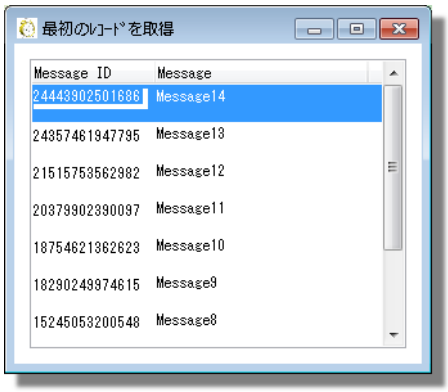
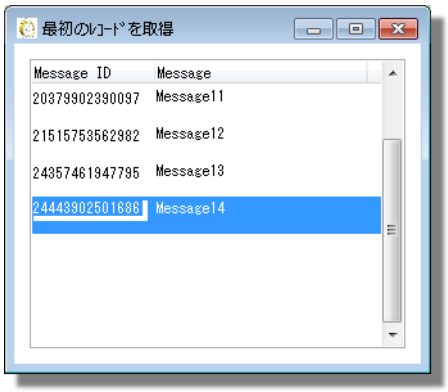
2. 検索順を設定する



次に、最初か最後かを決める必要があります。最初のレコードを取得する場合は何もする必要はありません。しかし、最後のレコードを取得したい場合は、テーブルの最後からレコードが並ぶように指定する必要があります（テーブルを最後まで検索するには非効率なためです）。

最後のレコードを取得するには、**範囲順序**か**位置付順序**（タスク環境→範囲／位置付→式）を ***D= 降順***に指定します。

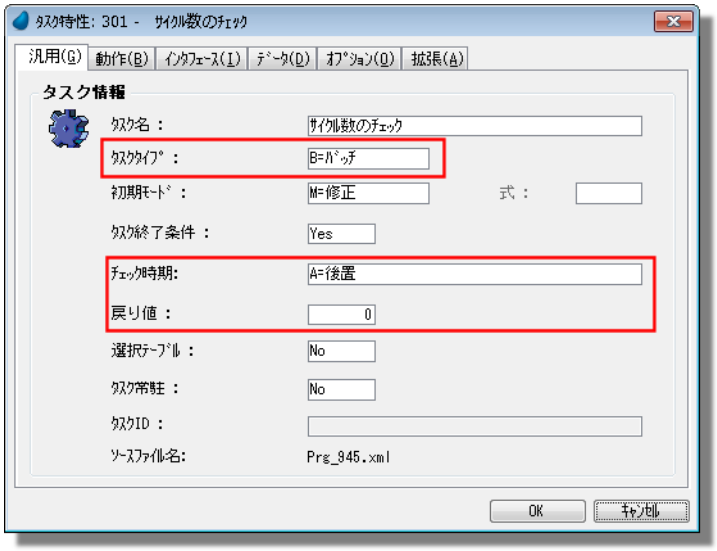
範囲順序と位置付順序

範囲順	位置付順
	
テーブルは逆の順序で検索されるため、メッセージ 4 が先頭になります。	テーブルの表示順は昇順ですが、カーソルが最後のレコードにパークします。

最後のレコードを取得する場合、**範囲順序**か**位置付順序**のどちらかを指定して行います。これらの機能は上図のように異なります。オンラインタスクの場合は、違いが明白になりますが、1 サイクルだけのバッチタスクとして実行すると同じ結果になります。

しかし、このような場合は、両方を降順にした方が理解し易くなります。

3. サイクル数をチェックする

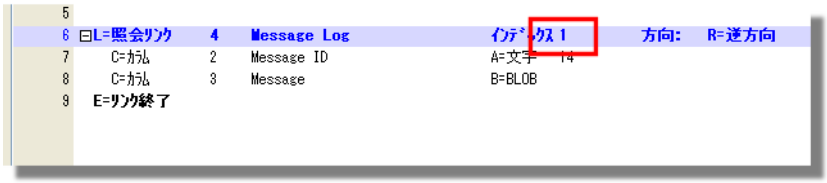


1 レコードのみを取得する場合、一サイクルのみ処理するバッチタスクを使用することになります。この場合、**タスク特性**（*Ctrl+P*）の**汎用**タブで以下の特性値を設定します。

- **タスクタイプ** ：*B= バッチ*
- **タスク終了条件** ：*Yes*
- **チェック時期**：*A= 後置*

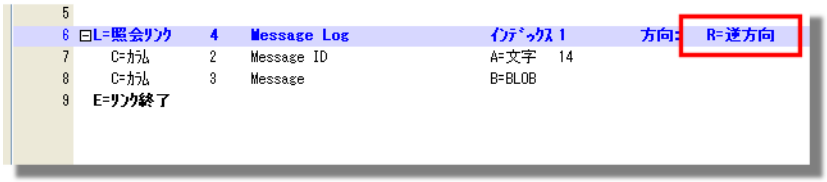
リンクテーブルの最初または最後のレコードの取得

1. インデックスを定義する



最初または最後のレコードを取得する場合に最初に考えることは、何をもとに最初と最後にするかということです。これは、テーブルには複数のインデックスが定義されている場合があるからです。例えば、顧客テーブルにおいて最初の顧客は、顧客 ID が最も小さい場合であったり、顧客名がアイウエオ順で先頭であったり、最も小さな郵便番号を持つ顧客であるなど色々考えられます。

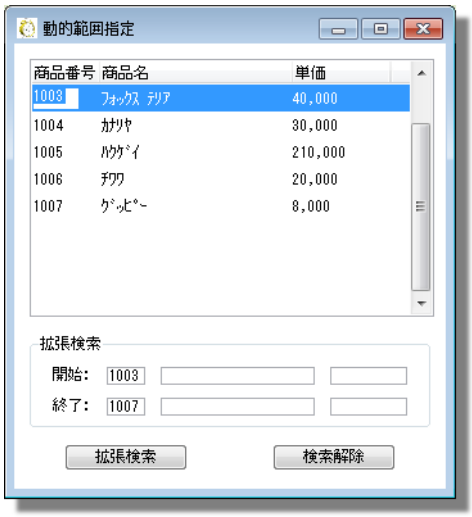
2. 検索方向を設定する



次に、検索する方向を決める必要があります。最初のレコードを取得する場合は**方向**特性を ***D= デフォルト***に設定します。最後のレコードを取得したい場合は、***R= 逆方向***に設定します。

必要な設定はこれだけです。範囲や位置付の設定は必要ありません。

データビューの範囲を動的に定義するには



タスクのデータビューの範囲を動的に定義したい場合があるかもしれません。たとえば、エンドユーザが実行時に「拡張検索」できるようにすることは、一般的なことです。

これを実現する便利な方法は、**RangeAdd()** や **RangeReset()** の関数を使用することです。この例では、ユーザはフォーム上の開始と終了の一組の変数に値を入力することができるようになっています。**RangeAdd()** 関数は、範囲値を設定するために使用されます。**ビュー再表示**が発行され、新しい結果が表示されるようになります。

検査解除ボタンを**クリック**すると、**RangeReset()** 関数が実行され、変数の内容が消去され、**ビュー再表示**が発行されます。

データビューの位置付けを動的に定義するには

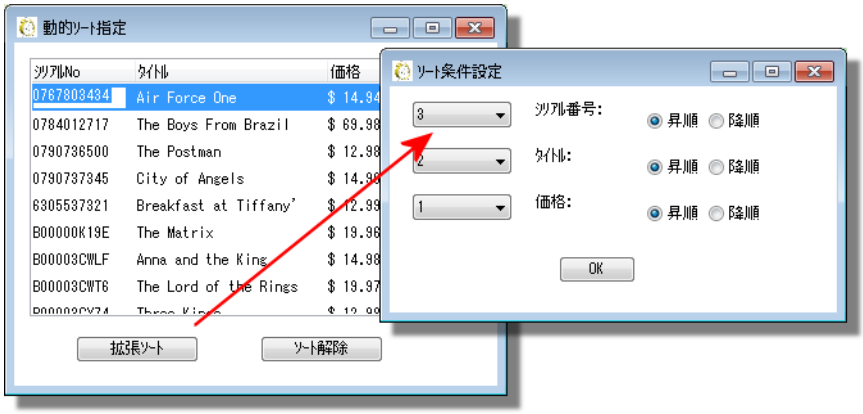
色々な方法で位置付け条件を設定することができます。Magic xpa は、位置付け機能が組み込まれていますが、不慣れなユーザはインデックスとして割り当てられていないフィールドを使用して、非常に遅い検索を行う事があります。簡単に実現でき、さらにユーザが何をもって位置付けを行うかをコントロールすることができるようにする方法として、**LocateAdd()** と **LocateReset()** の 2 つの関数を使用することができます。

この例では、位置付用に拡張検索と検索解除の 2 つのボタンが定義されています。設定ボタンは、ユーザが値を入力できるようにします。次に、**LocateAdd()** 関数が起動され、位置付値として入力された値が設定されます。それから、**ビュー再表示**イベントが発行されビューが再表示されます。

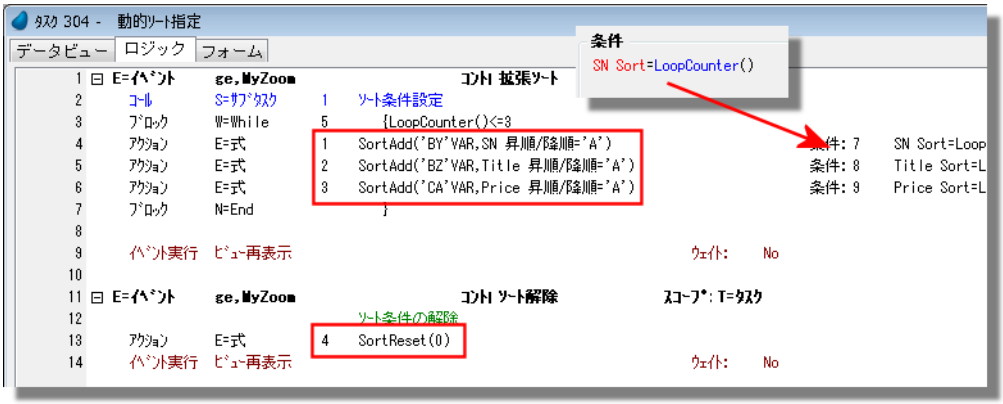
検索解除ボタンは、**LocateReset(0)** を実行し、位置付け処理を解除します。次に、**ビュー再表示**イベントが発行されビューが再表示されます。

次に位置付ボタンをクリックすると、**位置付／次候補**の内部イベントが発行されます。これは、エンドユーザに対するホットキーやメニュー（**オプション→位置付／次候補**、または ***Ctrl+Shift+L***）から実行させることもできます。

データビューのソートを動的に定義するには



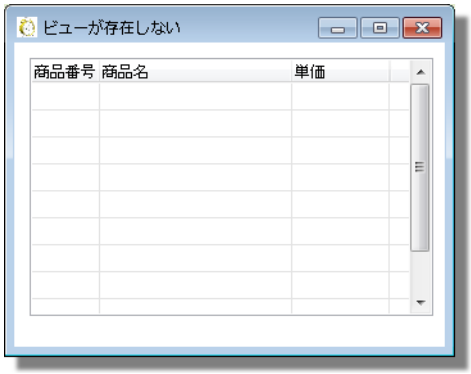
Magic xpa には、ソート機能が組み込まれています。**SortAdd()** や **SortReset()** 関数を使用してこの機能を簡単にカスタマイズすることができます。



1. ソート順序を保持するプログラムを作成します。この例では、各々のフィールドは、1 と 3 の間の数値が割り当てられています。ループ処理は 3 回循環します。そして、**LoopCounter()** の戻り値と一致したフィールドごとに **SortAdd()** を実行します。
2. 降順、または昇順の選択内容を保持するための変数を定義します。
3. ソート処理に追加されるフィールドごとに、**SortAdd()** 関数を実行します。
4. **ビュー再表示**イベントを発行します。

SortReset() 関数を使用して、ソート表示を元に戻します。

レコードがビューに存在しない状態に対応するには

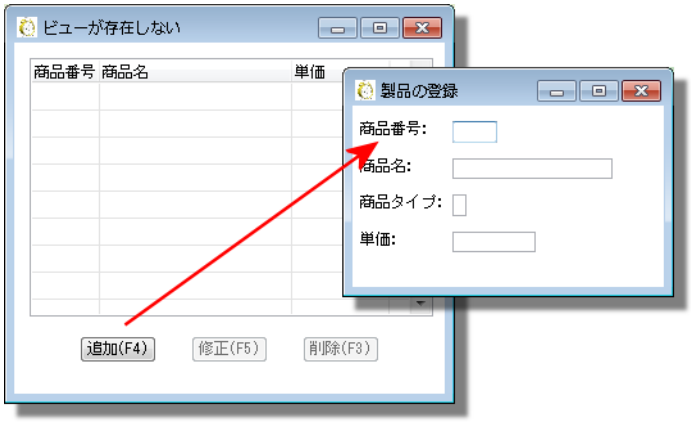


データが無かったり、データが検索条件と一致していない場合、エンドユーザはレコードがデータビューにない状況に陥ります。この状況では、ユーザは空白のウィンドウを表示させることになります。その際、ユーザは次に何をするべきか迷うかもしれません。このセクションは、このような状況に対応するため、色々なオプションについて説明します。

照会または修正？

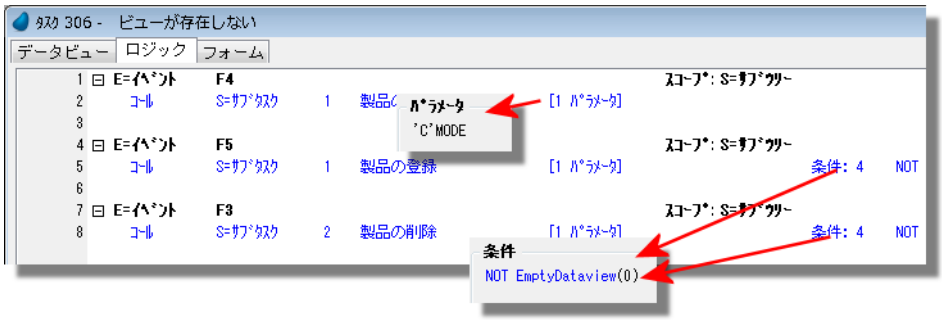
最初の問題は、このウィンドウが単にレコードを一覧表示するか、レコードの表示と修正を可能にするように設計されているかどうかということです。一般的に、一覧表示の画面は、データを簡単に検索できるように設計されていますが、レコードの修正ができるようにするためには、十分な入力フィールドや正しいレコードロック定義が必要です。他方、多くの一覧画面は、多目的に設計されています。空のデータビューの対応は状況によって異なるため、個別に説明します。

一覧のみのウィンドウ



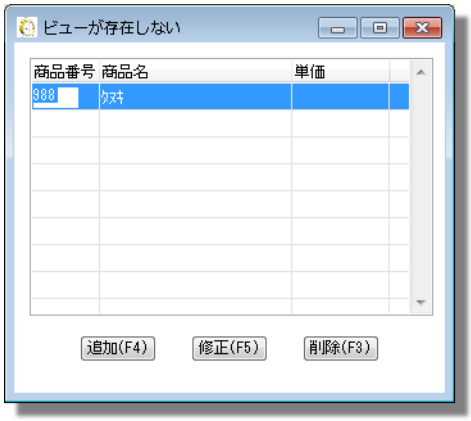
一覧のみのウィンドウでは、通常はレコードをウィンドウ上で編集することはありません。データが一覧表示されますが、読み込み専用モードでデータがオープンされるため、トランザクションはオープンされず、ロックも発生しません。実行できるオプションは制限されます。つまり、**オプション→修正**または、**編集→行作成**のコマンドは、メニュー上無効になります。

このシナリオでレコードを作成するには、Magic xpa に組み込まれている編集コマンドの他のものが必要となります。このような場合、フォーム上に追加や、修正、削除のボタンを配置したり、コンテキストメニューにコマンドを追加することで実現できます。この例で分かるように、同じショートカットキーを使用することができます。



レコードがない場合に修正や削除のボタンを無効にするには、**有効**特性を **Not(EmptyDataview(*0*))** に設定します。これは、有効なレコードが存在しない場合に、追加コマンドのみが利用できる唯一のコマンドであることをエンドユーザに示すことができます。

一覧／修正ウィンドウ



多くのウィンドウは多目的に使用されます。データを一覧表示したり、それを修正するために使用することができます。熟知したユーザは、照会、修正、位置付、範囲の各モード間をキー操作やメニューによって切り替えることができます。プッシュボタンに同じ機能を割り当てることもできます。このようなウィンドウは非常に便利で、簡単にデータを入力することができます。

しかし、ウィンドウに表示されるデータが空の場合、戸惑うことがあるかもしれません。この場合、タスクは照会モードになっています。そして、画面が修正モードになるまで *F4* を押下したり**編集→行作成**を選択しても動作しません。

これに対応する 1 つの方法は、デフォルトのショートカットキーを受け付けるイベントを使用して、このイベントをプッシュボタンに割り当てることです。しかし、この場合は、**レコード修正**の内部イベントを**行作成**イベントの前に発行することです。この例では、タスクが照会モードの場合のみ、イベントが発行されます。

従来の動作

1 つの問題が、古いプログラムの中に存在します。Magic xpa の以前のバージョンでは、空のデータビューの場合、登録モードに自動的に戻ります。エンドユーザは、この動作を使用するか、それを予想したプログラムロジックを作成するかもしれません。

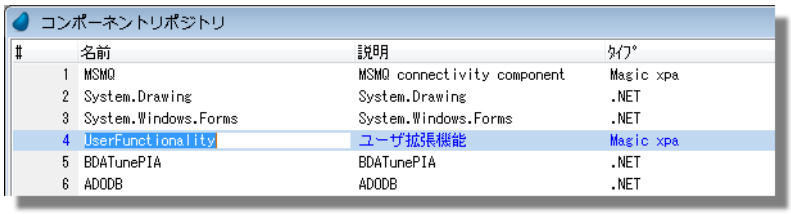
この場合、今までの動作になるようにプログラムを定義するには、**タスク特性**の**空のデータビューを許可**を *No* に設定します。これによって、レコードが存在しない場合、登録モードになります。

エンドユーザ向けに範囲 / 位置付 / ソート機能を強化するには

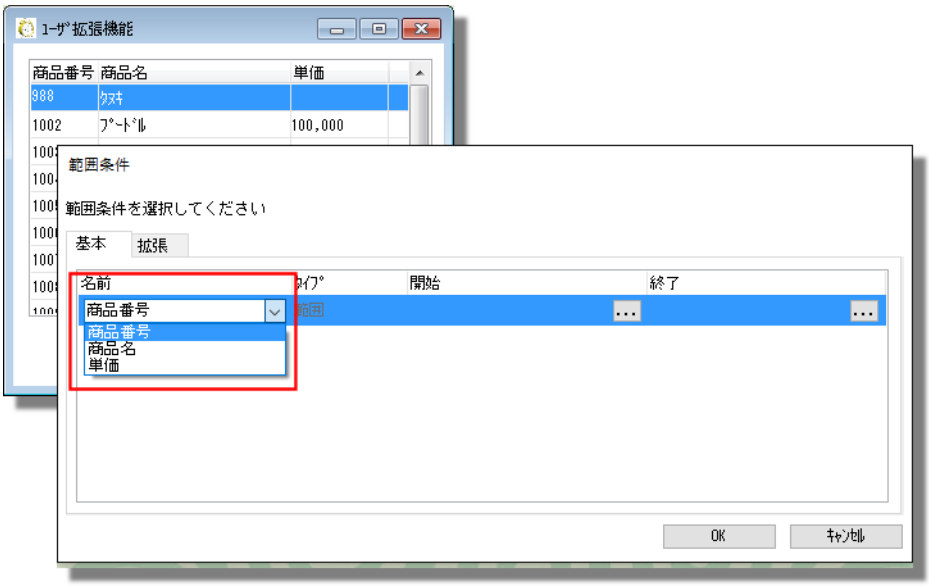
Magic xpa は、範囲指定や位置付、ソート機能が組み込まれています。これらは、プルダウンメニューから呼び出すことができます。また、データ項目や関数を使用することで、範囲指定や位置付、ソートの処理を独自にカスタマイズすることもできます（「データビューの範囲を動的に定義するには」（454 ページ）を参照 s）。

しかし、両端の間のどこでも利用できる汎用的な方法が欲しい場合があります。つまり、様々な画面から呼び出すことができ、使い易くて、制御しやすい方法です。このセクションでは、アプリケーションでこのコンポーネントを使用する方法について説明しています。

1. 最初に、コンポーネントをアプリケーションにロードする必要があります。コンポーネントは、Magic xpa の AddOn ディレクトリにコピーされています。コンポーネントを読み込む方法についてよく知らない場合は、第 15 章 :「コンポーネント」（357 ページ）を参照してください。



2. 一旦コンポーネントが読み込まれると、エンドユーザが**オプション→範囲**、または**オプション→位置付**、**オプション→ソート**を選択すると、自動的に実行されます。これらは、同じように動作するため、ここでは、範囲指定の場合で説明します。



3. これで、エンドユーザが ***Ctrl+R*** を押下するか、**オプション→範囲**を選択すると、組み込み機能とは異なる範囲選択ダイアログが表示されます。エンドユーザは、左のカラムでソートするフィールドを選択することができます。開始値と終了値を入力します。
4. コンポーネントアプリケーションは、オープンソースのため、このプログラムをカスタマイズすることができます。たとえば、ユーザが特定のフィールドで範囲指定を行いたいだけの場合を考えてみます。関係するフィールドだけを選択ボックスに表示するようにすることで、特殊文字をそれらのフィールドに追加することができ、コンポーネントを変更することができます。各ユーザ毎に、名前で特定の範囲条件を保持するように定義し、後日、または、特別なレポートのために使用することができます。
5. エンドユーザーは、**拡張**タブで複雑な式を使用した範囲を定義することができます。このタブには、式の中で使用することができる関数のリストが表示されています。項目リストには、現在のタスクに定義されているデータ項目が表示されています。項目を追加するには、波括弧（{}）内に項目名を入力するか、項目リストでダブルクリックするか、式欄にドラッグ & ドロップしてください。
   'abc' のようにシングルクオーテーションで文字列を囲んで追加することもできます。

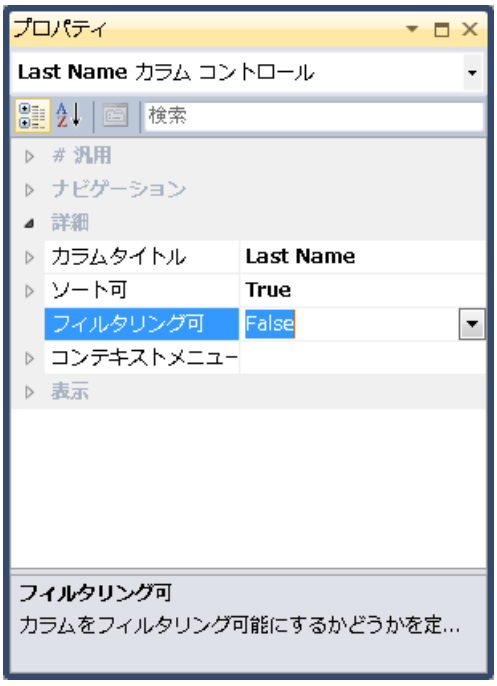
この機能をサポートする Magic プログラムは、簡単で、変更も容易です。このようなプログラムは、「データビューの範囲を動的に定義するには」（454 ページ）や、「データビューの位置付けを動的に定義するには」（455 ページ）、「データビューのソートを動的に定義するには」（456 ページ）で説明された関数を利用しています。

カラムでフィルタリングを設定するには

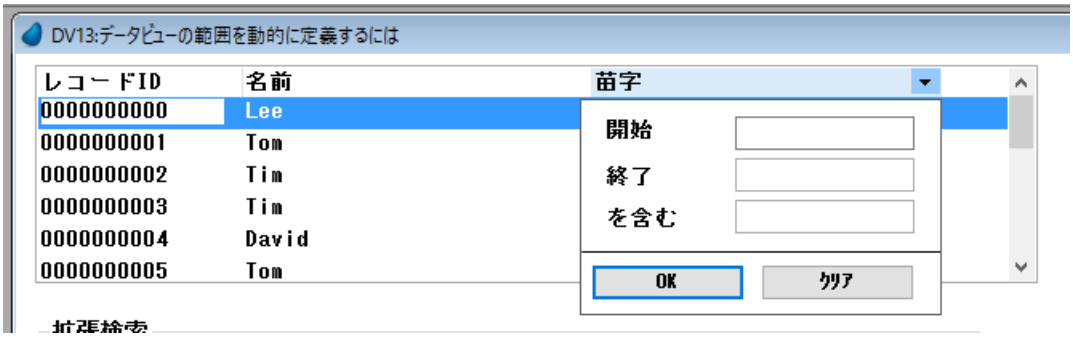
Magic xpa は、テーブルコントロールにフィルタリング機能を提供しています。実行中は、エンドユーザがカラムタイトル上でホバリングすると、カラム上に矢印が表示されます。矢印をクリックすると、カラムにフィルタが可能な場所でダイアログが表示されます。

この機能を有効にするには、エンドユーザ機能コンポーネントをロードする必要があります。エンドユーザ機能コンポーネントは、アプリケーションを新規作成すると、デフォルトでアプリケーションにロードされるように定義されます。このコンポーネントが定義されていない既存のアプリケーションがある場合は、Add_On¥UserFunctionality ディレクトリにあるコンポーネント（UserFunctionality.eci）を読み込んで追加することができます。

このコンポーネントを定義したら、カラムコントロールのフィルタリング可特性を *True* に設定してください。



プログラムを実行し、カラムのタイトル上でホバリングすると矢印が表示されます。矢印をクリックすると、カラムにフィルタ条件を定義することができます。



サポートバージョン :3.1

第 20 章： 式

式エディタ内で式の体裁を整えるには

式の内容によっては記述内容が長く複雑になる場合があります。例えば、ネストされた IF 文を定義する場合は以下のようになります。

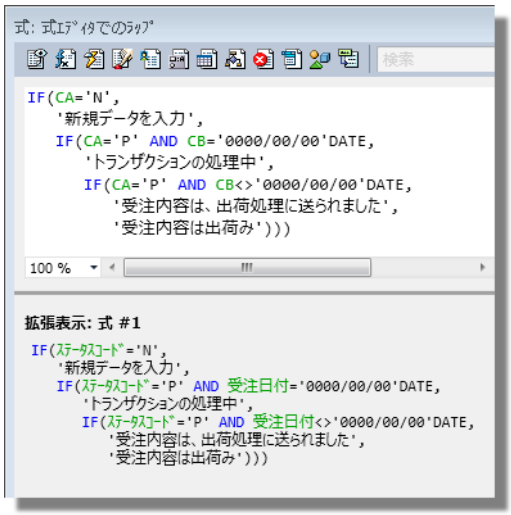
```
IF(I='N', '新規データを入力', IF(I='P' and BN='0000/00/00'DATE, 'トランザクションの処理
中', IF(I='P' and BN<>'000/00/00'DATE,'受注内容は、出荷処理に送られました', '受注内容は
出荷済み')))
```

構文上この式が正しくても、人が理解するには難しいものがあります。この式に改行や空白を追加することで、より理解しやすいものにすることができます。例えば以下のようになります。

```
IF(I='N',
   '新規データを入力',
    IF(I='P' and BN='0000/00/00'DATE,
       'トランザクションの処理中',
       IF(I='P' and BN<>'0000/00/00'DATE,
          '受注内容は、出荷処理に送られました',
          '受注内容は出荷済み')))
```

ここでは、Magic xpa の式エディタ内で式の体裁を整える方法について説明します。

式を改行入力する



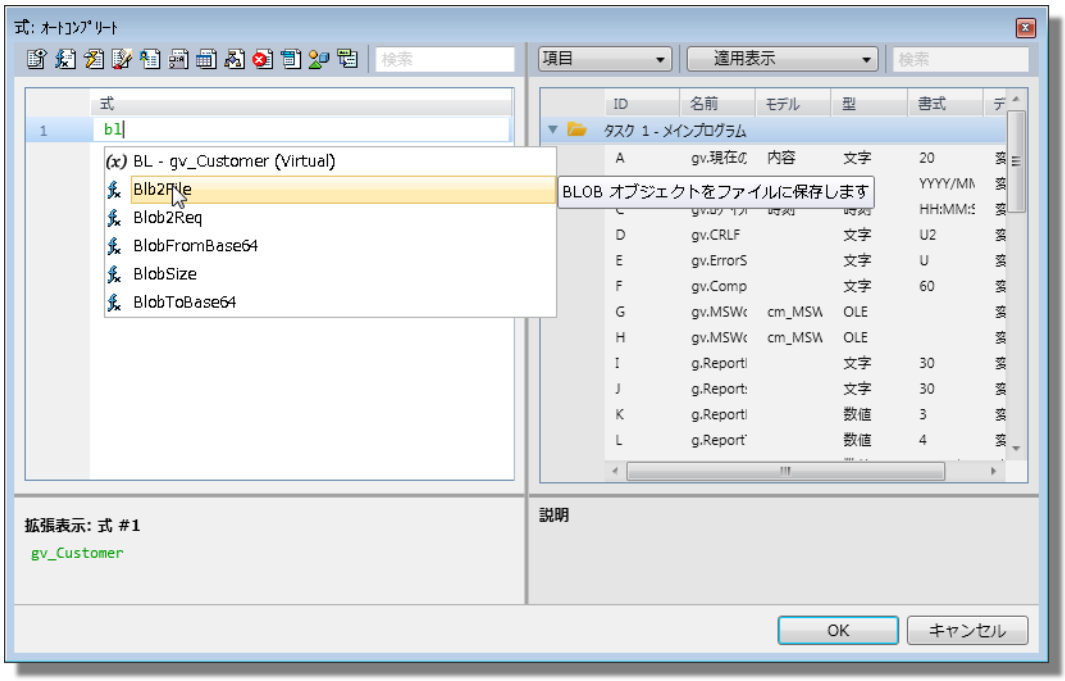
1. 式の定義欄からズーム（*F5*、またはダブルクリック）して式エディタを開きます。*Ctrl+E* を押下することでタスクのどこからでも開くこともできます。
2. *F6*（編集→広域表示）を押下します。この操作で現在パークしている式が拡張表示されます。
3. 広域表示されている場合、*Enter* を押下することで改行表示されます。また、インデントを加えるために空白や *Tab* を入力することもできます。
4. 編集が終了したら、OK ボタンや右上の  ボタンをクリックするか、*Esc* を 2 回押下します。*Enter* は改行として扱われるため、広域モード内では編集内容の確定処理には使用できません。

改行を入力すると式の中に CRLF コードが入力されます。これは Magic xpa の式内であれば問題はありませんが、メッセージボックスなどでこの内容を出力すると改行されて出力されます。この機能は、開発者に対して式の内容を理解しやすくするためのものです。

関数名の入力を簡単に行うには

キー操作を減らすために、Magic xpa には関数名に対してオートコンプリート機能が備わっています。関数名を入力する場合、入力操作のほとんどを Magic xpa に依存させることができます。

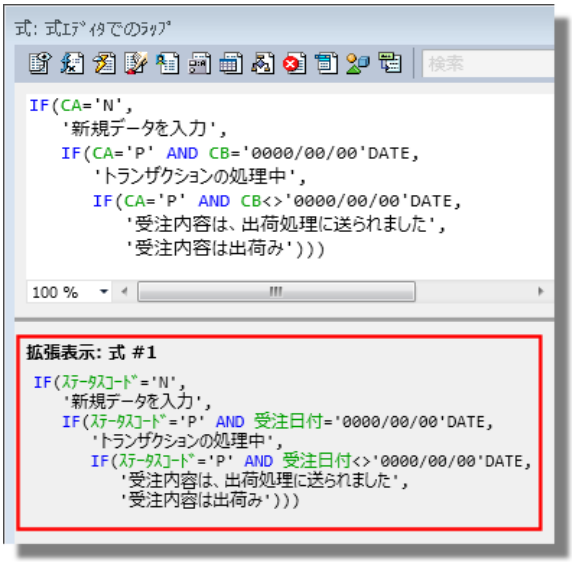
オートコンプリート機能を使用する



1. 入力したい関数名の最初の数文字を入力します。この例では、'bl' が入力されています。大文字小文字は区別しません。
2. 関数リストが表示され、名前の最初の文字が入力した文字と一致する関数に位置付けされます。
3. カーソルを下に動かして使用したい関数に位置付けます。この場合、カーソルキーを使用して動かすこともできますが、関数名を続けて入力することでも自動的にカーソルが移動します。
4. 使用したい関数名にカーソルが位置付いたら、*Tab* を押下します。関数名が最初の括弧とともに追加されます。

注： 入力文字に該当する関数が 1 つだけの場合は、リストボックスが開かず自動的にその関数が挿入されます。

式の拡張表示で色付けされる要素の色を設定するには

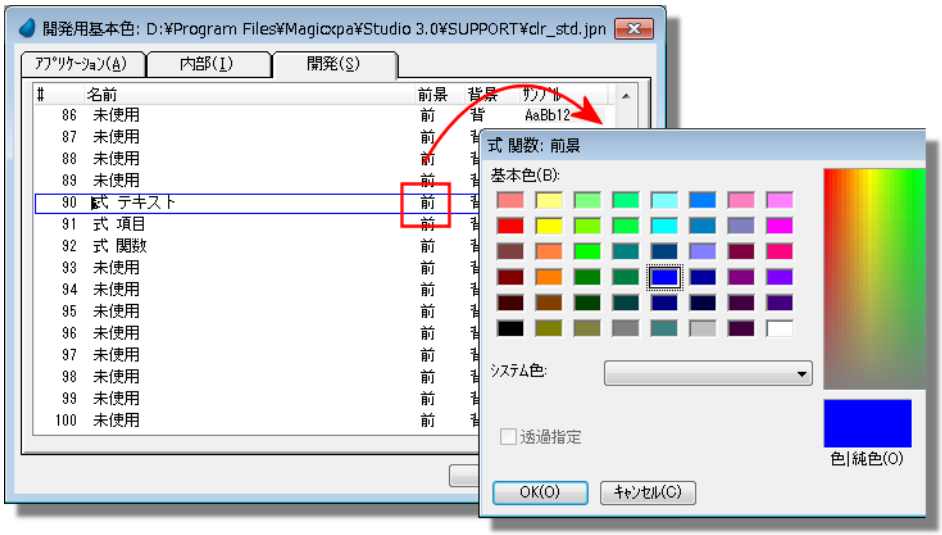


拡張表示領域に表示される式は、色分けされています。ここには以下の３つの色が使用されています。

- テキスト
- 関数名
- 項目名

これらの色は任意に設定することができます。ここでは設定方法について説明します。

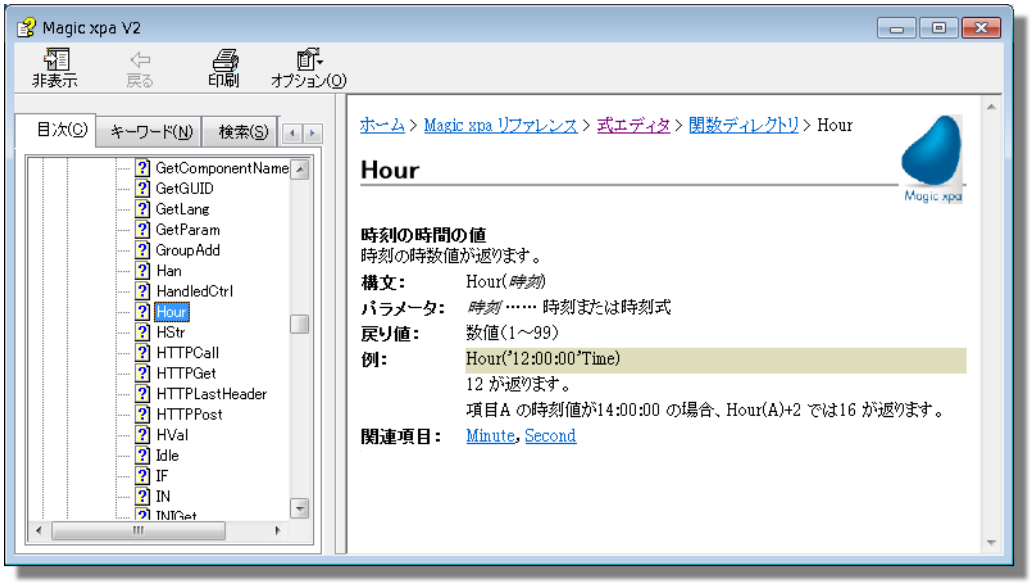
式で使用する色を設定する



1. **基本色**テーブル（**オプション→設定→基本色**）を開きます。
2. **開発**タブを**クリック**します。
3. #90 ～ 92 の行に移動します。各色には、２つのカラム（**前景**と**背景**）があります。
4. 変更したいカラムに移動し、**ズーム**します。**基本色定義**パレットが表示されます。ここで色を指定し、OK を**クリック**します。

基本色定義パレットには、２つの特別な色を指定するオプションがあります。**透過指定**は背景色でのみ指定できます。**システム色**は Windows から色を引き継ぎます。

関数の関連ヘルプを表示させるには



どのような言語で開発する場合でも、全ての関数（メソッド）がどのように動作するものかをすべて覚えていることは難しいものです。Magic xpa では、プログラムを作成中に該当するオブジェクトについて説明されたヘルプトピックを表示する機能があります。ヘルプを表示させるには、２つの方法があります。

任意の場所から関数のヘルプトピックを探す

開発環境ではどこからでも関数のヘルプトピックを表示させることができます。

1. *F1* を押下します。これによって現在のオブジェクトに対応したヘルプトピックが表示されます。
2. キーワードタブをクリックします。
3. 関数の名前を入力します。

入力した関数に対応するトピックが表示されます。また、検索タブを使用することでこの関数名を使用しているトピックを探すことができます。

ヒント : *開発作業中（特に Magic xpa の勉強中）は、ヘルプを開いたままにしておくことで参照にかかる時間が早くなります。*

式エディタでヘルプを使用する

関数が含まれている式から、その関数に対応するヘルプトピックを表示させることができます。

1. 式エディタ（*Ctrl+E*）を開きます。
2. 関数の記述場所にカーソルを置きます。
3. *F1* を押下します。

これで、パークした関数に対するヘルプトピックが表示されます。

関数一覧からヘルプを表示する

関数を入力する便利な機能の１つに関数一覧があります。ここでは簡単にヘルプを表示させることができるだけでなく、関数を選択すると、括弧やパラメータのプレイスホルダが自動的に入り、関数入力が楽になります。

関数の入力方法については、「関数一覧から関数を選択する」（468 ページ）を参照してください。

長い式の編集を容易にするために式の入力行を拡張するには

式の編集中に、*F6* を押下すると編集領域が拡張表示されます。この状態の場合、ウィンドウには現在編集中の式のみ表示され、改行や空白の入力ができるようになります。詳細は、「式を改行入力する」（461 ページ）を参照してください。

文字型項目内で改行データを入力するには

式エディタ内で文字列を編集する場合、テキストエディタのように文字列を入力することができます。つまり、タブや改行および空白を入力することができることを意味しています。

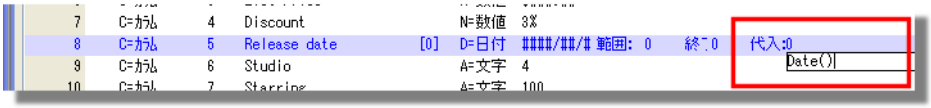
このようなテキスト入力を行うには、最初に広域モードにする必要があります。広域モードでは、小さなテキスト編集ボックスで式を編集することになります。詳細は、「式を改行入力する」（461 ページ）を参照してください。

タスクエディタで直接に式を編集するには

タスクのすべての式は**式エディタ**という 1 つのテーブルに定義されています。**式エディタ**を開くことで既存の式を探して再利用することができます。

しかし、簡単な式であればタスクの各エディタ上で、直接に入力することもできます。この機能を利用することで、短い式であれば入力する工数を削減することができます。

クイック式エディタを使用する



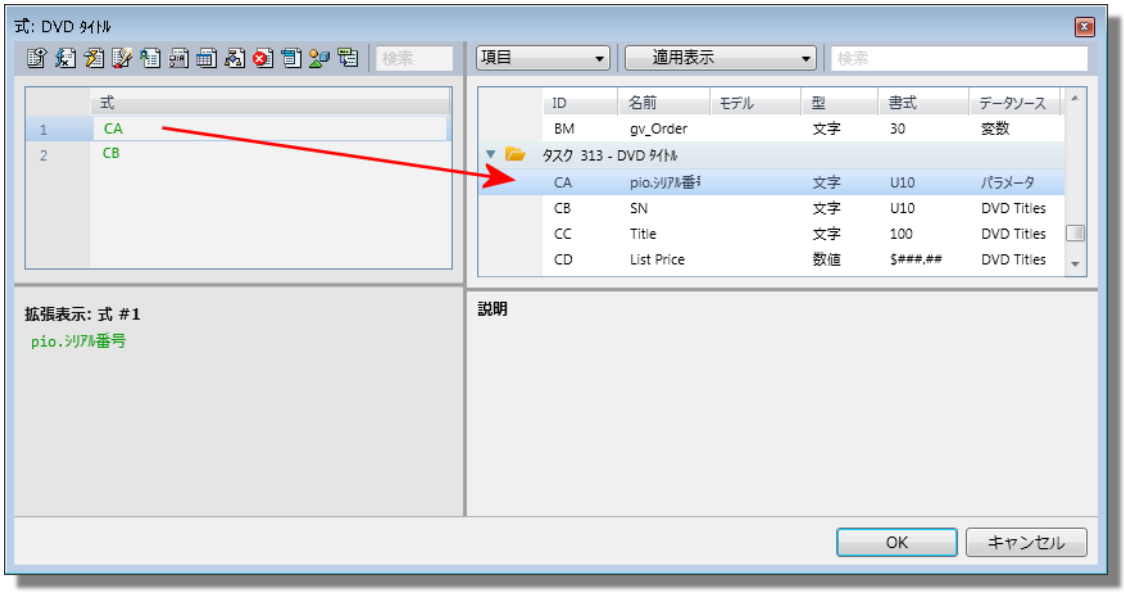
1. **データビューエディタ**や**ロジックエディタ**の**代入**特性や**条件**特性のような、式を定義する必要な場所に移動します。
2. 等号 (=) を入力します。
3. 式の内容を入力します。
4. 入力を終了する場合、*Enter* か *Esc* を押下すると式として保存され、*Ctrl+F2* を押下すると入力内容がキャンセルされます。

入力した式が**式エディタ**に存在していない場合、その式は自動的に追加され、式番号が表示されます。既に存在している場合その式が使用されるようになります。

式エディタから項目一覧に移動するには

式エディタのウィンドウの右側には、タスクがアクセスできる項目が一覧表示されています。ズームすることで項目一覧にカーソルを移動させることができます。

項目一覧からデータ項目を選択する

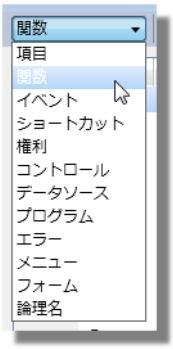


1. 式エディタの、項目を入力したい場所にカーソルを置きます。
2. *F5* を押下します。
3. ダイアログボックスの右側に項目一覧が表示されます。次のどちらかの方法で項目に位置付けます。
   - カーソルキーを使用して、カーソルを上下に移動します。
   - ダイアログボックスの右上隅の検索欄を使用して項目を探すことができます。
   - 項目名の最初の文字を入力することで項目に位置付けます。
4. 設定したい項目が見つかったら、*Enter* を押下するか選択ボタンをクリックすることで、式の中に項目が設定されます。

式エディタから関数一覧を開くには

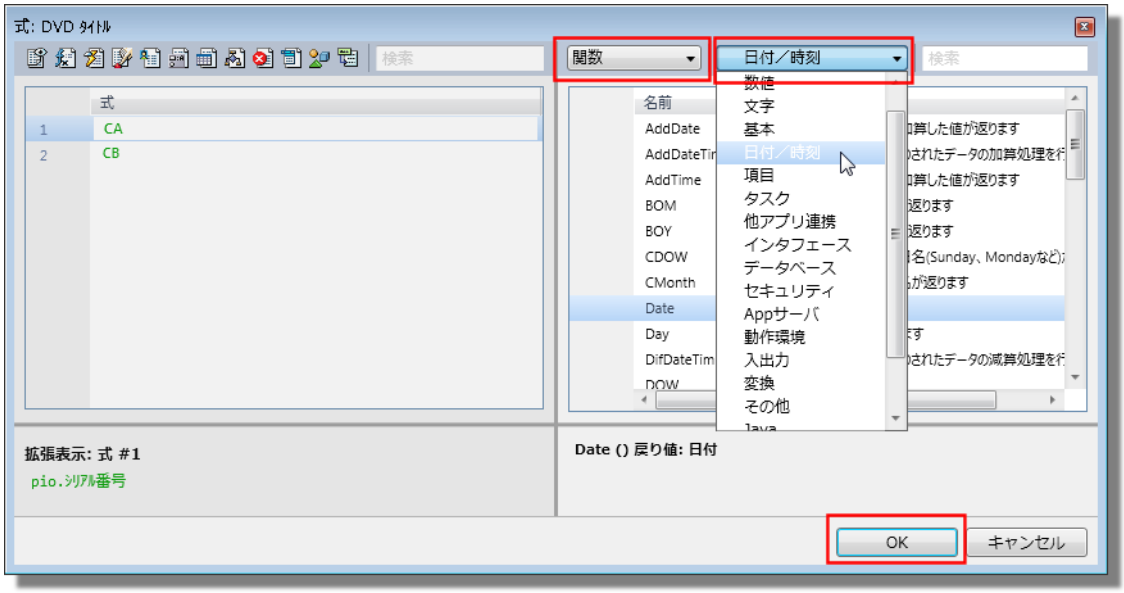
式エディタで編集中に、以下の操作を行うことで関数一覧を表示させることができます。

- *Ctrl+1* の押下
- 式エディタに表示されているツールバーから  アイコンをクリック
- ドロップダウンリストから選択（項目→関数）

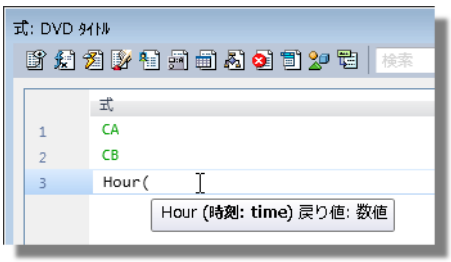


関数一覧を開くと、以下の手順で関数を選択することができます。

関数一覧から関数を選択する



1. 式エディタで、関数を設定したい場所にカーソルを置きます。
2. ドロップダウンリストから関数を選択します。
3. 関数一覧には、グループ毎に関数名が表示されます。使用する関数名が分からない場合は、さらに右側のドロップダウンリストを使用してグループ名で機能を絞って表示させることで選択しやすくなります。
4. 関数名の先頭文字を入力することで関数に位置付けることもできます。
5. 関数名にカーソルがある状態で *F1* を押下すると、その関数に対応するヘルプトピックが表示されます。使用する関数がはっきりしない場合は、ヘルプを表示させて内容を確認しながらで探していく処理を繰り返すことになります。
6. 使用したい関数が見つかったら、*Enter* を押下するか OK ボタンをクリックします。
7. これで、指定された関数が式に追加され、関数のツールチップが表示されます。



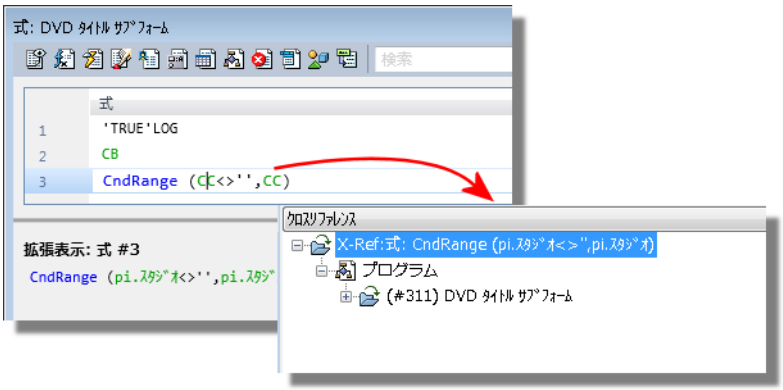
各選択一覧から選択する

関数以外の式で使用するオブジェクトを選択する場合も同じように、ツールバーやドロップダウンリストから一覧を開くことができます。

一覧名	ショートカット	アイコン
関数	*Ctrl+1*	
イベント	*Ctrl+2*	
ショートカット	*Ctrl+3*	
権利	*Ctrl+4*	
コントロール	*Ctrl+5*	
データソース	*Ctrl+6*	
プログラム	*Ctrl+7*	
エラー	*Ctrl+8*	
メニュー	*Ctrl+9*	
フォーム	*Ctrl+0*	
論理名	*Ctrl+ マイナス（-）* ※テンキーのーキーは不可	

式がどこで使用されているかを調べるには

式を変更する必要がある場合、それがどこで使用されるかを知る必要があります。クロスリファレンスユーティリティ（*Ctrl+F*）を使用することで、簡単に検索することができます。



式エディタ上でクロスリファレンスを使用する

1. 調べたい式にカーソルを置きます。
2. *Ctrl+F*（**編集→検索と置換→クロスリファレンス**）を押下します。
3. クロスリファレンスのダイアログボックスが表示されます。OK をクリックすると検索処理が実行されます。式を使用しているオブジェクトはプログラムだけなのでリポジトリ指定は必要ありません。
4. ナビゲータペインのクロスリファレンスの一覧に結果が表示されます。ツリーの最も低いレベルに、式を使用しているタスクが表示されています。このツリーをクリックすると、式を指定している設定欄にカーソルが移動します。

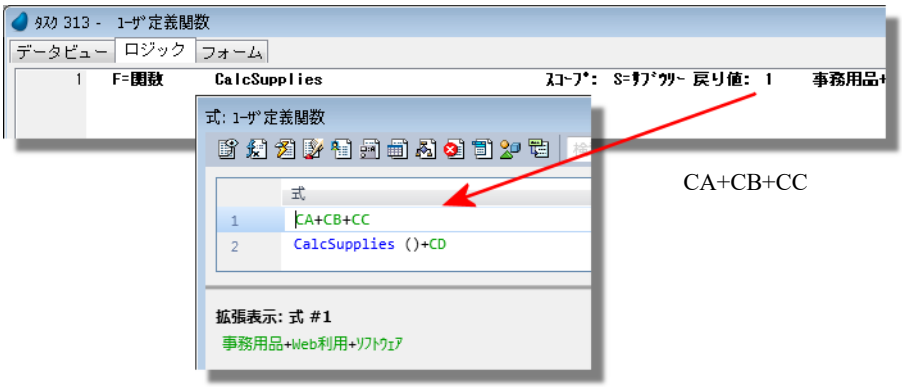
式の中で別の式を再利用するには

式エディタに定義された式は、同じタスク内で何度も使用することができます。同じ式は様々な計算処理（特に項目の初期設定）に使用されることが一般的です。特に、0、'TRUE'LOG、'FALSE'LOG、または **DATE()** などがよく使用されます。

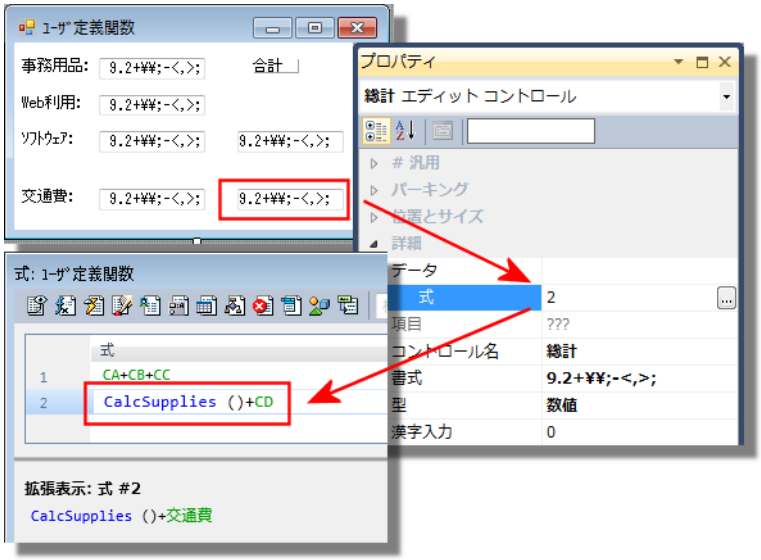
また、簡単なユーザ定義関数を作成したり、**ExpCalc()** 関数を使用することで 1 つの式を別の式の中で利用することができます。**ExpCalc()** は、複雑な計算式を 1 つ作成することで、これを他の式から再利用することを可能にするものです。

ここでは、式を再利用する 2 つの方法の例を紹介します。

ユーザ定義関数を使用する

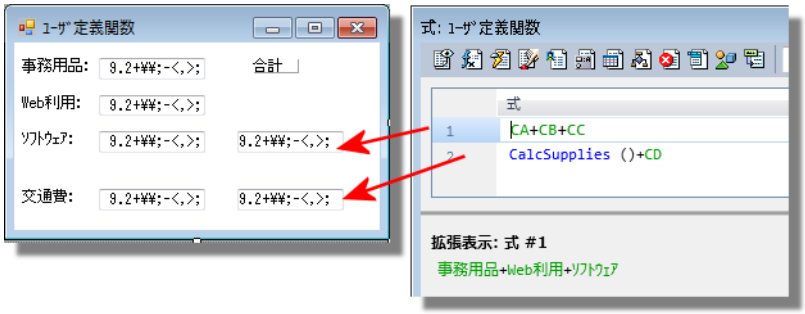


1. 最初に、式の値を返すユーザ定義関数を作成します。この例では、関数名を CalcSupplies として、式 CA+CB+CC を戻すようにしています。



これで、作成されたユーザ定義関数は、式の中で使用することができます。

ExpCalc() 関数を使用する



ExpCalc() は、式の中で別の式の実行結果を使用することを可能にします。この例では、３つの項目が式 **#1** で合計され、小計として表示されています。小計は式 **#2** で使用され、４番目の項目が更に加算されます。

1. **ExpCalc(** を入力します。
2. **EXP** リテラルを使用して、実行させたい式番号を入力します。この例では、式 **#1** を使用するため '1'EXP が入力されています。
3. 関数の括弧を閉じます。

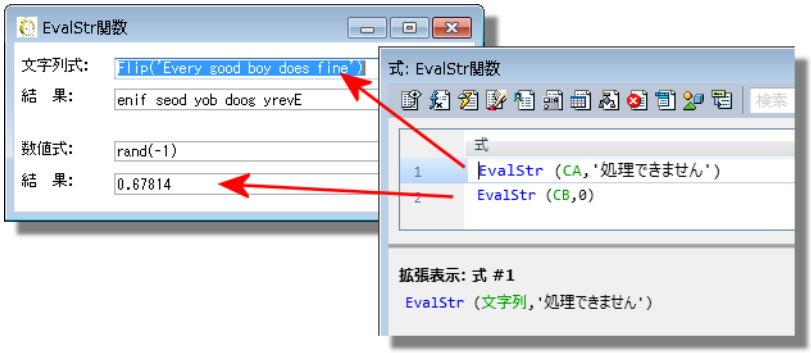
EXP リテラルを使用することで、式の位置が変更されても自動的に追随するようになります。例えば、**#1** の前に別の式を追加した場合、'1'EXP は自動的に '2'EXP に変更されます。

実行時に式の構文の確認と評価を行うには

実行時に Magic xpa の式を構成し、実行させることができます。例えば DB テーブル内にマクロ命令を格納したい場合に便利です。命令は、プログラムによって作成され、別のプログラムで実行させることができます。

EvalStr() 関数を使用することで、このような処理を実現することができます。この関数は、他の Magic 関数を実行させることのできる強力な関数です。この例では、式としてして評価される任意の文字列を入力し、それを実行するものです。

EvalStr 関数を使用する



EvalStr() 関数の構文は以下の通りです。

EvalStr(*expression string*, *default value*)

パラメータ：

- ***expression string*** ：評価される式を表す文字列です。シングルクォーテーションで囲まれた文字列や文字型項目を組み合わせて指定します。
- ***default value*** ：式の中に構文エラーが存在した場合に返される値です。この例では、文字列と評価されなかった場合に「処理できません」という文字列を返します。

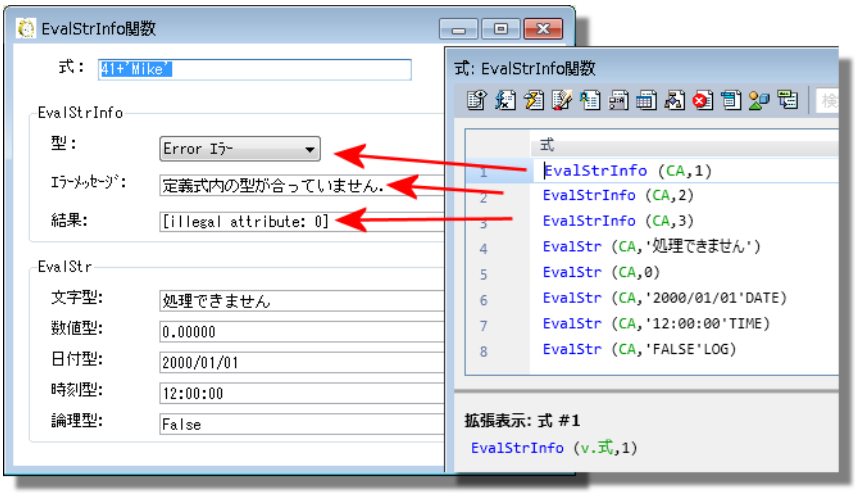
注： この関数を使用する上でいくつか注意することがあります。

第一に、結果として返るデータ型が、期待値と合っていることをあらかじめ確認しておく必要があります。Magic xpa の構文チェックユーティリティは、この式が実行時にどのように評価されるかが分かりません。この例では、２つの文字列をパラメータとして使用し、一方の式では文字列を返し、もう一方では数値を返すようにしています。

また、プログラムが実行されるまでこの文字列はチェックされません。無効な式が入力された場合、デフォルト値が返されますが、どのようなエラーが発生したのか確認できません。

EvalStrInfo() 関数を使用することでこれらの問題を対応することができます。使用方法は以下で説明します。

EvalStrInfo 関数を使用する



EvalStrInfo() 関数の構文は以下の通りです。

EvalStrInfo(*expression string*, *option*)

パラメータ：

- ***string***：評価される式を表す文字列です。文字列と文字型項目を組み合わせて指定します。
- ***option***：返される情報のタイプを表す数値です。以下の値が指定できます。
  - 1 = 戻り値のデータ型
  - 2 = 実行時に発生するエラー内容
  - 3 = 式の内容
    - Option =1 の場合、指定された式が返す値のデータ型を文字で返します（A= 文字型、N= 数値型など）。式にエラーがある場合は、E が返り、データ型が不明の場合は、* が返ります。この例では、結果を表示させる項目を決定するためにこの関数を使用しています。
    - Option=2 の場合、構文エラーをチェックします。この例では、不正なエラーを入力すると「式のデータが正しくありません」というメッセージを表示させるようにしています。
    - Option=3 の場合、渡された式が返ります。エラーにはなりません。

既存の式を複写するには

式エディタに定義されている式を複写したい場合、以下の２つの方法があります。

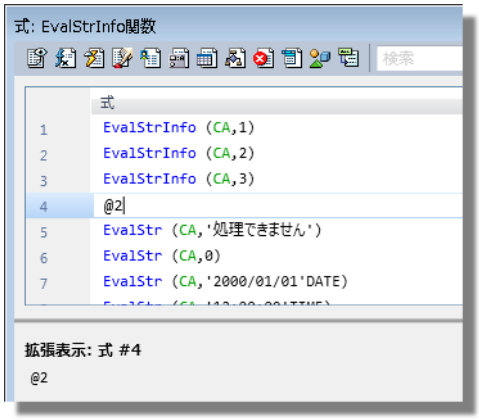
1. Windows で通常使用されるコピー & ペーストのショートカットキー（*Ctrl+C* と *Ctrl+V*）を使用します。
2. 式エディタでは、**@Line** という特別な関数が使用できます。

入力内容の複写処理については、第 1 章 :「入力行を複写するには」（12 ページ）も参照してください。

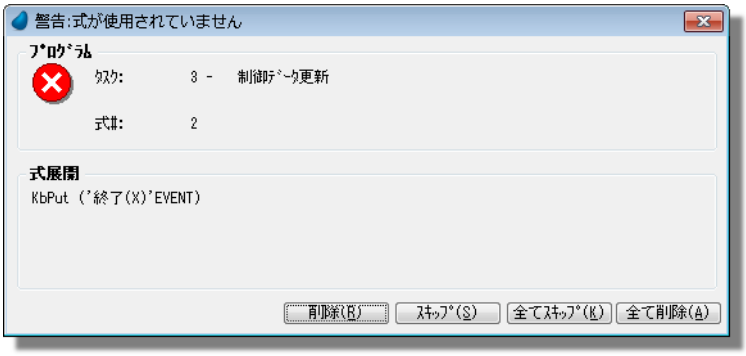
@Line を使用する

1. 式エディタ内で *F4* を押下して 1 行追加します。
2. @ を入力します。
3. 複写したい式番号を入力します。
4. *Tab* を押下します。

指定された行番号の式が複写されます。この例では、式 **#2** の内容が式 **#4** に複写されます。



使用していない式を検索するには



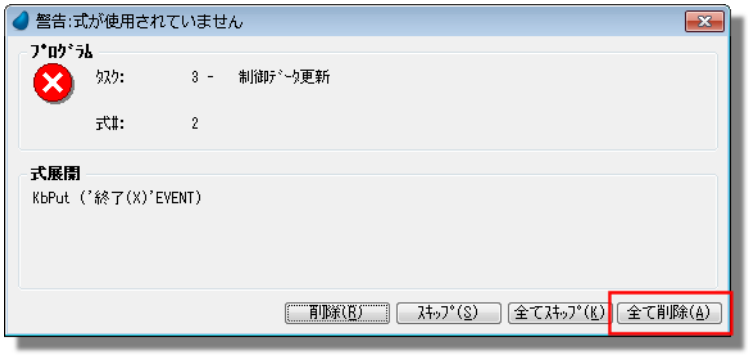
プログラム内に使用していない余分な式が定義されていると保守がしにくくなります。このため、これらを消去することを推奨します。Magic xpa では簡単にこれらを見つけることができます。

未使用の式を検索する

1. **最小チェックレベル**（設定→動作環境→動作設定）を ***W= 警告*** か ***R= 推奨*** に設定します。
2. チェックしたいプログラムに移動します（すべてのプログラムをチェックする場合は、プログラムリポジトリのヘッダ行にカーソルを置きます）。
3. ***F8***（リポジトリ全体をチェックする場合は ***Alt+F8***）を押下します。
4. 未使用の式が見つかるたびに警告ダイアログが表示されます。各警告に対して、開発者が対応することになります。
   - この式を無視して次に進める場合は、スキップをクリックします。
   - これ以降、見つかった未使用の式に対して全てスキップする場合は、全てスキップをクリックします。
   - この式を削除する場合は、削除をクリックします。
   - 式を表示させる場合は、***Esc*** を押下します。この場合は、構文チェックを中断し、式エディタが開いて使用していない式にカーソルが移動します。
   - すべての式を削除する場合は、全て削除をクリックします。

構文チェックの結果は、チェック結果ペインに表示されます。

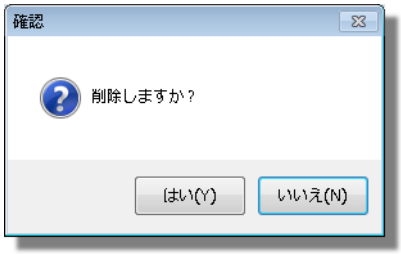
未使用の式をすべて削除するには



未使用の式を削除する場合は、「使用していない式を検索するには」（476 ページ）の説明に従って操作してください。

そして、最初の**警告**ダイアログが表示されたらに、全て削除を**クリック**します。すべての未使用の式が削除されます。

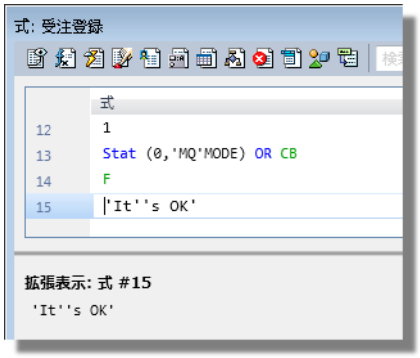
プログラムリポジトリ全体の構文チェック（***Alt+F8***）を行っている場合、右に表示されているような**確認**メッセージが表示されます。はいを**クリック**すると、未使用の式は自動的に削除されます。



式の中で文字列を定義するには

式を入力する際、すべての文字列はシングルクォーテーションで囲む必要があります。この指定をしない場合、文字列は項目のシンボル名（項目番号）か関数名として認識されます。

式の中で文字としてシングルクォーテーションを入力する必要がある場合は、２つのシングルクォーテーションを連続して入力します。



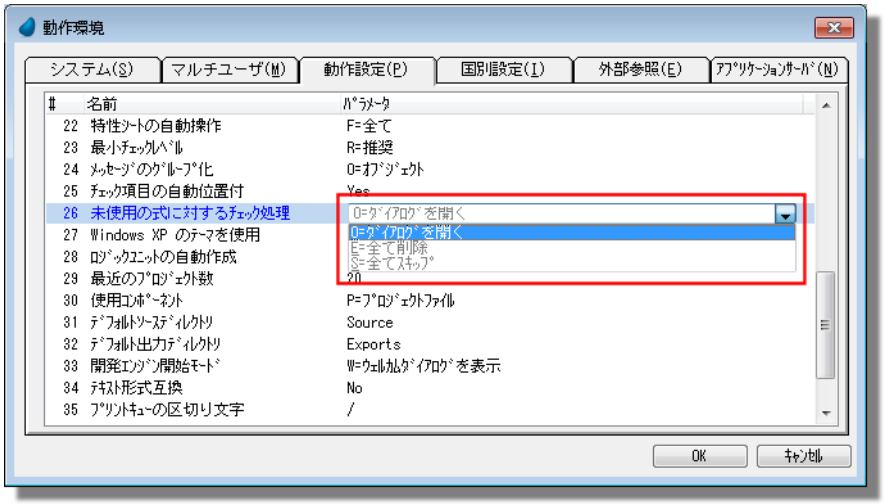
簡単な式が重複して定義されることを防ぐには

なるべく同じ式を共有することで、特に複雑なタスクを開発する場合などは管理が楽になります。例えば、'TRUE'LOG または 0 のような初期設定用の式などがこれに該当します。

式がすでに存在していて、**クイック式エディタ**で式を入力する場合は、このような式を入力する場合に使用できます。式がまだ定義されていない場合のみ、**式エディタ**内に追加されます。

クイック式エディタについては、「タスクエディタで直接に式を編集するには」（466 ページ）を参照してください。

使用していない式を自動的に削除するには



プログラムの構文チェックを行う場合、使用していない式がある旨のメッセージが表示される場合があります。つまり、どこからも参照されない式が式エディタに定義されているということです。使用していない式のチェック動作は、**設定→動作環境→動作設定**の**未使用の式に対するチェック処理**で設定することができます。

ここには、以下の 3 つの設定があります。

- ***ダイアログを開く***……使用していない式を見つけた場合、常にダイアログボックスを開きます。ダイアログボックス上で、他のオプションのどちらかを選択することもできます。
- ***すべてを削除***……ダイアログボックスを開くことなく、使用していない全ての式を自動的に削除します。
- ***すべてスキップ***……ダイアログボックスでも開くことなく、使用していない全ての式を無視します。

作業のスタイルに応じて最も適切なオプションを選択してください。

第 21 章 : レポート

タスクのデータビューを外部ファイルに出力するには

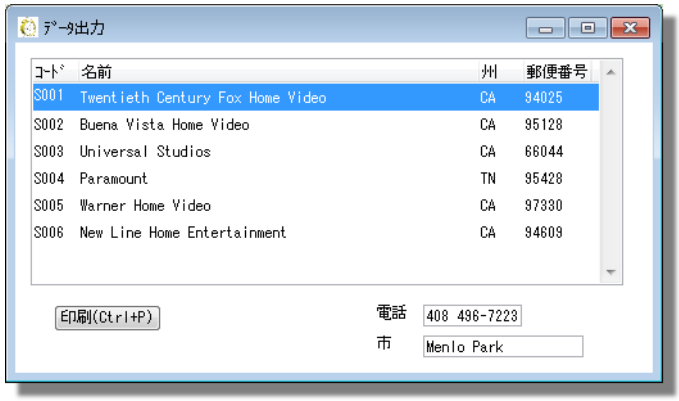
ユーザから最も要求される機能の 1 つに、データを表計算ソフトやワードプロセッサ、または他のソフトウェアで読み込めるようなファイルとして出力することです。現在 Magic xpa には、特別なプログラムを作成することなくこれらの処理を実現する機能が組み込まれています。

実現するには以下の 2 つの方法があります。

- データビュー関数 : プログラムで出力する機能として、**DataViewTo（）**という Magic xpa の内部関数が使用できます。これらの関数は、出力内容と書式などを指定して定義します。「データを HTML ファイルとして出力するには」（489 ページ）を参照してください。
- データ出力ウィザード : エンドユーザに対して、オンラインタスク上からからデータ出力ウィザードを呼び出すことができます。このユーティリティはエンドユーザによって操作することができるものです。「動的にデータ出力させるには」（482 ページ）を参照してください。

どちらの方法も同じような出力オプション（テキスト、CSV、XML、または HTML）があります。

動的にデータ出力させるには

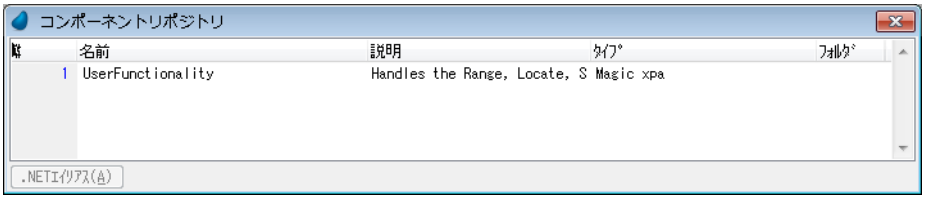


データ出力ウィザードを使用することで、ユーザが表示画面の内容を出力させることができます。このユーティリティはレポート作成プログラムのように動作します。このユーティリティは、ユーザが出力形式を選択することができます。表示されている項目が出力対象となり、表示順に出力されます。

参照： 「データを HTML ファイルとして出力するには」（489 ページ）

エンドユーザ機能コンポーネントを定義する

エンドユーザ機能コンポーネントをアプリケーションに読み込ませる必要があります。コンポーネントは、Magic xpa の AddOn ディレクトリにインストールされています。コンポーネントの読み込みについて、よく知らない場合は、コンポーネントを参照してください。



注： コンポーネントはオープンソースのため、カスタマイズすることができます。たとえば、エンドユーザが特定のフィールドでのみ範囲指定を行うことを要求している場合を想定してください。それらのフィールドだけが選択ボックスが表示されるように、特殊文字をそのフィールドに追加するようにコンポーネントを変更することができます。後で使用したり、特別な帳票で使用するために、各ユーザ名を範囲条件として保存させておくこともできます。

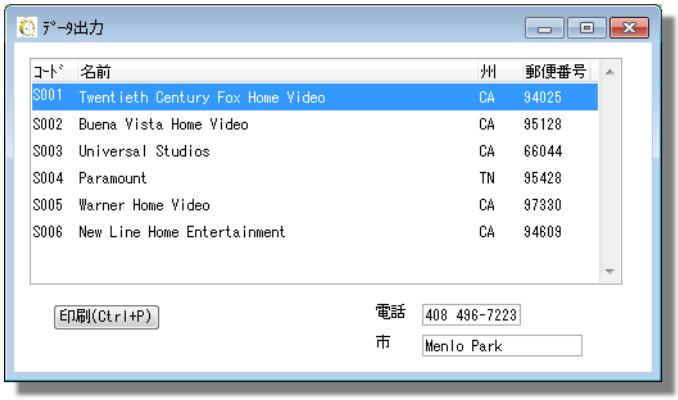
データ表示プログラムを作成する

1. ユーザが参照したいと思われるすべてのレコードとカラムが一覧表示されるタスクを作成します。最も簡単な方法は、**APG**（*Ctrl+G*）を使用して照会プログラムを作成することです（「簡単な参照プログラムを作成するには」（485 ページ）を参照してください）。このタスクの**データ出力**特性（タスク特性→オプションタブ）を *Yes* に設定します。
2. これで、プログラムを実行すると、以下に説明する手順でユーザは、データ出力ウィザード（*Ctrl+P* またはオプション→データ出力）を起動してデータを出力することができます。

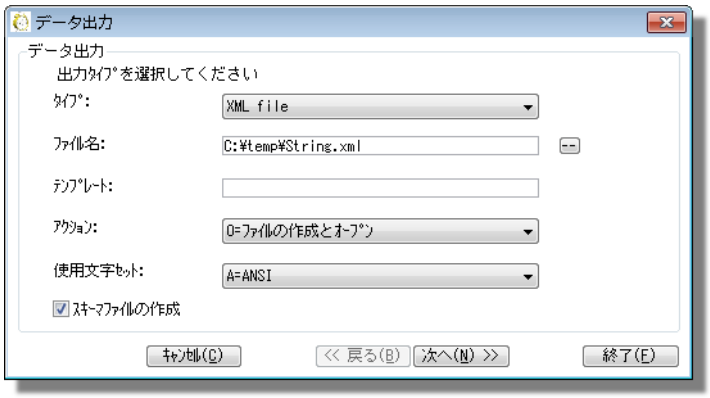
データ出力ウィザードを使用する

また、ユーザはプログラムのデータビューからデータを出力することができます。以下に、その方法について説明しています。

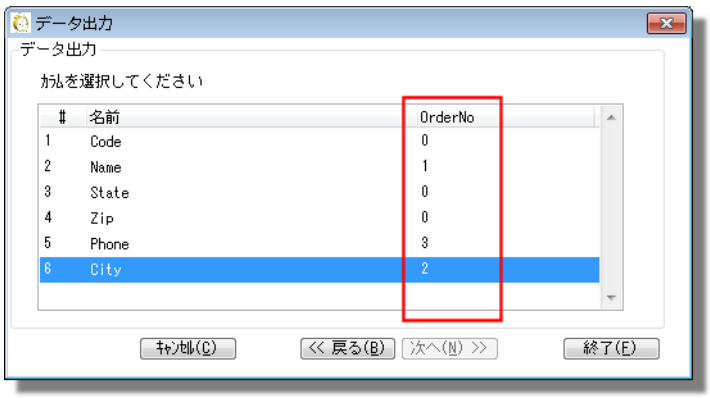
1. プログラムを実行します。
2. レコードが一覧表示されます。ソートや範囲オプションを利用することで必要に応じてレコードのソート順序を変更したり、表示されるレコードの内容を制限したりすることができます。



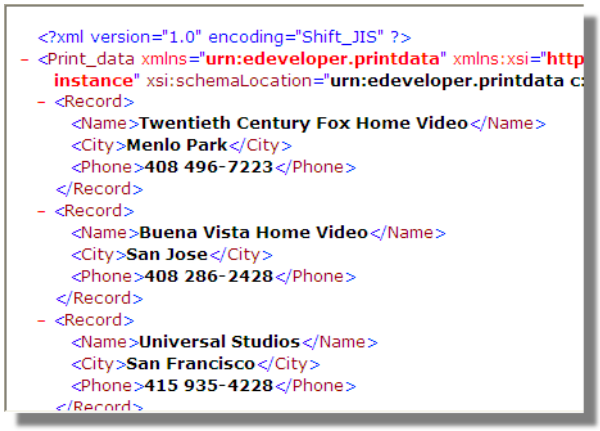
3. *Ctrl+P*（オプション→データ出力）を押下します。データ出力ウィザードのダイアログボックスが表示されます。出力したいデータ形式にもとづいて、オプションを選択し、次へをクリックします。



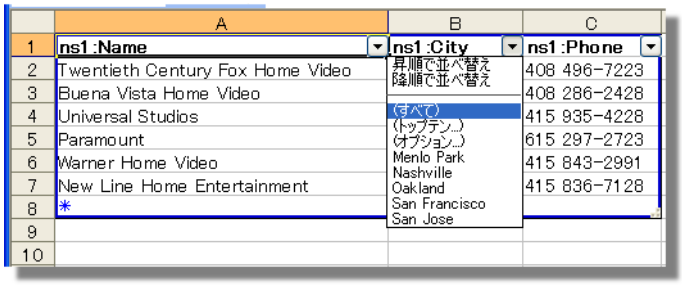
4. 次に、データビュー内のすべてのカラムの一覧がダイアログで表示されます。出力したくないカラムは、**カラムの順番**の値を *0* に設定します。また、**カラムの順番**の番号を付け替えることで出力する順番を変更することができます。
この例では、名前、都市、および電話をこの順番で出力するように設定しています。
次に、終了をクリックします。



5. 指定された内容にもとづきデータがファイルに出力されます。この例では、XML ファイルを選択したためファイルは XML 形式で出力されますが、HTML や CSV で出力することもできます。
作成と表示を選択すると、XML ファイルが PC に割り当てられたアプリケーションによってオープンされます。



ヒント : *データのタイプを XML としてデータを保存することができますが、ファイル名の拡張子を「.xls」に変更してみてください。これにより、Excel XML の形式として扱われ、Excel でオープンしやすくなります。*



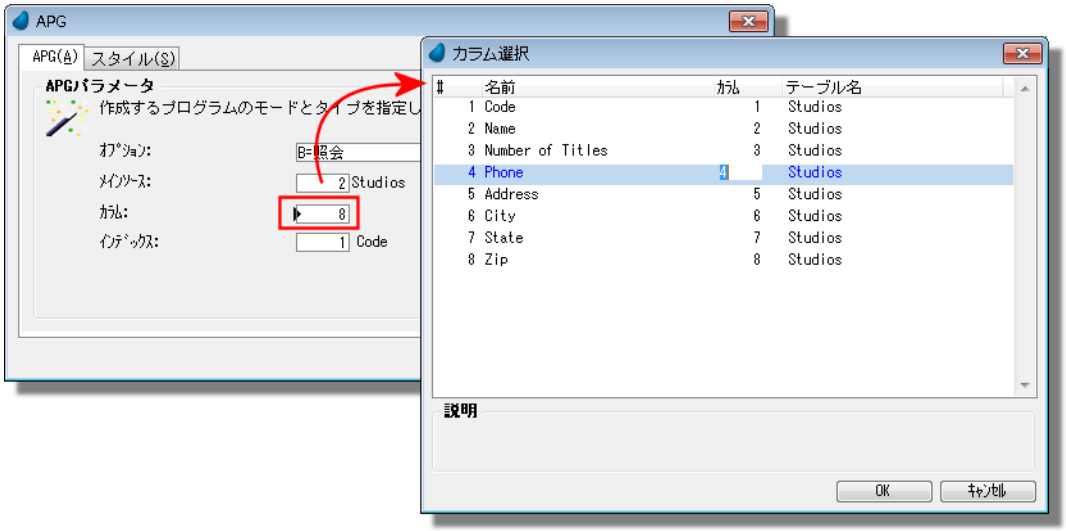
簡単な参照プログラムを作成するには

Magic プログラムは簡単に作成することができます。しかし、**APG**（*Ctrl+G*）ユーティリティを使用することでより簡単にプログラムを作成することができます。

レコードをファイルとして出力する前に、ユーザが範囲やソート機能を利用できるようにすることで、作成されるプログラムは、データビューを表示するプログラムとして利用しやすいものになります。

簡単な参照プログラムを作成する

1. *F4*（編集→行作成）を押下して、プログラムリポジトリに 1 行追加します。
2. *Ctrl+G*（オプション→ APG）を押下します。



3. **APG ダイアログ**が表示されます。以下のオプションを設定します。

   オプション：*Q= 照会*

   メインソース：出力したいデータソース。**ズーム**で一覧を開くことで選択できます。

   カラム：出力するカラムを選択するためにズームします。デフォルトは、すべてのカラムが定義順に指定されています。この例では名前、都市、州、電話、アドレスのみを出力するように設定しています。
4. **OK** を**クリック**します。

プログラムが作成されます。これを実行すると、**メインソース**のすべてのレコードが表示されます。このプログラムは必要に応じて修正できます。フォームを変更したり、範囲を指定したりすることができます。

現在のビューを HTML や XML、CSV 形式でテキスト出力するには

現在のデータビューをテキストファイルに出力する場合、**DataViewTo** 関数を使用することができます。DataView 関数には３つ関数があります。各関数についての説明は、該当するセクションを参照してください。

- **DataViewToHTML()** ：「データを HTML ファイルとして出力するには」（489 ページ）
- **DataViewToXML()** :「データを XML ファイルとして出力するには」（492 ページ）
- **DataViewToText()** :「データをテキストファイルとして出力するには」（487 ページ）と「データを CSV ファイルとして出力するには」（488 ページ）

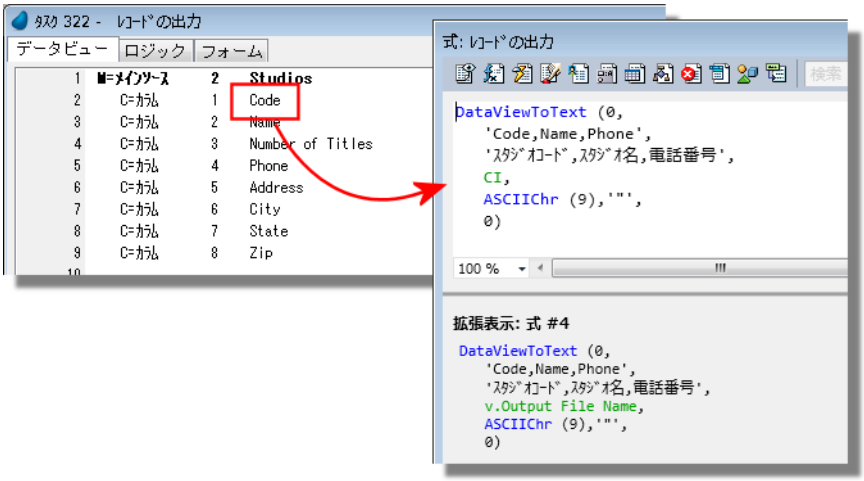
また、プログラムを作成することなく、エンドユーザに対して動的に現在のデータビューを出力させることもできます。「動的にデータ出力させるには」（482 ページ）を参照してください。

参照 : 「タスクのデータビューを外部ファイルに出力するには」（481 ページ）
第 5 章 :「データソースの内容をテキストファイルに出力したり、テキストファイルをデータソースに入力するには」（85 ページ）

データをテキストファイルとして出力するには

DtaViewToText() 関数を使用することで、データビューの内容をテキストファイルとして出力することができます。この関数を使用することで、出力したいレコードを制限させる等、出力内容を制御することができます。

DataViewToText() 関数を使用する



DataViewToText() 関数は、現在のデータビューの内容をテキスト出力します。構文は以下の通りです。

DataViewToText(*Generation*, *VariableList*, *HeaderList*, *OutputFileName*, *DelimiterString*, *StringIdentifier*, *CharSet*)

パラメータ：

- *Generation* ：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- *VariableList* ：出力したい項目名のリスト。タスクのデータビューエディタに定義されている項目名を指定します。複数の項目名を、カンマ区切りで指定します。項目名の間には、空白は入れないでください。また大文字小文字を区別します。
- *HeaderList* ：出力ファイル内の項目タイトルのリスト。この例では、Code 項目に対してスタジオコード というタイトルが指定されています。このパラメータが指定されていない場合、ヘッダは出力されません。@ が指定された場合、*VariableList* で指定された名前がそのまま使用されます。
- *OutputFileName* ：出力ファイル名の名前。この例では、v.Output FileName 項目が指定されています。
- *DelimiterString* ：出力される項目の区切り文字。この例では、タブコードとして **ASCIIChr(9)** が指定されています。
- *StringIdentifier* ：出力文字列の前後に付加する文字。この例では、各文字列をダブルクォーテーションで囲むように指定されています。
- *CharSet* ：使用する文字セットを表す番号。以下の番号が指定できます。
  - 0=ANSI
  - 1=Unicode
  - 2=UTF-8

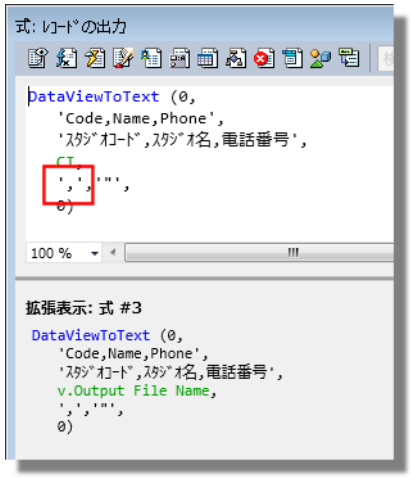
前述の例を実行した場合、以下のような内容のテキストが出力されます。タブコードは表示されませんが、これによってデータが区切られていることを確認できるはずです。

参照： 第 5 章：「データソースの内容をテキストファイルに出力したり、テキストファイルをデータソースに入力するには」（85 ページ）
「動的にデータ出力させるには」（482 ページ）

データを CSV ファイルとして出力するには

DataViewToText() 関数を使用することでデータを CSV ファイルに出力することができます。この例で示すように、必要な設定は、区切り文字としてカンマ指定し、文字列の認識用にダブルクォーテーションを使用することです。

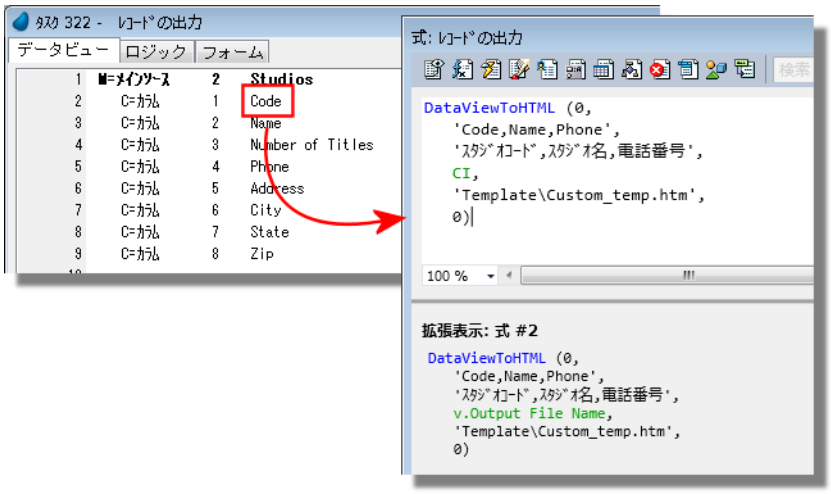
DataViewToText() 関数については、「データをテキストファイルとして出力するには」（487 ページ）を参照してください。



データを HTML ファイルとして出力するには

DataViewToHTML() 関数を使用することで､データビューの内容を HTML 形式のテキストファイルとして出力することができます。この関数を使用することで、出力したいレコードを制限させる等、出力内容を制御することができます。

DataViewToHTML() 関数を使用する



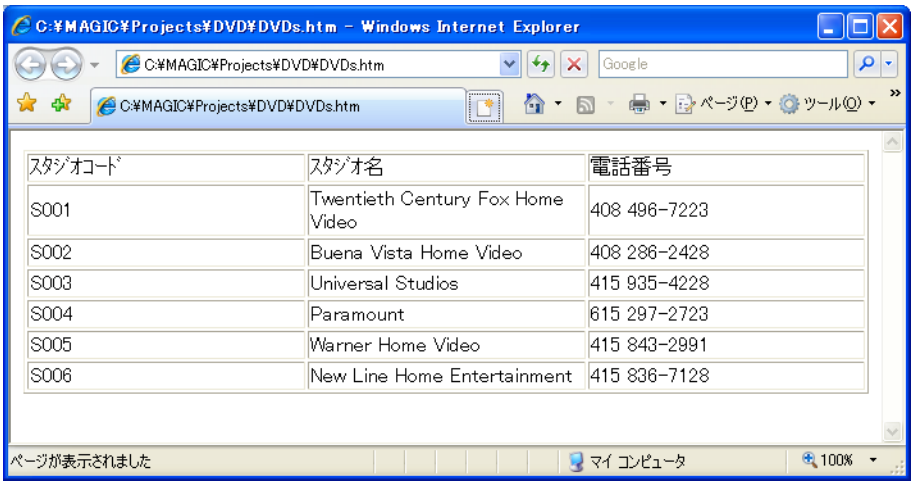
DataViewToHTML() 関数は、現在のデータビューの内容をテキスト出力します。構文は以下の通りです。

DataViewToHTML(*Generation*, *VariableList*, *HeaderList*, *OutputFileName*, *TemplateFile*, *CharSet*)

パラメータ：

- *Generation* ：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- *VariableList* ：出力したい項目名のリスト。タスクのデータビューエディタに定義されている項目名を指定します。複数の項目名を、カンマ区切りで指定します。項目名の間には、空白は入れないでください。また大文字小文字を区別します。
- *HeaderList* ：出力ファイル内の項目タイトルのリスト。この例では、Code 項目に対してスタジオコード というタイトルが指定されています。このパラメータが指定されていない場合、ヘッダは出力されません。@ が指定された場合、*VariableList* で指定された名前がそのまま使用されます。
- *OutputFileName* ：出力ファイル名の名前。この例では、v.Output FileName 項目が指定されています。
- *TemplateFile* ：HTML ファイルの作成時に使用するテンプレートファイル名（オプション）。
- *CharSet* ：使用する文字セットを表す番号。以下の番号が指定できます。
  - 0=ANSI
  - 1=Unicode
  - 2=UTF-8

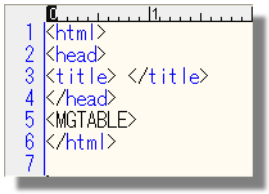
上記のプログラム例を実行すると、以下のような結果になります。



HTML テンプレートファイルを使用する

必要であれば、データを出力する際に、HTML テンプレートファイルを指定することができます。HTML テンプレートファイルは、左側で示されている最小の HTML ファイル形式をもとにカスタマイズすることで利用することができます。**<MGTABLE>** タグは、HTML テーブルのテキストによってデータ出力時に置き換えられます。

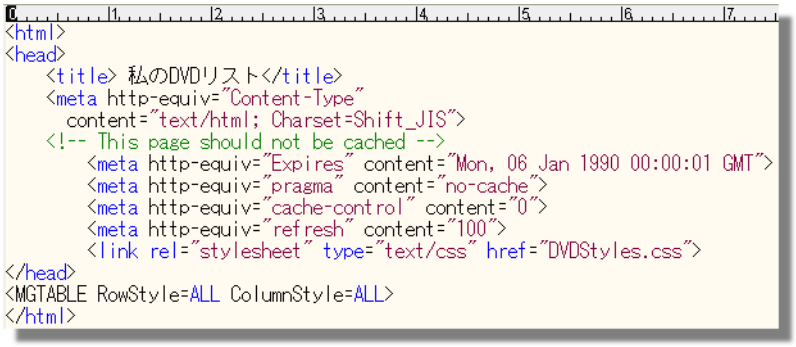
最小のテンプレート



```
1 <html>
2 <head>
3 <title> </title>
4 </head>
5 <MGTABLE>
6 </html>
7
```

これは、テンプレートの最小構成です。**<MGTABLE>** は、実際のテーブルデータによって実行時に置き換えられます。

カスタマイズされたテンプレート



```
<html>
<head>
    <title> 私のDVDリスト</title>
    <meta http-equiv="Content-Type"
      content="text/html; Charset=Shift_JIS">
    <!-- This page should not be cached -->
        <meta http-equiv="Expires" content="Mon, 06 Jan 1990 00:00:01 GMT">
        <meta http-equiv="pragma" content="no-cache">
        <meta http-equiv="cache-control" content="0">
        <meta http-equiv="refresh" content="100">
        <link rel="stylesheet" type="text/css" href="DVDStyles.css">
</head>
<MGTABLE RowStyle=ALL ColumnStyle=ALL>
</html>
```

これは、カスタマイズされたテンプレートです。いくつかの Meta タグが追加され、スタイルシートにリンクされています。

MGTABLE スタイル

<MGTABLE> タグには、RowStyle と ColumnStyle というテンプレート内で使用できる 2 つのスタイルタグがあります。これらは、スタイルタグが作成される表をどのように表示させるかを指定させることができます。

RowStyle:

All	Magic xpa は、各行とタイトルにスタイルを作成します。
EvenAndOdd	Magic xpa は 2 つのスタイル（MG_Even_Row と MG_Odd_Row）を作成します。
Equal	Magic xpa は、すべての行とタイトルに対して同じスタイルを作成します。

Column Style:

All	Magic xpa は、各カラムに対してスタイルを作成します。
Equal	Magic xpa は、すべてのカラムに対して同じスタイルを作成します。

従って、前述のテンプレート例を使用した場合、以下のような HTML テーブルが作成されます。class=tag は、すべて Magic xpa のマージタグです。これらのタグは、テーブルの表示内容をカスタマイズするため使用することができます。

```
</head>
<table border="1" width="100%" class="MG_TABLE">
<THEAD>
 <tr>
  <td width="33%" class="MG_TITLE1">ｽﾀｼﾞｵｺｰﾄﾞ</td>
  <td width="33%" class="MG_TITLE2">ｽﾀｼﾞｵ名</td>
  <td width="33%" class="MG_TITLE3">電話番号</td>
 </tr>
</THEAD>
<TBODY>
<tr class="MG_ROW1">
  <td width="33%" class="MG_DATA1">S001  </td>
  <td width="33%" class="MG_DATA2">Twentieth Century Fox Home Video  </td>
  <td width="33%" class="MG_DATA3">408 496-7223  </td>
</tr>
<tr class="MG_ROW2">
  <td width="33%" class="MG_DATA1">S002  </td>
  <td width="33%" class="MG_DATA2">Buena Vista Home Video  </td>
  <td width="33%" class="MG_DATA3">408 286-2428  </td>
</tr>
<tr class="MG_ROW3">
  <td width="33%" class="MG_DATA1">S003  </td>
  <td width="33%" class="MG_DATA2">Universal Studios  </td>
  <td width="33%" class="MG_DATA3">415 935-4228  </td>
```

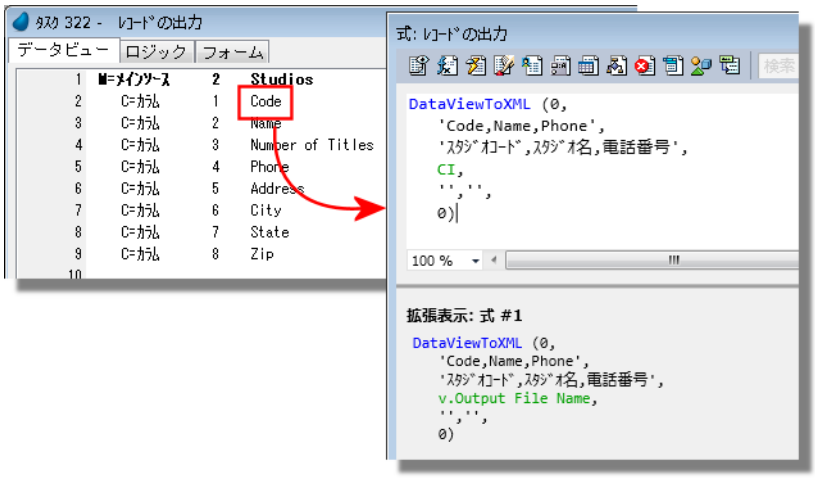
以下は、HTML 形式の DVD 一覧をカスタマイズしたものです。

ｽﾀｼﾞｵｺｰﾄﾞ	ｽﾀｼﾞｵ名	電話番号
S001	Twentieth Century Fox Home Video	408 496-7223
S002	Buena Vista Home Video	408 286-2428
S003	Universal Studios	415 935-4228
S004	Paramount	615 297-2723
S005	Warner Home Video	415 843-2991
S006	New Line Home Entertainment	415 836-7128

データを XML ファイルとして出力するには

DataViewToXML() 関数を使用することで、データビューの内容を XML 形式のテキストファイルとして出力することができます。この関数を使用することで、出力したいレコードを制限させる等、出力内容を制御することができます。

DataViewToXML() 関数を使用する



DataViewToXML() 関数は、現在のデータビューの内容をテキスト出力します。構文は以下の通りです。

DataViewToXML(*Generation*, *VariableList*, *HeaderList*, *OutputFileName*, *SchemaFileName*, *TemplateFileName*, *CharSet*)

パラメータ：

- *Generation* ：タスクの世代番号です。0 が現在のタスク、1 は親タスクになります。
- *VariableList* ：出力したい項目名のリスト。タスクのデータビューエディタに定義されている項目名を指定します。複数の項目名を、カンマ区切りで指定します。項目名の間には、空白は入れないでください。また大文字小文字を区別します。
- *HeaderList* ：出力ファイル内の項目タイトルのリスト。この例では、Code 項目に対してスタジオコード というタイトルが指定されています。このパラメータが指定されていない場合、ヘッダは出力されません。@ が指定された場合、*VariableList* で指定された名前がそのまま使用されます。
- *OutputFileName* ：出力ファイル名の名前。この例では、v.Output FileName 項目が指定されています。
- *SchemaFileName*（オプション）：作成されるスキーマファイル名。空白の場合は、スキーマを作成しません。
- *TemplateFileName*（オプション）：XML ファイルの作成時に使用するテンプレートファイル名。
- *CharSet* ：使用する文字セットを表す番号。以下の番号が指定できます。
  - 0=ANSI
  - 1=Unicode
  - 2=UTF-8

上記のプログラム例を実行すると右のような XML ファイルが作成されます。

```
<?xml version="1.0" encoding="Shift_JIS" ?>
- <Print_data>
  + <Record>
  - <Record>
      <ｽﾀｼﾞｵｺｰﾄﾞ>S002</ｽﾀｼﾞｵｺｰﾄﾞ>
      <ｽﾀｼﾞｵ名>Buena Vista Home Video</ｽﾀｼﾞｵ名>
      <電話番号>408 286-2428</電話番号>
    </Record>
  - <Record>
      <ｽﾀｼﾞｵｺｰﾄﾞ>S003</ｽﾀｼﾞｵｺｰﾄﾞ>
      <ｽﾀｼﾞｵ名>Universal Studios</ｽﾀｼﾞｵ名>
      <電話番号>415 935-4228</電話番号>
    </Record>
  - <Record>
      <ｽﾀｼﾞｵｺｰﾄﾞ>S004</ｽﾀｼﾞｵｺｰﾄﾞ>
      <ｽﾀｼﾞｵ名>Paramount</ｽﾀｼﾞｵ名>
      <電話番号>615 297-2723</電話番号>
    </Record>
```

スキーマファイルを作成する

DataViewToXML() 関数でスキーマファイル名を指定すると、スキーマファイルが生成されます。

XML テンプレートファイルを使用する

必要であれば、XML をフォーマットするために XML テンプレートファイルを指定することができます。テンプレートは、<Print_data> タグと </Print_data> タグの間にテキストを記述しない XML ファイルです。Magic xpa は、このタグの間に <Record> タグを使用してレコードデータをマージします。テンプレートファイルに XSL スタイルシートを指定することにより表示内容をカスタマイズすることができます。

帳票を作成するには

データ出力ウィザード（「タスクのデータビューを外部ファイルに出力するには」（481 ページ）を参照）を使用することで、データを表計算ソフトや、レポートツール用に簡単に出力することができます。この機能は、エンドユーザに対してデータを動的に処理させたい場合に便利です。

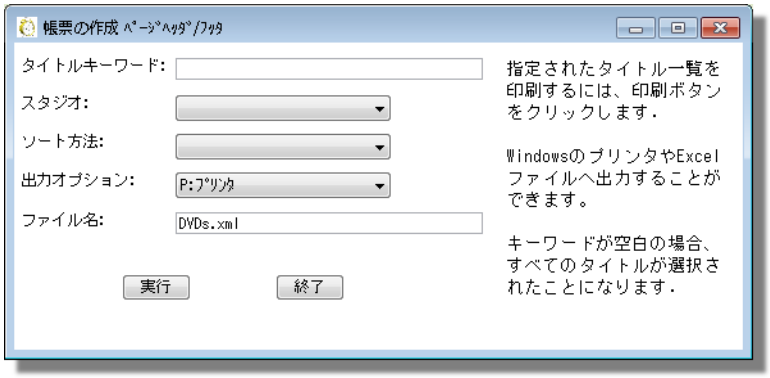
また、Magic xpa にはこの他にもデータ出力用の機能が組み込まれています。毎月の請求書発行など、定型的な書式のデータを作成する場合、Magic xpa のデータ出力機能を使用してたプログラムによって出力した方が便利です。

ここでは、簡単な帳票を作成する基本的な手順について説明します。同じような手続で、複雑な帳票も作成できますが、出力内容が増えたり以下で説明するような高度な機能を利用する必要が出てきます。

- 「帳票のテーブルに対して見出しを繰り返し出力させるには」（512 ページ）
- 「ReportsMagic 用フラットファイルを出力には」（513 ページ）
- 「ページヘッダ / フッタの情報を定義するには」（500 ページ）
- 「帳票のブレイクレベルを作成するには」（507 ページ）
- 「帳票に複数行のコントロールを定義するには」（510 ページ）

Magic xpa で帳票を作成する

1. “ 起動画面 ” を作成する

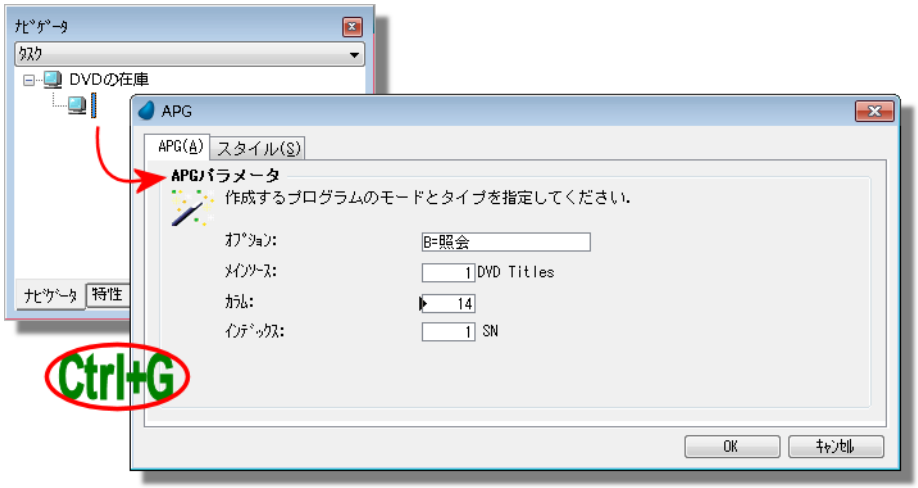


必ずしも必要ではありませんが、帳票のタイトルや何のための帳票か、またユーザに出力先を選択できるような起動画面を作った方が親切です。これは、簡単なオンラインタスクで実現できます（オンラインタスクの作成についての詳細は、第 5 章 :「簡単なプログラムを作成するには」（74 ページ）を参照してください）。

同じ画面を利用して、GUI 形式フォームでの印刷や Excel へのデータ出力を行うことができます。印刷プレビューやソート順序をユーザに選択させることもできます。

このような起動画面のデザインの標準化を検討することは有益です。標準化によってユーザ教育が容易となり、新規に作成する際も既存のタスクをコピーして利用することができます。

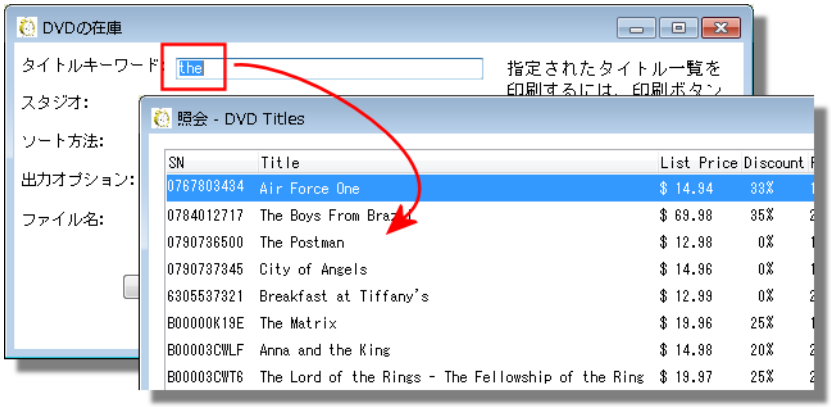
2. 簡単なデータ照会プログラムを作成する



APG を利用して簡単な照会プログラムを作成します（「簡単な参照プログラムを作成するには」（485 ページ）を参照）。

このタスクを呼び出す起動ボタンを設定します。起動ボタンが正しく動作した場合、作成された照会プログラムが表示されます。

3. ソート順序と範囲を確認する



帳票出力プログラムを作成する際に考慮すべき点の 1 つに、範囲とソート処理が正しく動作するかどうかを確認することです。このような場合、オンラインタスクで確認した方が簡単です。このためここでは照会プログラムを最初に作成しています。

複数のソートや出力オプションがある場合、選択されたオプションにもとづいて異なる出力タスクを起動する必要があるかもしれません。しかし、1 つの出力プログラムを作成したら、次のバージョンのためにテンプレートとしてこのプログラムをコピーして使用することができます。

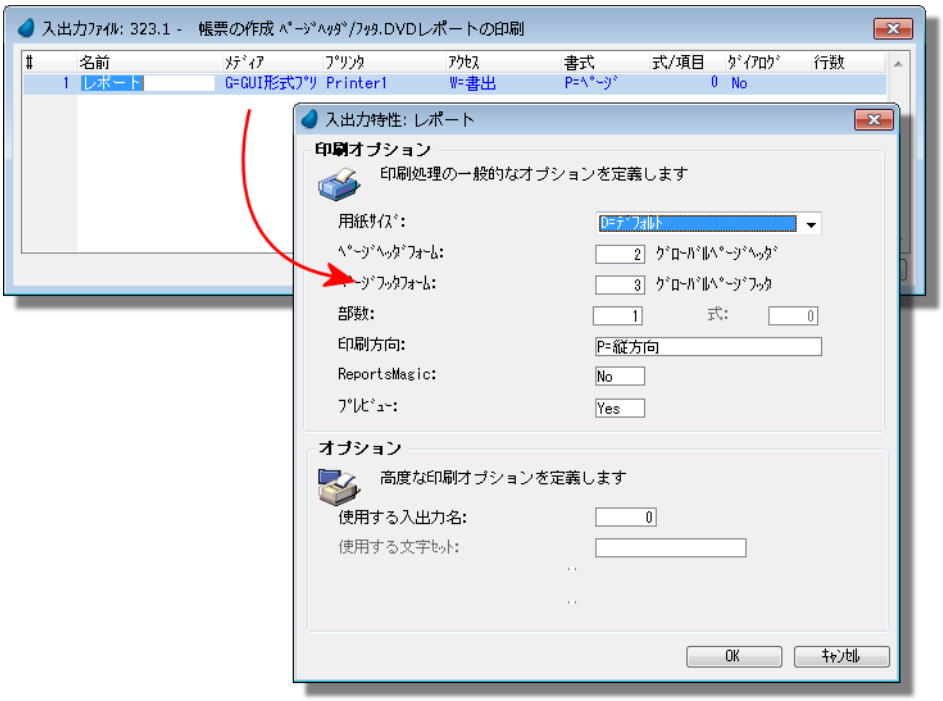
4. バッチタスクを定義する

デバッグを終了したら、タスク特性→汎用タブで**タスクタイプ**特性を（オンラインから）*B= バッチ*に変更することにより、サブタスクをバッチタスクに変更します。

また、レコードの更新がない場合は、タスク特性→データで**トランザクションモード**特性が *P= 物理*で、**トランザクション開始**特性が、*N= なし*（トランザクションはタスクの処理速度を劣化させるため）になっていることを確認します。同様に、メインソースの**アクセス**特性が *R= 読込*になっていることも確認します。

最後に、表示されるフォームをテーブル形式ではなく簡単なデータ表示になるように変更します。作成されたテーブルを削除し、ユーザに対して、動作状況が確認できるようなメッセージを表示するように、1 ～ 2 つのカラムのみ配置するようにします。フォーム上に様々な処理を実行させないようにすることが、処理速度を低下させないために必要です。

5. 入出力ファイルを設定する



次に、データ出力用の入出力ファイルを設定する必要があります。入出力ファイルを定義することでデータをどのように出力するかが決定されます。

1. データ出力用のサブタスクに移動します。
2. *Ctrl+I*（タスク環境→入出ファイル）を押下して入出力ファイルテーブルを開きます。
3. *F4* を押下して、入出力ファイルテーブルに 1 行追加します。
4. **メディア**カラムで *G=GUI 形式プリンタ*を選択します。
5. 入出力特性（*Alt+Enter*）を開きます。**プレビュー**特性を *Yes* に設定します。印刷プレビューは、デバッグの際に有効です。

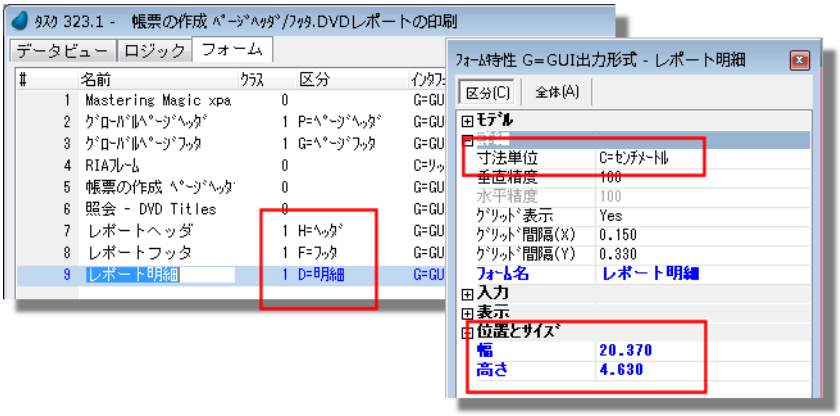
上記の図に表示されているように、入出力特性には様々なオプションがあります。各特性欄にカーソルを置いて *F1* を押下すると該当するヘルプファイルが表示されます。

テストを行う場合は、印刷プレビューは非常に有益です。しかし、実際に作成するプログラムでは、**印刷**ダイアログを使用して直接プリンタに出力しないようにする場合があるかもしれません。**印刷**ダイアログを表示させたいのであれば、**ダイアログ**カラムを *Yes* に設定します。

また、請求書のような定型の帳票を印刷する場合は、出力先を固定にする必要があります。その場合は、**ダイアログ**カラムを *No* に設定します。この場合は、**プレビュー**特性も *No* に設定します。

これらのオプションのほとんどは、式で指定することもできます。

フォームを作成する



次に、フォームを作成します。基本的なステップは以下の通りです。

1. **フォーム**タブに移動します。
2. *F4* を押下して 1 行追加します。
3. フォームに**名前**を入力し、**クラス**カラムに 1 以上の値を入力します。
4. フォームの機能にもとづいて**区分**カラムを入力します。この例では、***ページヘッダ***と***ページフッタ***、および***明細***を 1 つずつ定義します。
5. **インタフェースタイプ**を *G=GUI 出力*に設定します。

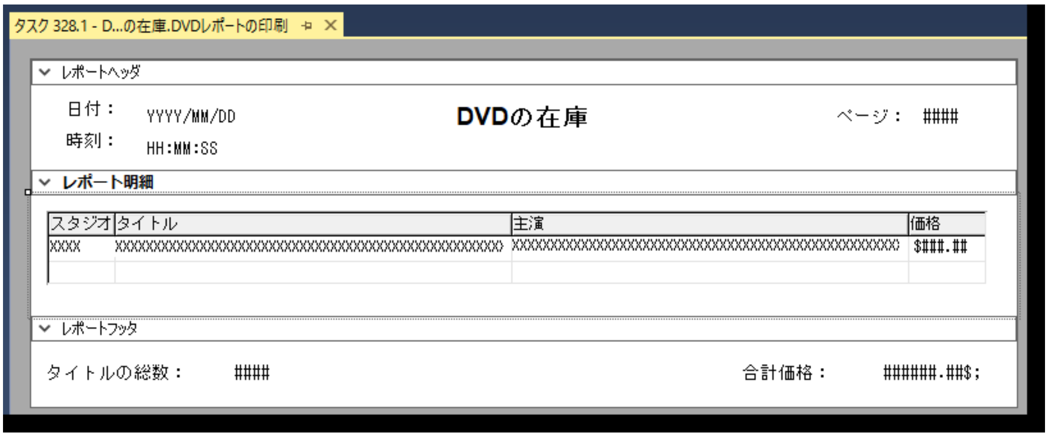
フォーム特性で、**寸法単位**特性は *I= インチ*か *C= センチ*に設定してください。*D= ダイアログ*に設定した場合、フォームは自動的にリサイズされます。**幅**特性は、用紙のサイズに合わせます。**高さ**特性は、各フォーム毎に異なり、マウスを使用することで簡単に変更できます。

注：　インターナショナル版（ダイアログ）と日本語版（センチ）の Magic xpa は、寸法単位のデフォルト値が異なります。このため、日本語版の Magic xpa Studio で開発したアプリケーションを英語版で実行した場合、フォームのサイズが変わる可能性があります。

帳票フォームに対応したモデルを作成して利用すると便利です。この場合、幅やフォント、寸法単位を自動的に一貫性を持って設定することができます。

ヒント :編集するフォームのサイズが非常に大きい場合、フォームの編集に入る前に、手動でサイズを設定することもできます。フォーム特性の幅特性と高さ特性に妥当なサイズを入力することで変更できます。

フォームを編集する



クラス特性が ***1*** のフォームのどれかにズームすると、同じクラスの３つのフォームがすべて表示されます。フォームエディタ上の順番にかかわらず、ヘッダは先頭に、フッタは最後に表示されます。1 つまたは 2 つのフォームを同時に編集する方が便利な場合があります。このような場合は、**クラス**特性を一時的に他の値に設定します。このようにすることで、これらのフォームを個別に編集することができます。

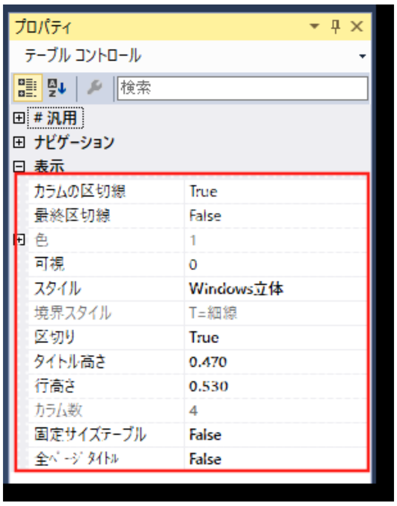
フォームを開くと、オンラインのフォームと同じように項目を選択してフォームに配置することでフォームの編集を行うことができます。項目の様々なタイプに対応した標準的なモデルをあらかじめ定義しておくと便利です。

明細フォームは基本的にテーブルです。帳票に境界線を出力させないようにする場合は、右に示すようにテーブルのコントロール特性を設定します。他のテーブルに対しても同じように定義します。フォーム上にテーブルコントロールを配置し、その上に項目を配置します。

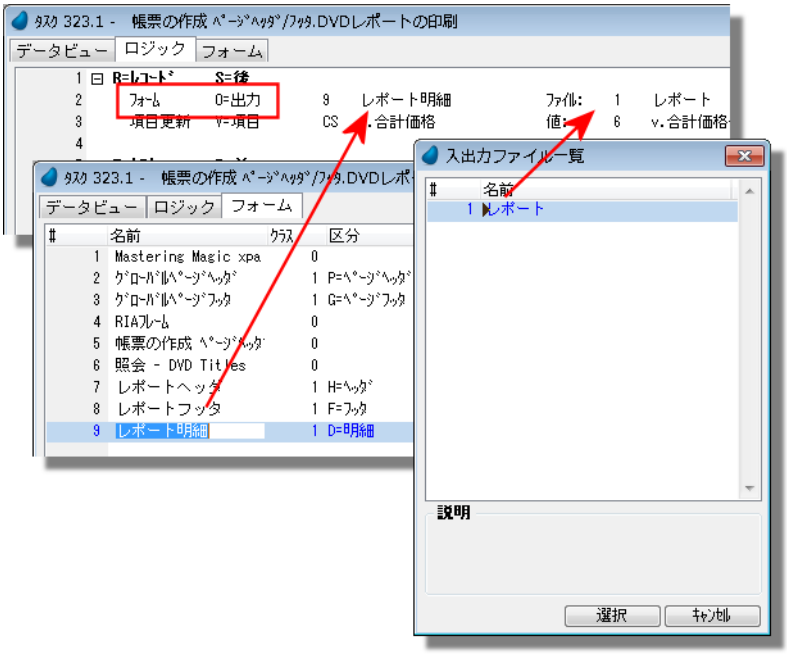
テーブルコントロールが帳票フォームに定義された場合、オンラインフォーム上に定義された場合と特性値が異なります。

全ページタイトル特性は、タイトル行が自動的に出力されるかどうかに関わらず、すべてのページに自動的にタイトルを出力するかどうかを定義します。通常は、***Yes*** を設定します。

固定サイズテーブル特性は、テーブルの出力行数を固定にするかどうかを指定します。***Yes*** の場合、フォームエディタ上に定義された行数分のレコードが 1 ページ内に出力されます。***No*** の場合は、帳票のサイズの応じて 1 ページ内に出力される行数が変更されます。この場合、オンラインフォームと同じようにフォームの下の余白のサイズは維持されます。



明細フォームの出力処理を定義する



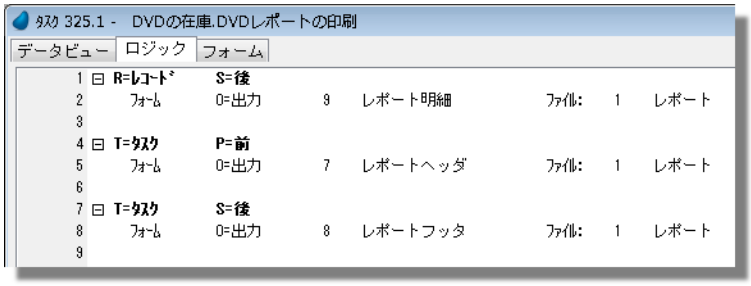
次に、帳票データを出力させる必要があります。通常、明細行は 1 レコード毎に出力されるので、上記の例のように**レコード後**ロジックユニットで出力処理を定義します。しかし、概要を出力する帳票の場合、**項目変更**ロジックユニットで定義します。意図する**ロジックユニット**で出力処理を定義することができます。

1. フォームを出力させたい**ロジックユニット**に移動します。上記の例では、**レコード後**です。
2. *F4* を押下して 1 行追加します。
3. *F4* を入力して**フォーム**処理コマンドを選択します。次のカラムには ***O= 出力*** （デフォルト）が設定されます。
4. 次のカラムに移動し、**ズーム**して出力させたいフォームを選択します。
5. **ファイル**カラムには、デフォルトの入出力ファイル *1* が設定されます。入出力ファイルが複数定義されている場合、ここから**ズーム**して選択できます。

これでプログラムを実行すると、印刷プレビュアーが起動され、出力結果が表示されます。

次にヘッダとフッタを追加します。

ヘッダを出力する

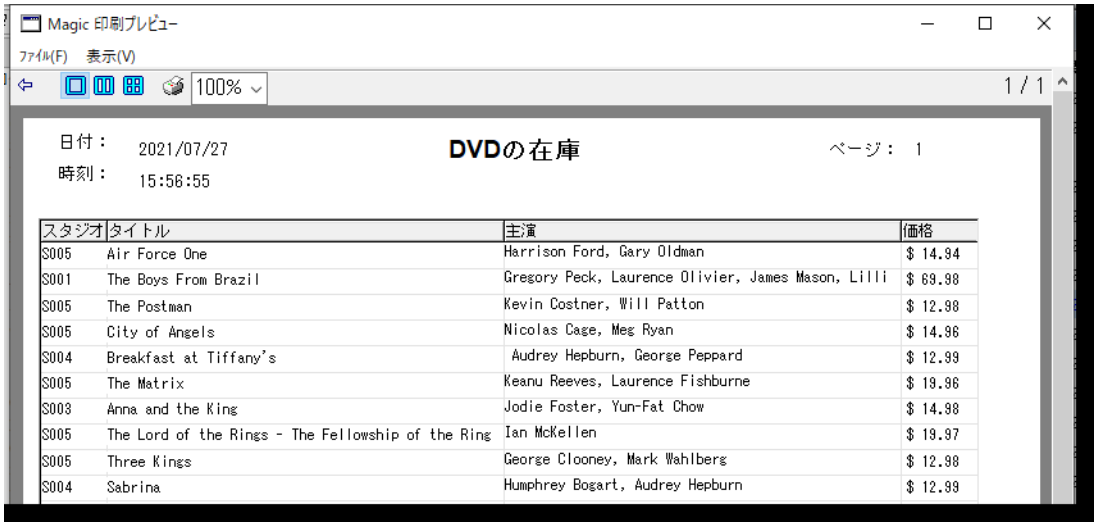


帳票ヘッダを出力する場合は、明細行を出力する場合と同じような作業を行います。ただし、ヘッダは通常**タスク前**で最初に出力されます。その後、改ページが行われるたびに自動的に出力するようにします。ヘッダフォームは複数定義されている場合があり、これらのすべてが改ページされた時点で出力されます。

別の方法として、フォームをページ ヘッダとして定義することができます。この場合、あらゆるページの先頭で自動的に出力され、**タスク前**に出力処理を定義する必要がありません（「ページヘッダ / フッタの情報を定義するには」（500 ページ）を参照）。

フッタを出力する

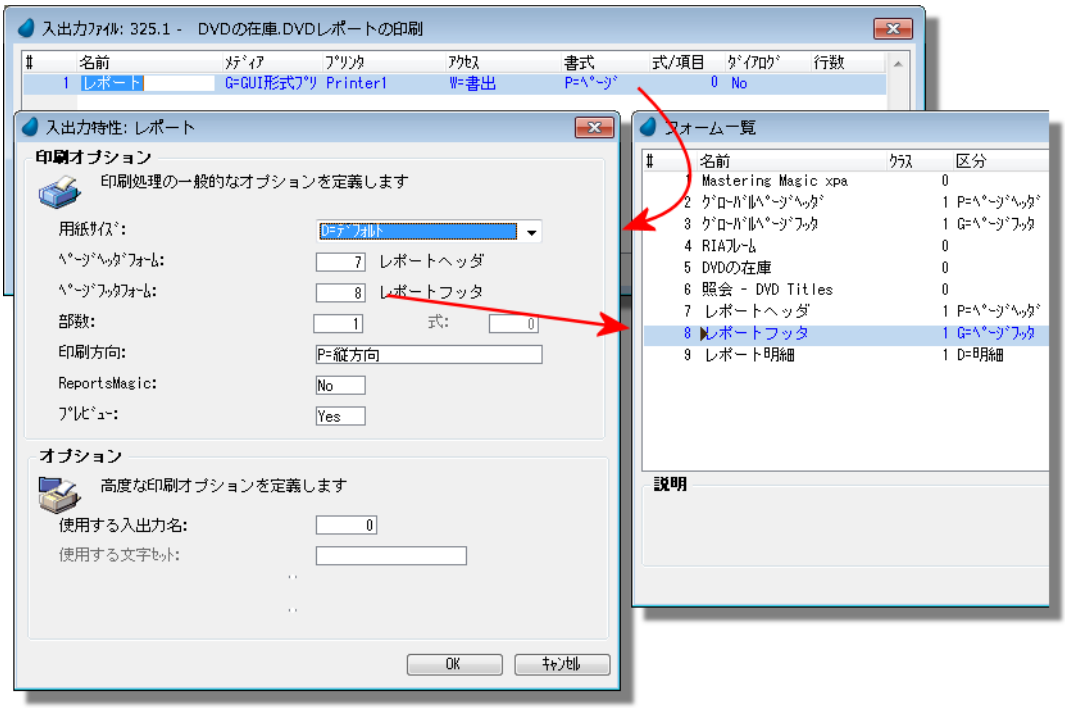
帳票フッタは最後のページが印刷された後に**タスク後**で出力されます。この場合（最後のレコードが印刷された後に）1 回だけ印刷されます。このため**タスク後**に処理を定義します。小計を印刷するためにブレイクが発生した場合にもフッタを出力することがあります。これについては、「ブレイク レベル毎の総計を定義するには」（509 ページ）を参照してください。



これでヘッダとフッタを含めた帳票出力プログラムが作成できました。

参照：
「帳票のブレイクレベルを作成するには」（507 ページ）
「ブレイクレベル毎の総計を定義するには」（509 ページ）
「帳票に複数行のコントロールを定義するには」（510 ページ）
「帳票のテーブルに対して見出しを繰り返し出力させるには」（512 ページ）

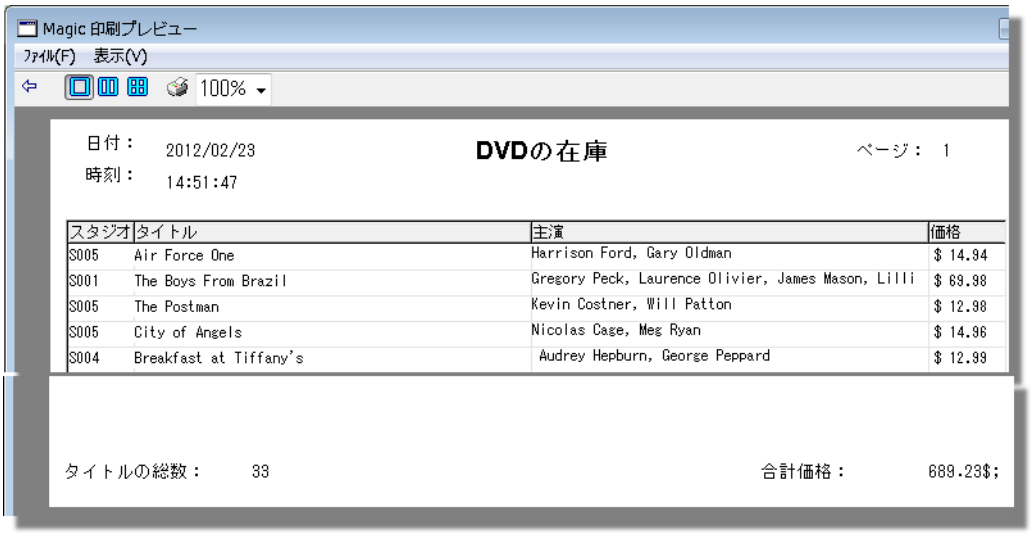
ページヘッダ / フッタの情報を定義するには



ページヘッダと***フッタ***は入出力ファイルの入出力特性で指定されます。位置は自動的に処理されます。***ページヘッダ***はページの先頭に、***ページフッタ***はページの最後に出力されます。

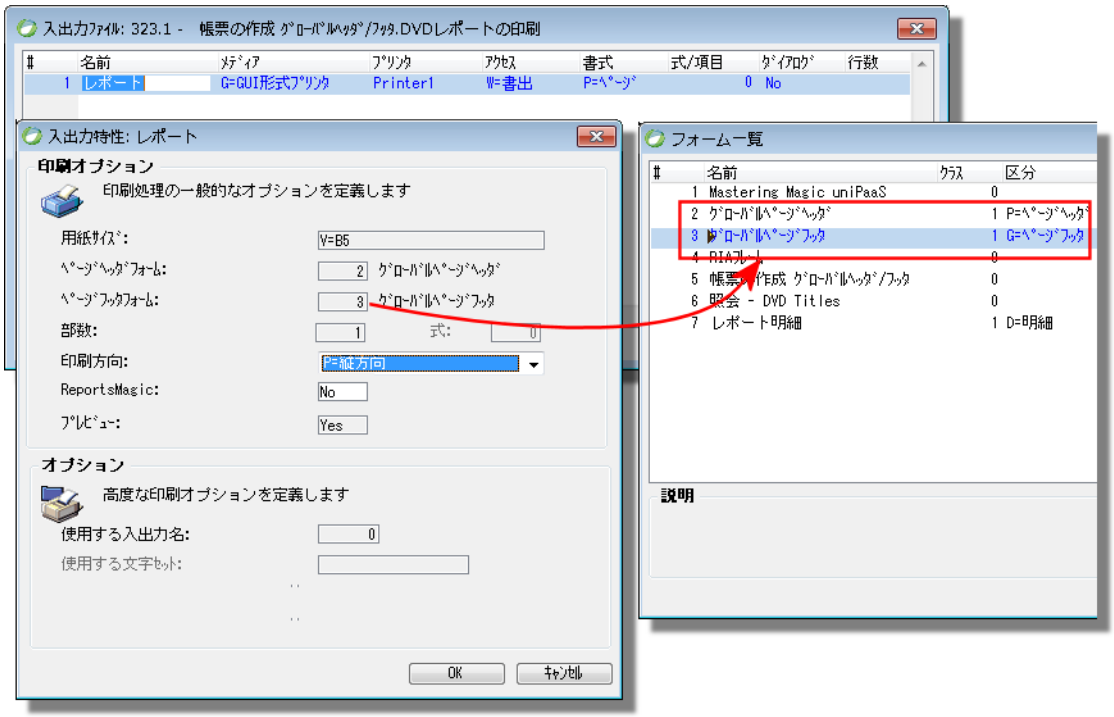
ページヘッダと***フッタ***は、必ずしも帳票のヘッダやフッタと同じではないことに注意してください。***ページヘッダ***と***フッタ***はすべてのページに出力され、すべてのページが同じ種類の帳票のものであるようにします。例えば、典型的な***ページフッタ***は各ページの下部の同じ位置に出力され、罫線とページ番号のみが出力されるようにします。しかし、***レポートフッタ***は合計を出力したり、帳票の最終行のすぐ後に出力されたりします。

ページヘッダと***フッタ***のために使用されるフォームは、現在のタスクに定義される必要はありません。***ページヘッダ***をメインプログラムで定義し、すべての帳票フォームで利用することもできます。この方法については、「グローバルなページヘッダ / フッタを定義するには」（501 ページ）を参照してください。



ヒント : *印刷などのページのタイプにもとづいたヘッダやフッタフォームの出力内容を変更するために**可視**特性を使用することができます。ある条件下で表示される項目と、その他の条件で表示される項目を組み合わせることで、条件内容にもとづいて出力される項目の内容を変えることができます。*

グローバルなページヘッダ / フッタを定義するには



帳票をページヘッダ付きで出力する方が便利な場合があります。帳票の書式の一貫性が保たれたり、新しい帳票を作成する際の手間を省く場合もあります。また、会社名が変更された場合に帳票の設定を簡単に変更することもできます。

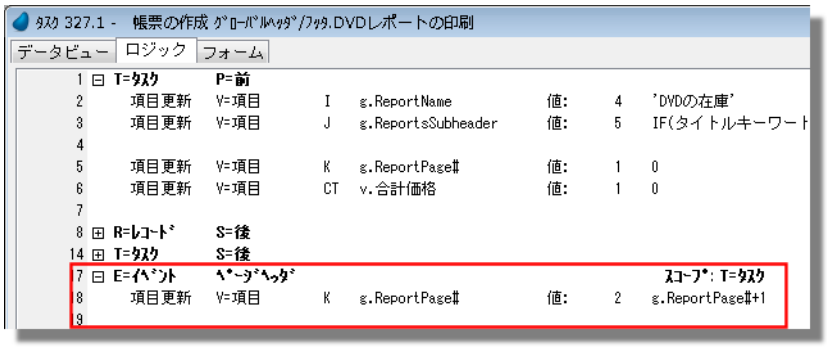
ページヘッダと**フッタ**は、入出力特性で定義するため、タスクのローカルフォームからグローバルなフォームに切り替えることが簡単にできます。

グローバルフォームを作成し使用する

メインプログラムでフォームを作成することにより、グローバルなフォームを簡単に作成することができます。作業手順は、通常のタスクでフォームを作成する方法とまったく同じです。

フォームが作成されると、すべてのタスクのフォームエディタ上にローカルフォームより上に表示されます。上記の例では、2 つのグローバルなフォームが表示されています（フォーム #2 とフォーム #3）。このタスクで定義されているローカルなフォームは、フォーム #6 と #7 です。

グローバルなページヘッダ / フッタ上でデータ項目を扱う

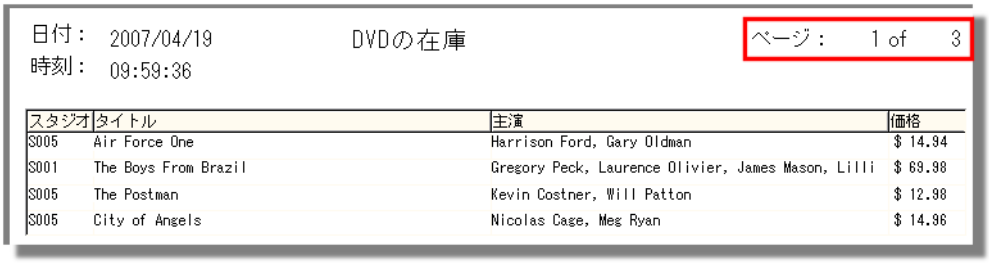


グローバルなヘッダとフッタを作成する際にいくつか考慮する点があります。

第一に、ヘッダには一般的に帳票名が出力されます。これは、何らかのフィルタ機能を使用して出力するようにします。グローバル変数をメインプログラムに定義し、タスク前でこれらを更新させることで簡単に処理することができます。

第二に、メインプログラムでは、帳票用に使用される **Page(*n,n*)** 関数が動作しないため、ローカルタスク内でページ数の値を更新させる必要があります。これは、ページヘッダイベントのロジックユニット内で処理を定義するで実現できます。ページヘッダイベントが実行されると、グローバル変数として定義されたページ番号を更新するようにします。上記の例では、入出力ファイルは現在のタスクの最初に定義されたもののため、**Page(*0,1*)** が定義されています。

帳票ページを列挙するには



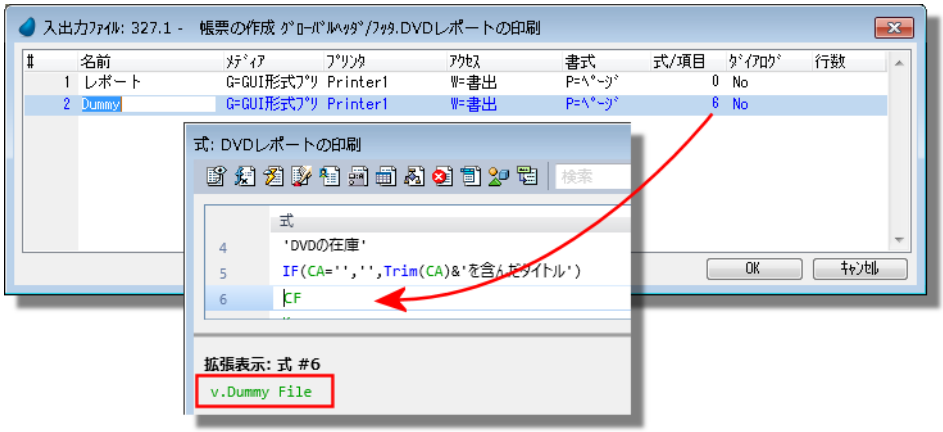
ユーザによっては、「*n of nn*」という付番方式を使用してページ番号を出力する場合があります。

しかし、出力するレコード数や指定されたプリンタをもとに全体のページ数を事前に取得することは不可能です。印刷マージンやフォントを変更するだけでもページサイズは変わります。このような処理を実現するには、出力処理を２回（1 番目：ページ総数の取得、２番目：実際の出力処理）繰り返す必要があります。

このようにすれば、Magic xpa で実現することができます。以下でこの方法について説明します。

この例では、ヘッダかフッタに出力するページ番号をメインプログラム内で格納するようにします。従って、変数項目の g.Report Page# は現在のページ番号を保持し、g.Report Total Pages は出力するページの総数を保持するものとして定義します。グローバルなページヘッダ／フッタの使用方法については、「グローバルなページヘッダ / フッタを定義するには」（501 ページ）を参照してください。

1. ２つの入出力ファイルを定義する



帳票出力用のサブタスクには、2 つの入出力ファイルを定義します。最初は、実際に出力するプリンタです。2 番目は、ファイル（%TEMP%Dummy.txt のような）として出力されるものです。ファイル名を保持するために、親タスクで変数項目を定義し、**FileDelete()** 関数を使用してサブタスクが終了した時点でファイルを削除するようにします。

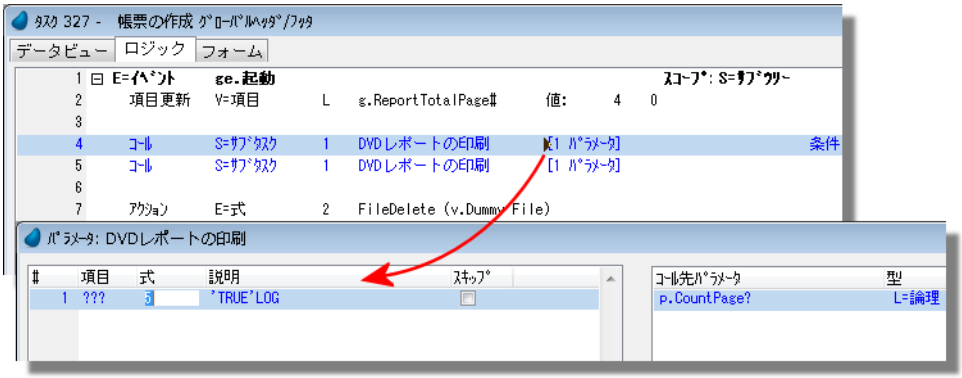
このファイル名の存在は、実際の印刷から出力内容を保護する働きをします。データが印刷されているかのようにフォーマットされますが、実際はファイルに保存されます。

これらは同じ出力先として定義されていることが重要な点です。例えば、同じヘッダとフッターを使用していても、出力先が異なる場合はページ数が同じにならない場合があります。

また、出力用のサブタスクでパラメータ項目（p.Count Page?）を作成します。これが True の場合、タスクはダミーを出力し、実際には印刷されません。

注： 通常は、両方の入出力ファイルで同じプリンタを定義することになります。しかし、Windows に複数のプリンタがセットアップされており、同じ処理を２回繰り返さないように警告を出すようにプリンタドライバが設定されている場合もあります。このような場合、プリンタテーブル（オプション→設定→プリンタ）で DummyPrinter を設定してください。

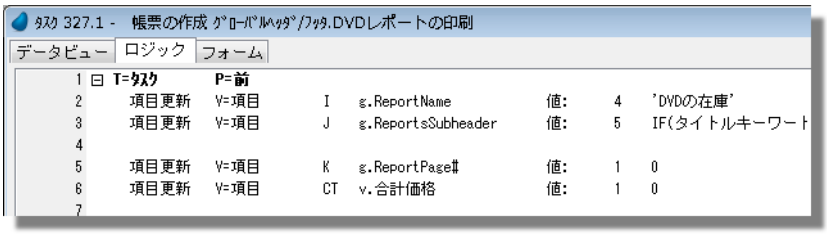
2. タスクの呼び出し処理を定義する



タスクの呼び出し処理では、4 つの処理を定義します。

1. グローバル変数「g.ReportTotalPages」の値を 0 に設定します。
2. p.Count Pages? = TRUE としてパラメータを渡し、サブタスクを呼び出します。ここでは、ページ総数を算出します。
3. p.Count Pages? = FALSE としてパラメータを渡し、サブタスクを呼び出します。ここでは、実際に印刷処理を行います。
4. 一時ファイルを削除します。

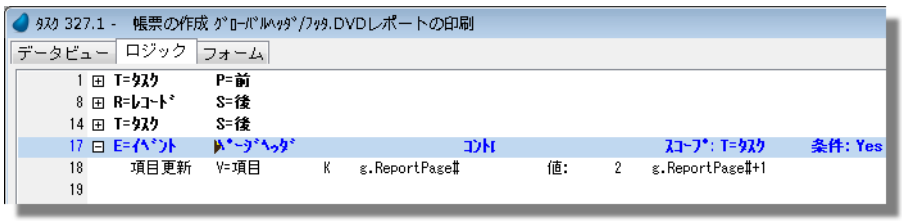
3. タスク前で出力項目を初期化する



サブタスクでは、グローバル変数「g.Report Page#」やその他の演算用の変数項目の値を初期化します。

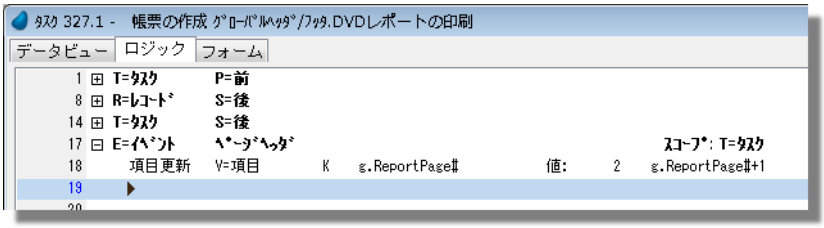
注： タスク実行される際に、自動的に値が初期設定されるため、総額のような項目を 0 に設定する必要はありません。しかし、時々タスクが常駐するように設定されている場合もあるため、とにかく 0 に設定しておいた方が無難です。このような場合、ユーザが印刷処理を２回実行した場合、合計が突然 0 に設定される場合があります。

4. ページヘッダイベントのロジックユニットを作成する



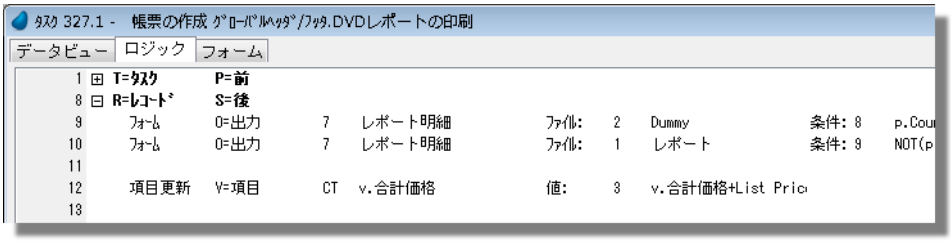
グローバルなページ番号を更新するために、ページヘッダイベントのロジックユニットを作成します。これは、どちらの入出力ファイルに対するページ番号かを注意して作成する必要があります。

5. 最初の処理で取得されたページ総数を格納する



次に、タスク後で、グローバル変数「g.ReportTotalPages」の内容を現在のページ番号によって更新します。この更新処理には条件が定義されており、最初の出力処理でのみ実行されます。結果として返る値は、各出力処理で同じになるため、この処理は厳密には不要ですが、ロジックを明確にするために定義しています。

6. ２つの入出力ファイルに出力する



さて、ここには出力処理に関するすべての処理が定義されています。各フォーム出力処理コマンド（ほとんどの帳票では、あまり多くはありませんが、これはそのほんの一例です）を複写します。p.Count Pages が True の場合、最初のフォーム出力はダミーファイルの出力処理です。p.Count Pages が False の場合、2 番目のフォーム出力はプリンタへの出力になります。

さて、このプログラムが実行されると、サブタスクが 2 回実行され、「*n of nn*」の正確なヘッダデータが取得できます。

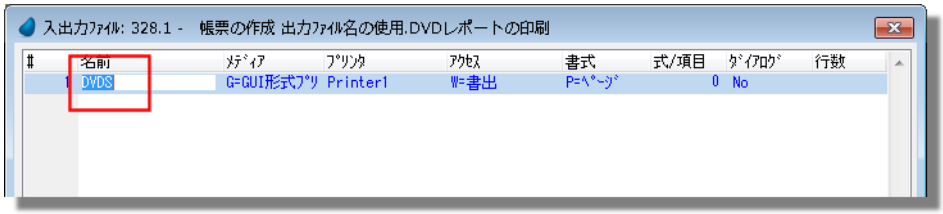
複数のプログラムで同じ入出力ファイルを使用して出力するには

入出力ファイルの範囲は、項目やフォームとほぼ同じです。すなわち各タスクは上位タスクの入出力ファイルにアクセスすることができ、プログラム内で使用することができます。

しかし、印刷処理のために全く別のプログラムを呼び出すこともできます。例えば、要約情報を印刷したり、テキストをフォーマットするための汎用プログラムが必要な場合があります。このような場合、**使用する入出力ファイル名**と呼ばれる機能を使用することができます。

この例では、いくつかの帳票の最後に印刷するスタジオデータというプログラムを作成しています。これによって、新規ユーザがスタジオコードを把握することができるようになっています。コードを出力するプログラムをすでに持っている場合、他のプログラムからどのように出力させるようにするかを以下に示します。

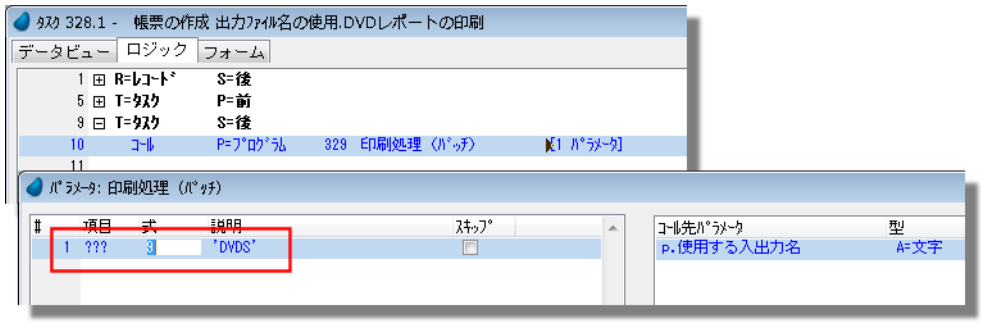
プログラムの呼び出し処理を設定する



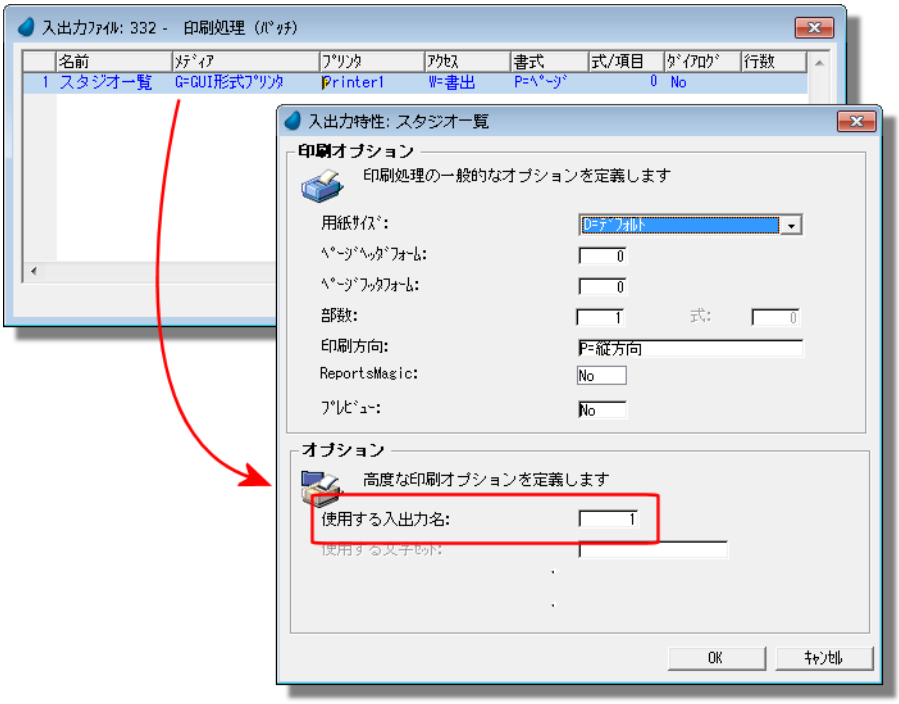
最初に、入出力ファイルを開くプログラムを作成する必要があります。ここでは、以下の２つの定義処理を行います。

1. 入出力ファイルを定義します。空白を含まない簡単な名前を設定します。
2. 入出力ファイル名を呼び出されたプログラムに渡します。プログラム内にハードコートされている場合は必要ありません。

この例では入出力ファイル名を DVDS としています。また、式エディタで DVDS という文字列を定義し、呼び出されたプログラムで渡すようにしています。



起動されるプログラムを設定する



さて、呼び出されるプログラムで**入出力ファイル**の定義を行います。**入出力特性**で、**使用する入出力ファイル名**特性に渡されたパラメータ値を設定します（実行時に DVDS と評価されます）。

これで呼び出されたプログラムの出力内容は、最初のプログラムでオープンされた入出力ファイルで指定されたデバイスに出力されます。

ヒント : *場合によっては、出力用に２つ以上の個別のプログラムを呼び出す「コントロールタスク」を定義する必要があるかもしれません。この場合は、コントロールタスク内で入出力ファイルをオープンします。そのタスクがオンラインタスクであったり、出力処理が定義されていなかったりしている場合、空白ページが出力されることを防止するために、* **設定→動作環境→動作設定**の**入出力デバイスのオープンタイミング**を ***0= 利用時***に設定します。

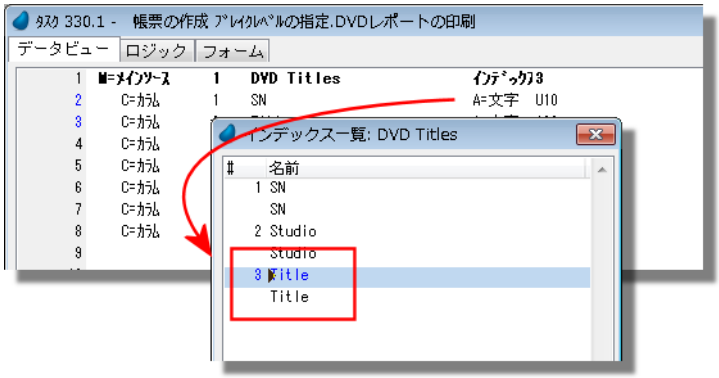
帳票のブレイクレベルを作成するには

データがグループ分けされ、小計を印刷する場合、帳票プログラムではブレイクレベルを定義する必要があります。これは低レベルの言語で行うような処理ですが、Magic xpa には複雑なブレイクレベルであっても簡単に処理できる機能が組み込まれています。このセクションでは、ブレイクレベルの設定方法について説明します。作業手順は以下の通りです。

1. ブレイクレベルに対応したレコードの並び順を設定します。
2. ブレイクの発生対象となるデータ項目を決めます。
3. グループレベルを設定します。
4. 合計を算出します。

次に、これらの手順について説明します。

1. ブレイクレベルに対応したレコードの並び順を設定する



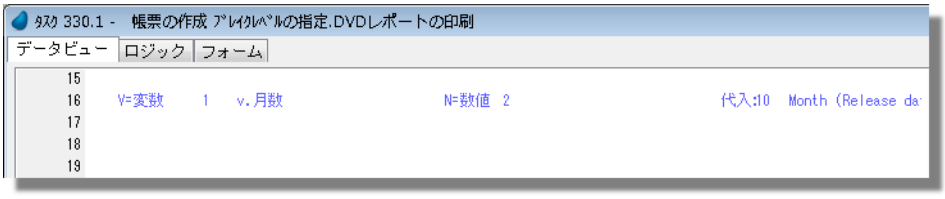
ブレイクレベル帳票を行う上で、最も一般的な間違いの 1 つは間違ったレコードの並び順を使用することです。この例では、Studio コードでレコードをグループ分けしています。従って、Studio インデックスを使用する必要があります。

最適なインデックスが定義されていない場合は、最適な並び順になるようにソート定義（**タスク環境→ソート**）を行う必要があります（第 17 章：「定義されたインデックスを使用しないでレコードをソートするには」（411 ページ）を参照してください）。

2. ブレイクの発生対象となるデータ項目を決定する

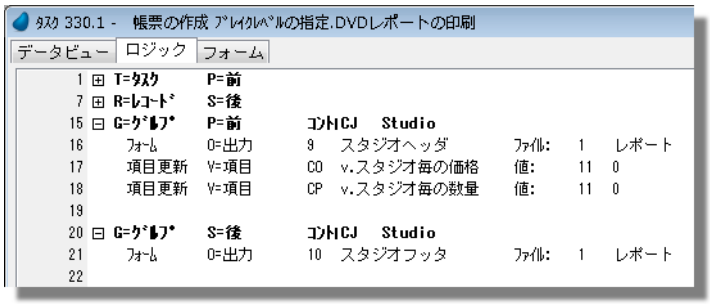
通常、ブレイクレベルで使用するブレイク項目は、DB ソース内のカラムを使用します。この例では、レコードが Studio コード順に出力されるため、Studio カラムを使用してブレイクします。

しかし、適当なカラムが存在しない場合もあります。例えば、月毎にデータを要約する帳票を作成したくても日付カラムしか存在しない場合があります。レコードは日付カラムによって正しくソートされますが、月が変わった場合にブレイクさせるためのカラムはありません。



このような場合、例えばここで示されているように、v. 月数と呼ばれる項目を設定し、**Month()** 関数を使用して日付の月の値を設定することでブレイクさせるようにします。または、項目のいずれか 1 つが変わる時だけブレイクが発生するように複数の項目を連結して 1 つのブレイク項目にすることもできます。変数項目は、カラム項目と同じようにブレイクレベルで使用することができます。

3. グループレベルを設定する

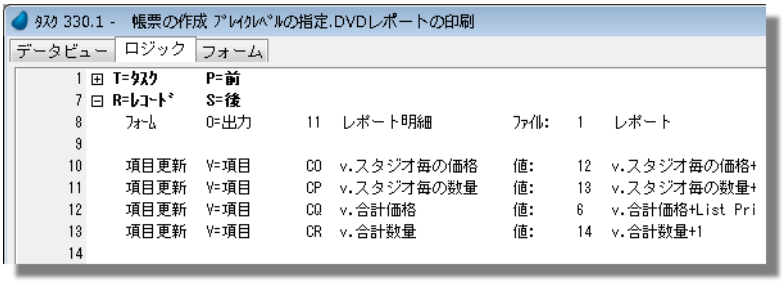


次に、グループレベルを設定します。グループレベルは、ブレイク処理で使用する項目を指定します（この例では、Studio カラムです）。

グループ前では、（必要であれば）ヘッダの出力と合計値の演算用変数の初期化を行います。

グループ後では、（必要であれば）フッタを出力します。グループ後では、フォームの印刷後に変数の初期化を行うこともできます。どちらの方法にしても、アプリケーションで一貫した方法で定義する必要があります。

4. 合計値の計算を行う



最後に、レコードを印刷する場合は常に、合計値を更新する必要があります。通常、このような処理はレコード後で簡単な更新処理によって行われます。

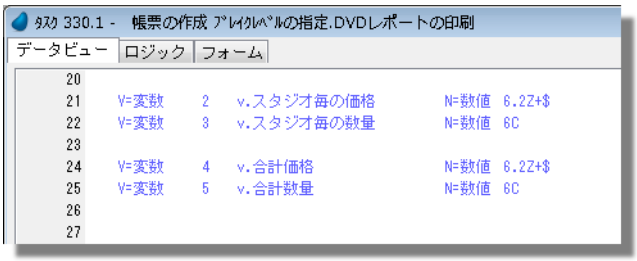
ブレイクレベル毎の総計を定義するには

ブレイクレベル毎の総計を定義するには、以下の３つの手順を実行します。

1. 総計用項目の定義
2. 総計用項目の更新
3. 総計の初期化

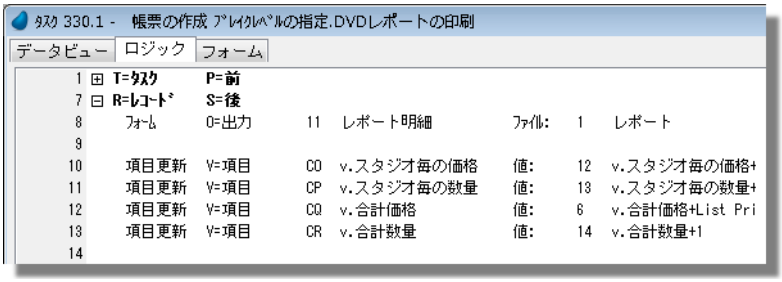
以下これらの手順について説明します。

1. 総計用項目を定義する



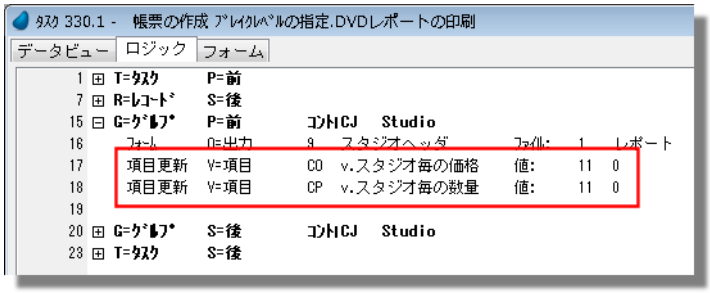
総計用に使用される項目は、通常は変数項目を使用します。これらは、データビューエディタで定義されます。これらの変数項目には目的がわかるような名前を定義しておきます。この例では、どの項目がどのブレイクレベルで使用されるかが分かるように、大文字の接頭辞を追加しています。

2. 総計用項目を更新する



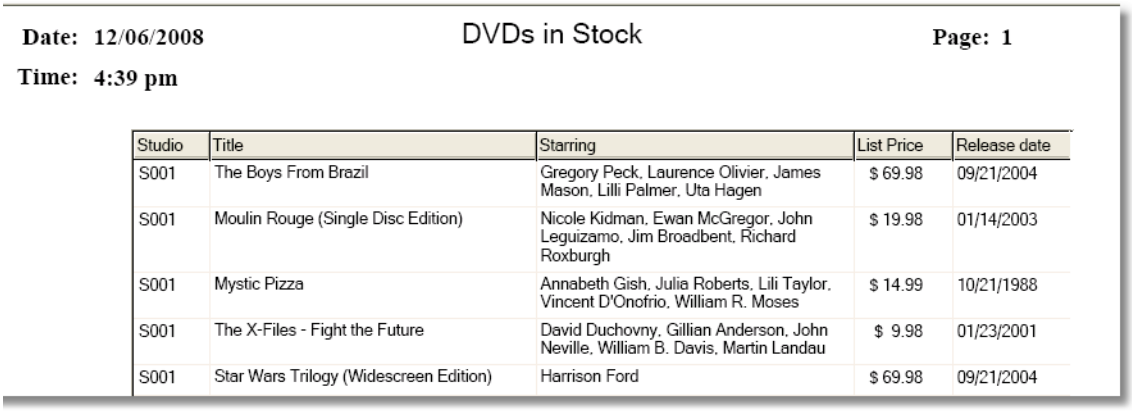
各レコードが読み込まれるたびに、総計用項目を更新するための簡単な方法は、加算することです。この方法により、すべての更新処理は一箇所に定義されます。これによってデバッグしやすくなります。

3. 総計用項目を初期化する



最後に、総計用項目の初期化処理が必要になります。この処理は、グループ前かグループ後で印刷処理の終了後に実行します。

帳票に複数行のコントロールを定義するには



Date: 12/06/2008 DVDs in Stock Page: 1

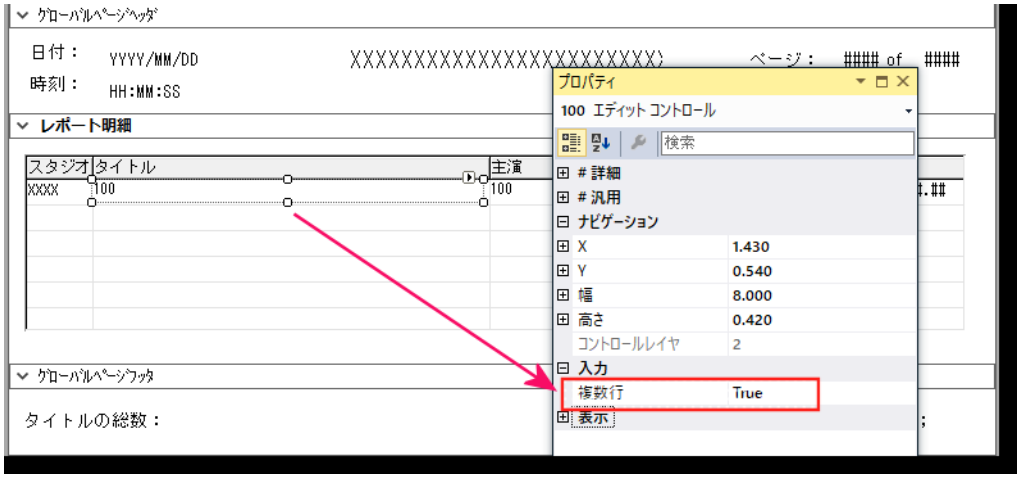
Time: 4:39 pm

Studio	Title	Starring	List Price	Release date
S001	The Boys From Brazil	Gregory Peck, Laurence Olivier, James Mason, Lilli Palmer, Uta Hagen	$ 69.98	09/21/2004
S001	Moulin Rouge (Single Disc Edition)	Nicole Kidman, Ewan McGregor, John Leguizamo, Jim Broadbent, Richard Roxburgh	$ 19.98	01/14/2003
S001	Mystic Pizza	Annabeth Gish, Julia Roberts, Lili Taylor, Vincent D'Onofrio, William R. Moses	$ 14.99	10/21/1988
S001	The X-Files - Fight the Future	David Duchovny, Gillian Anderson, John Neville, William B. Davis, Martin Landau	$ 9.98	01/23/2001
S001	Star Wars Trilogy (Widescreen Edition)	Harrison Ford	$ 69.98	09/21/2004

帳票のデータは、一定でない場合があります。例えば、この例では、タイトルと主演のデータが長すぎたり短すぎたりしています。このような場合、テーブルのカラム内容を折り返して出力するようにした方が効率よく収まります。

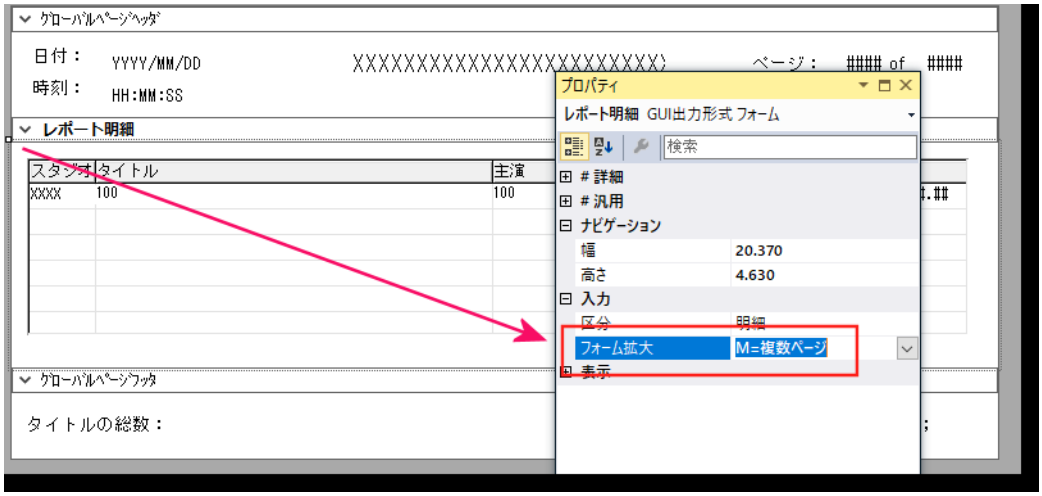
Magic xpa では、**項目特性**を変更することで、データをより多く出力させることができます。

1. コントロール項目特性を変更する



出力フィールドを拡張するには、**コントロール特性**の**複数行編集**特性を *Yes* に設定します。

2. フォーム特性を変更する

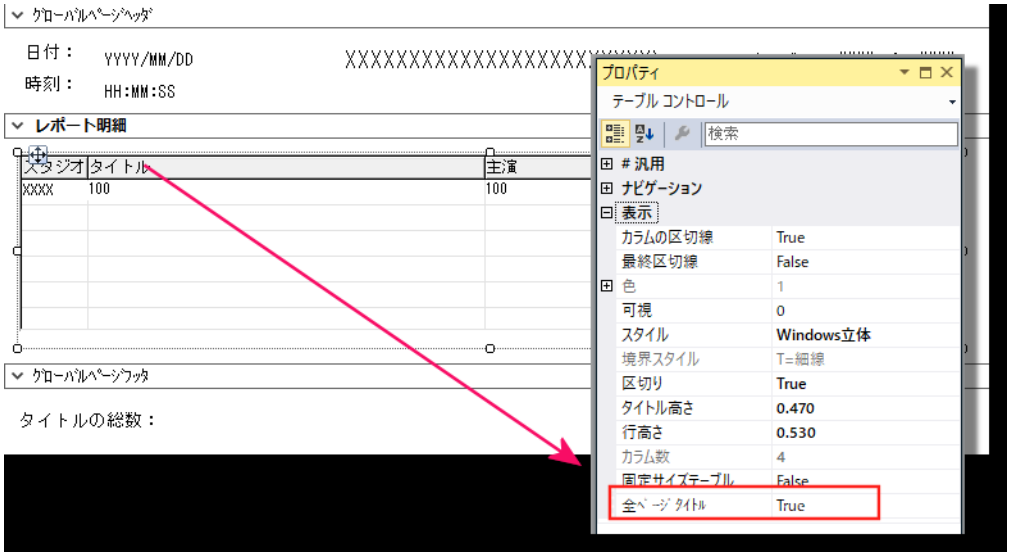


拡張フィールドを含んでいるフォームに対して、**フォーム特性**の**フォーム拡大**特性を *M= 複数ページ* に変更します。

これで、データが 1 行に納まらない場合、テーブル行が拡張して出力されます。

注： 印刷プレビュアーでは、折り返し行の最後の行が正しく表示されない場合があります。これは印刷プレビュアーの問題であり、プリンタには正しく出力されます。

帳票のテーブルに対して見出しを繰り返し出力させるには



テーブルの見出しは、通常各ページごとに先頭に一回出力されます。例えば、コントロールブレイクを持っている場合、テーブルが繰り返される度に出力されます。

この動作は、テーブルコントロールの**全ページタイトル**特性よって設定できます。ページごとに、見出しを出力させたい場合は、この特性値を ***Yes*** に設定します。最初のページのみに見出しを出力させたい場合は、***No*** を設定します。

ReportsMagic 用フラットファイルを出力には

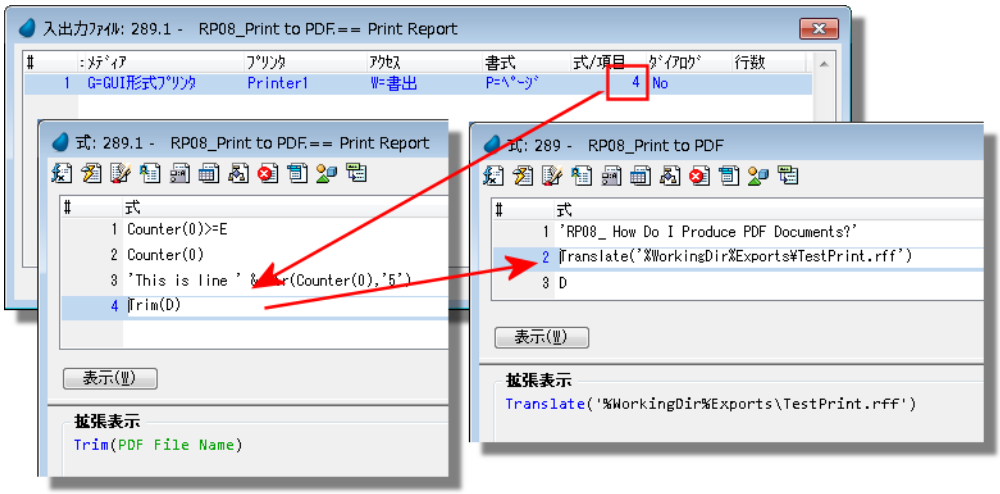
Magic xpa Ver3 では、入出力特性に **ReportsMagic** 特性が追加されました。この特性を有効にすると、印刷結果を ReportsMagic 用のフラットファイルとして出力することができます。

また、Magic xpa Ver3 の Studio/Client にバンドルされている **RMViewe** を起動してフラットファイルを表示させることで、プレビューワ機能を実現することができます。

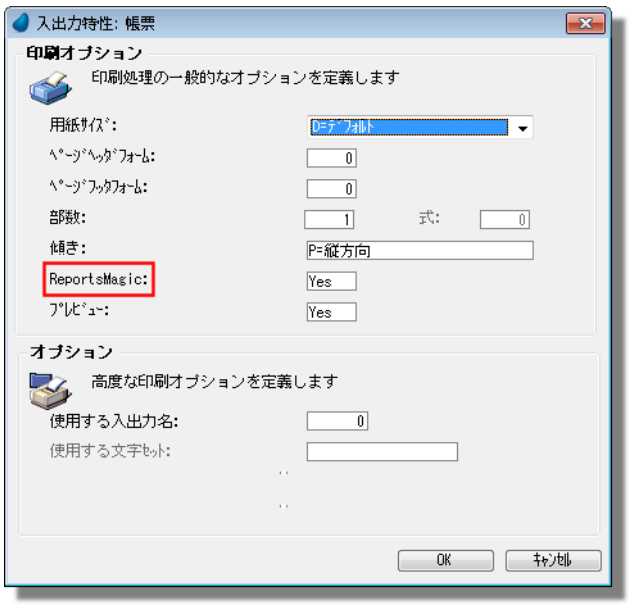
ReportsMagic の RMViewr をプレビューワとして使用する

データ出力用の入出力ファイルを設定します。

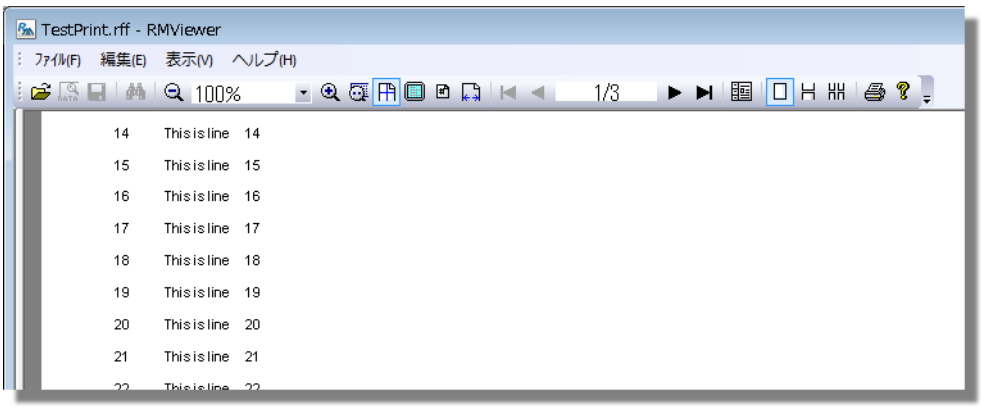
1. データ出力用のサブタスクに移動します。
2. ***Ctrl+I***（**タスク環境→入出ファイル**）を押下して**入出力ファイル**テーブルを開きます。
3. ***F4*** を押下して、**入出力ファイル**テーブルに 1 行追加します。
4. **メディア**カラムで ***G=GUI 形式プリンタ***を選択します。
5. **式／項目**カラムで任意の名前のフラットファイル（拡張子が ".rff"）を式で定義します。



6. **入出力特性**（***Alt+Enter***）を開き以下の特性を設定します。
   - ReportsMagic …… Yes
   - プレビュー…… Yes



これでプログラムを実行すると、RMViewer が起動して出力結果が表示されます。RMViewe からプリンタへの印刷を行うこともできます。



ReportsMagic の Active X 版 RMViewr をプレビューワとして使用する

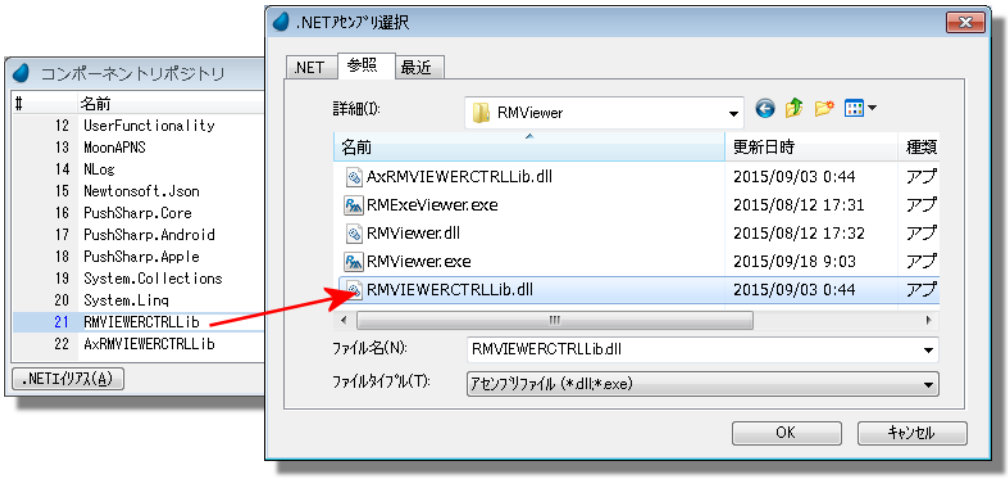
RMViewer の ActiveX 版は、画面を持った ActiveX です。Magic xpa のオンライン画面に配置することで帳票データを画面に表示させることができます。

.Net アセンブリを設定する

RMViewer ActiveX モジュールは、RMViewerCtrl.ocx です。Magic xpa では、オンラインタスクのアーキテクチャーが、Win32 から .NET Framework へ移行したため、Active X がサポートされなくなり、.NET コントロールを利用することになりました。

.NET コントロールを使用する場合には、あらかじめ、**コンポーネント**リポジトリに‘RMVIEWERCTRLLib’と’AxRMVIEWERCTRLLib’を設定します。

1. **コンポーネント**リポジトリを開き、F4 を押下して、1 行追加します。
2. **タイプ**カラムで ***.NET*** を選択します。
3. **名前**カラムで**ズーム**（***F5***）して **.NET アセンブリ選択**ダイアログを開きます。
4. **参照**タブを開きます。Magic xpa Studio のインストールフォルダ内の RMViewer サブフォルダに移動して、RMVIEWERCTRLLib.dll を選択し、OK を**クリック**します。
5. 同様に AxRMVIEWERCTRLLib.dll も登録します。



データビューの定義

印刷処理の結果を表示するオンラインプログラムを作成します。.Net アセブリオブジェクトのプロパティやメソッドを設定するため、**データビュー**エディタに .Net 型の変数を定義して .Net オブジェクトを定義します。

1. **データビュー**エディタで F4（編集→行作成）を押下して、行を追加し、以下の変数項目を追加します。
   - **項目名** …… AxRMViewerCtrlLib
   - **データ型** …… .Net
   - **オブジェクトタイプ特性** …… AxRMVIEWERCTRLLib.AxRMViewerCtrl

ロジックの定義

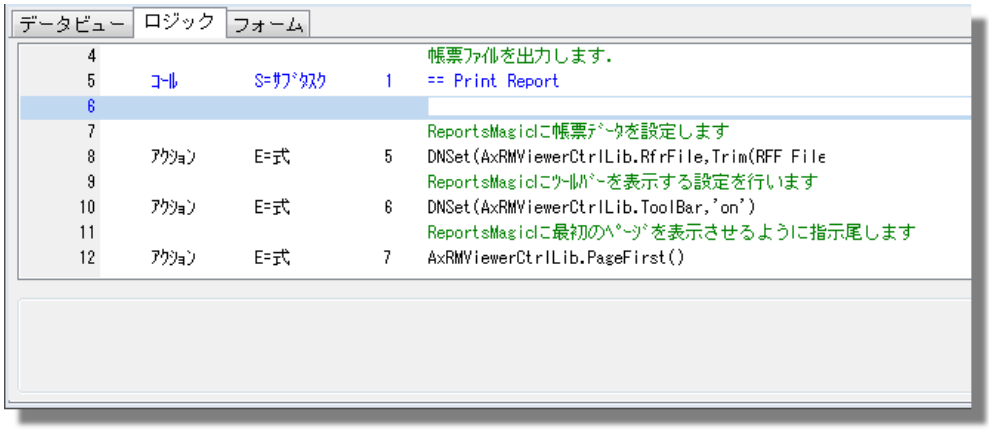
ロジックエディタを開き、帳票出力と表示処理のロジックを定義します。帳票出力は、前述の「ReportsMagic の RMViewr をプレビューワとして使用する」（513 ページ）で作成したプログラム（または、サブタスク）を呼び出すことで対応できます（この場合は、**プレビュー**特性を *No* に設定します）。

.Net オブジェクトにパラメータを渡すロジックの定義について説明します。

1. **ロジック**エディタを開き、*F4*（**編集→行作成**）を押下して行を追加します。
2. **アクション**処理コマンドを使用して以下のような式を発行するように定義します。
   - 出力する帳票データの指定：
     DNSet(*F.RfrFile,Trim(D)*)
   - ReportsMagic のツールバーを表示する指定：
     DNSet(*F.Toolbar,'on'*)
   - 最初のページを表示する指定：
     F.PageFirst()

パラメータ：

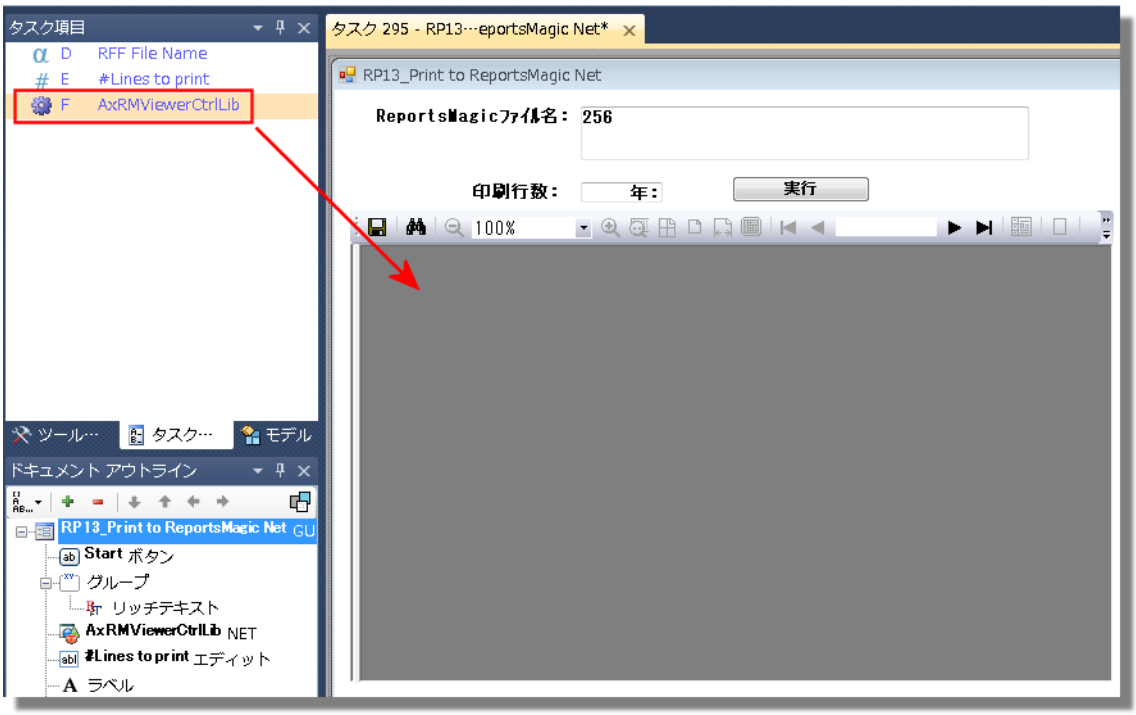
- F:.Net 項目（AxRMViewerCtrlLib）の項目 ID。**式**エディタでこの変数項目を選択すると .Net オブジェクトのプロパティやメソッド名が表示されます。
- D: 出力する ReportsMagic のフラットファイルのファイル名。



表示フォームの定義

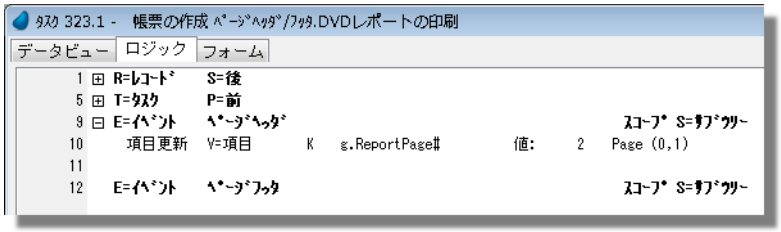
GUI 表示フォームに .Net オブジェクトを配置することで出力内容を表示する画面が定義されます。

1. **フォーム**エディタを開き、**名前**カラムから**ズーム**します。編集用フォームが開きます。
2. **タスク項目**ペインに表示されている変数項目から AxRMViewerCtrlLib をクリックして選択し、フォームにドラッグ＆ドロップします。
3. ReportsMagic の画面が配置されるので、マウスで位置やサイズを調整します。



このプログラムを実行すると、バッチタスクで作成された ReportMagic 用のフラットファイルがオンライン画面に表示されます。

ページ毎に計算処理を実行させるには



ページ毎に合計値やその他の計算結果を算出する必要があるかもしれません。例えば、「ページ毎の総費用」だったり、ページ番号を加算したりする場合などが考えられます。しかし、GUI 形式の帳票ではページ毎に出力する行数は可変のため、印刷中に手動で算出することは非常に難しい処理になります。従って、Magic xpa ではページヘッダとページフッタの２つの内部イベントを提供することで対応しています。これらのイベントを利用することで、ページヘッダやページフッタを出力する前に処理を実行させることができます。

ページヘッダやページフッタ自体は、自動的に出力されるため、これらのイベントを使用する必要はありません。

参照： 「ブレイクレベル毎の総計を定義するには」（509 ページ）
「ページヘッダ / フッタの情報を定義するには」（500 ページ）
「グローバルなページヘッダ / フッタを定義するには」（501 ページ）

帳票にマージンを設定するには

出力するプリンタや帳票の種類によって、周辺の余白のサイズが異なる場合があります。このような場合、印刷時にマージンを指定することで調整することができます。ここでは、マージンの設定方法について説明します。

注： 印刷マージンの設定は、日本語版の Magic xpa でのみ有効です。

プリンタテーブルに設定する

1. プリンタテーブル（オプション→設定→プリンタ）を開きます。
2. マージンを設定するプリンタのキューを以下のように設定します。

   Windows のプリンタ名 /M ＜上マージン＞ X ＜左マージン＞ X ＜下マージン＞ X ＜右マージン＞

例えば、Adobe PDF に対して、上マージン：1mm、左マージン：2mm、下マージン：3mm、右マージン：4mm の場合は、以下のようになります。

```
Adobe PDF/M10X20X30X40
```

スタイル設定ユーティリティを使用する

1. スタイル設定ユーティリティ（**Setstyle.exe**）を起動します。
2. **編集対象 MAGIC.INI** 欄に設定したい Magic.ini を選択します。
3. **プリンタ名**欄で、設定したいキューを選択します。
4. スタイル設定をクリックし、**スタイル設定**ダイアログを開きます。
5. **マージン設定**のチェックボックスをオンに設定にし、下の表示されている**マージン設定**欄を有効にします。
6. 上下左右のマージン（0.1mm 単位）を設定し、実行をクリックします。

全てのキューで同じマージンを設定する

1. プリンタ設定ユーティリティ（**Mgprn.exe**）を起動します。
2. **編集対象ファイル** 欄に設定したい Magic.ini を選択します。
3. **印刷形式**を ***GUI*** に設定し、**編集**をクリックし ます。**パラメータ設定**ダイアログが表示されます。
4. **プリンタ名**欄に設定したいキューを選択します。
5. **マージン設定**のチェックボックスをオンに設定して、下に表示されている**マージン設定**欄を有効にします。
6. 上下左右のマージン（0.1mm 単位）を設定し、実行をクリックします。

エンドユーザがデータソースのデータを Excel ファイルに出力させるには

Magic xpa には、エンドユーザがデータソースから Excel ファイルにデータを出力することができる機能があります。それらのデータからチャートを生成することもできます。［データ出力］の内部イベントを使用することでこれを行うことができます。

注： このイベントを有効にするには、エンドユーザがデータを Excel にエクスポートさせたいタスクの、**データ出力**特性を ***Yes*** に設定してください。このイベントは、メニューに定義してグローバルエントリとして使用したり、プログラム毎にコードを定義して発行させることができます。

オプション A: グローバルなメニューエントリを定義します

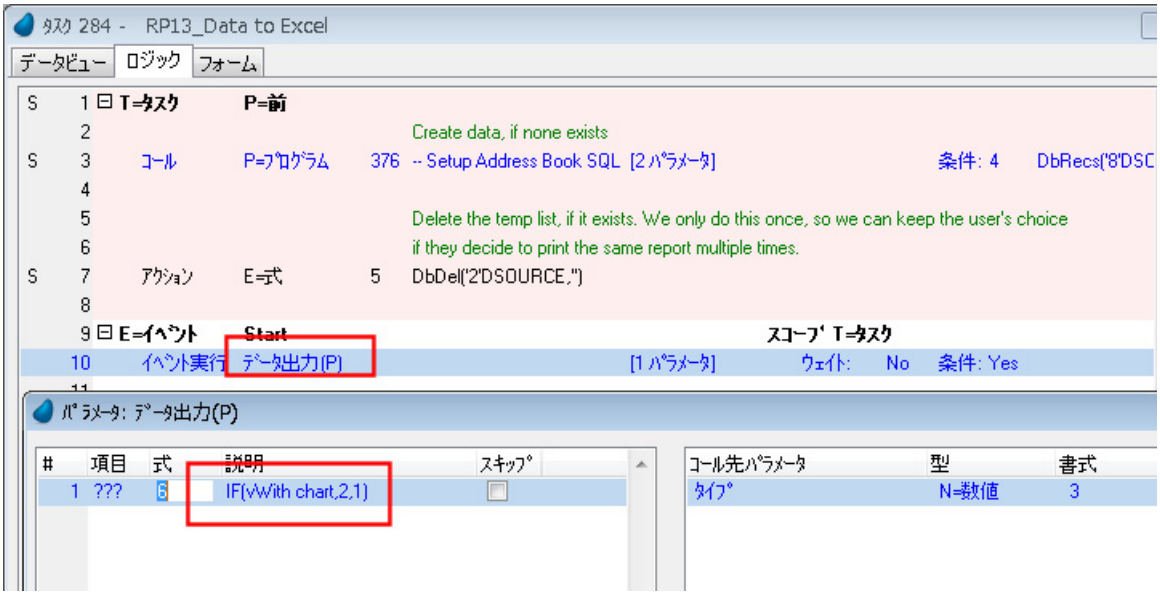
グローバルイベントエントリを入力を使用してイベントを発行します。

1. メインプログラムでズームします。
   a. 変数項目（Excel タイプ）を定義し、初期値に「1」を設定してください。
   b. 変数項目（Excel チャートタイプ）を定義し、初期値に「2」を設定してください。
2. メニューにズームします。
   a. イベントエントリを追加します。データ出力の内部イベントを選択し、メインプログラムで定義された（Excel タイプ）変数項目を渡してください。これはイベントを発行して、パラメータとして「1」の値を送ります。
   b. イベントエントリを追加します。データ出力の内部イベントを選択し、メインプログラムで定義された（Excel チャートタイプ）変数項目を渡してください。これはイベントを発行して、パラメータとして「2」の値を送ります。

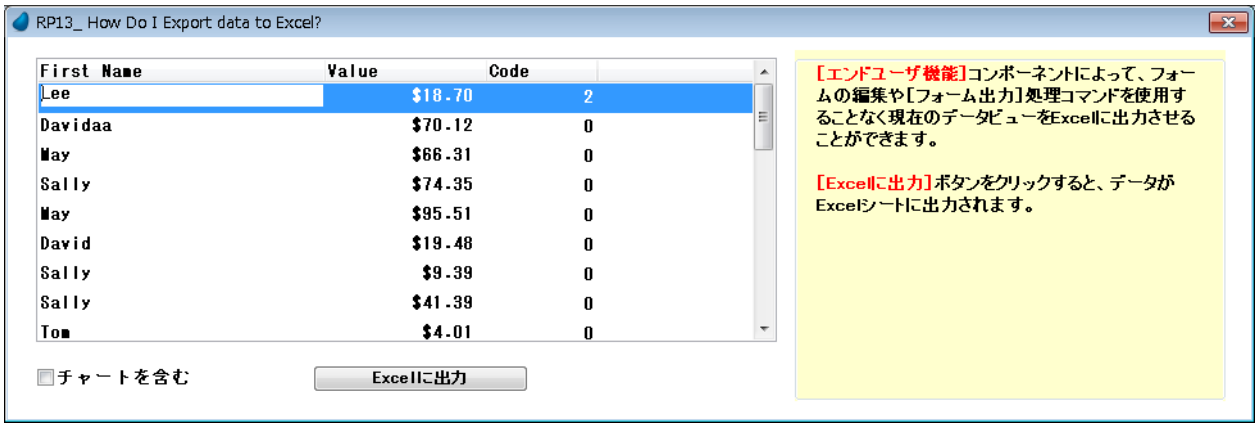
オプション B：プログラムに専用のコードを定義する

特定プログラムでイベントを発行します。

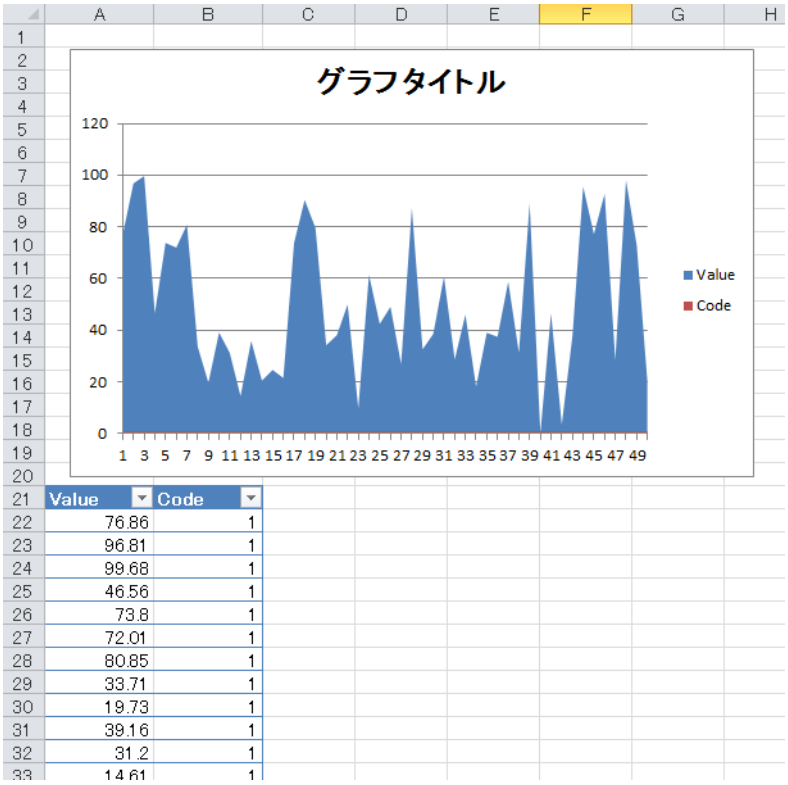
1. プログラムのロジックタブで、イベントロジックユニットを作成し、データ出力イベントを発行するロジックを定義します。
2. **パラメータ**欄からズームし、パラメータのタイプを定義することもできます。この場合、以下の値を持つことができます。
   • タイプが 0 または設定されていない場合、データ出力ウィザードが起動されます。
   • タイプが 1 の場合、データソースのデータは Excel ファイルに出力されます。
   • タイプが 2 の場合、データソースのデータはチャートに表示されます。画面が表示され、エンドユーザは、チャート内でどのカラムを表示させるか、またはどのようなタイプのチャートを表示させるかを選択することができます。



3. このイベントを発行するボタンとタイプのパラメータを指定できるチェックボックスを定義します。



4. プログラムが実行されると、エンドユーザはデータを Excel ファイルに出力するためにボタンをクリックすることができます。[チャートを含む] チェックボックスが選択されていると、チャートが表示されます。



5. OK をクリックすると、データとチャートは Excel ファイルに出力されます。

サポートバージョン :3.2

［このページは意図的に空白にしています。］

第 22 章 : マージ

テキストファイルにデータをマージするには

マージは、Magic xpa の機能の中でも有益で多用途な機能の 1 つです。Web アプリケーションを作成する場合に利用されるだけでなく、XML や RTF、または HTML などのテキストタイプのファイルにデータをマージさせる場合にも使用できます。

マージを行うタスクは、データを使用してループし、データを出力するための**フォーム出力**処理コマンドを使用するため、帳票の出力用タスクに似ています。帳票用タスクの作成方法については、第 21 章 :「帳票を作成するには」（493 ページ）を参照してください。

このセクションでは、マージ機能の基本的な作成方法について説明します。より詳細な内容は、以降のセクションで説明いたします。

データをマージするのための基本的な手順は以下の通りです。

1. テキストのテンプレートを作成します。
2. 出力ファイルを指定します。
3. マージフォームを作成します。
4. タグを選択します。
5. 各タグをデータと関連付けます。
6. マージフォームを出力します。

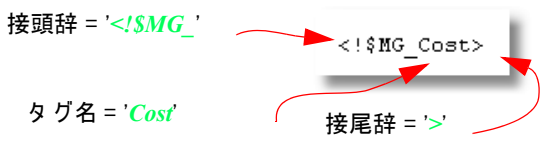
次に各ステップの詳細を説明します。

1. テキストのテンプレートを作成する

```
1
2  Dear <!$MG_Guest>
3
4  これは、在庫のあるDVDのリストです。<!$MGIF_KeywordSpecified>次のキーワードが含まれています <!$MG_Ke
5
6  もしどちらの注文かを伝えた場合、電子メールで回答します。
7  AAA DVDレンタルへお掛けくださいましたありがとうございます。
8
9  <!$MG_Salesrep>(<!$MG_Salesphone>)
10
11 ================================================================================
12      ID番号          タイトル                          価格
13 ================================================================================
14 <!$MGREPEAT>
15      <!$MG_SN>       <!$MG_MovieTitle>             <!$MG_Cost>
16 <!$MGENDREPEAT>
17 ================================================================================
```

マージを作成する上での最初の作業は、テンプレートを作成することです。テンプレートファイルは、編集可能なテキストファイルです。この例では、簡単なプレーンテキストを使用しています

このテンプレートを作成するには、テキストエディタで内容を入力し、DVDMerge.txt という名前で作業ディレクトリに保存します。また、Magic xpa でテンプレートを作成することもできます（「複数のタスクで 1 つのドキュメントにデータをマージするには」（535 ページ）を参照してください）。



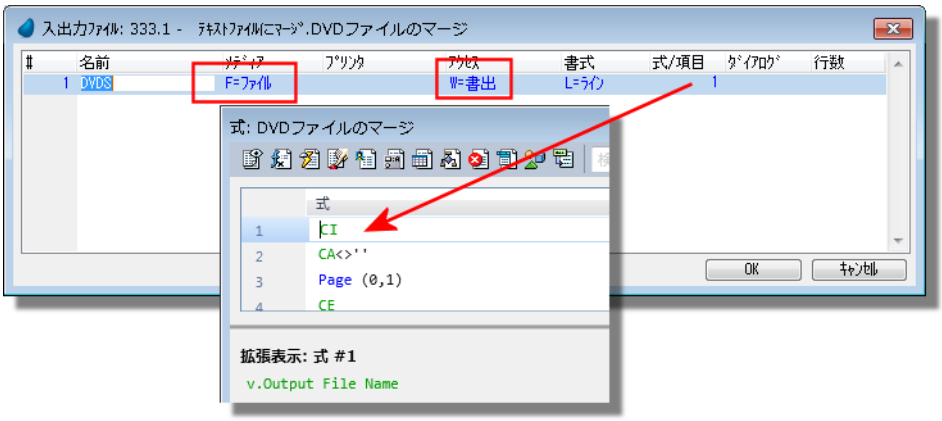
マージしたいデータ項目がある場合、実行時にその項目と置き換えるためにタグを使用します。各タグは、接頭辞（*!<$MG_*）、タグ名、および接尾辞（>）から構成されます。これらのタグは、大文字小文字を区別しています。

テンプレート内では、**<!$MGREPEAT>** タグを使用することで、データの繰り返しを指定したり、**<!$MGIF>** を使用して、条件付きのテキストを指定することができます。

参照 : 「テンプレートに置き換え可能なトークンを設定するには」（533 ページ）

「テンプレートへのデータ挿入を調整するには」（532 ページ）
「HTML テンプレートのテーブルに繰り返し可能なデータを挿入するには」（531 ページ）

2. 出力ファイルを指定する



マージを行うタスクは、データを使用してループし、データを出力するための**フォーム出力**処理コマンドを使用するため、帳票の出力用タスクに似ています。ただし、入出力ファイルやフォームの定義方法が異なります。

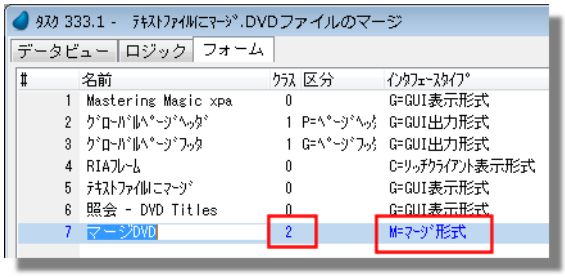
入出力ファイルの**メディア**カラムを ***F= ファイル***に設定し、**アクセス**カラムは ***W= 書出***に設定します。**式 / 項目**カラムにはファイル名を指定します。

注： Web アプリケーションでマージを利用する場合は、**メディア**カラムには ***R= リクエスタ***を設定したり、**メディア**カラムを ***V= 項目***に設定した場合は、BLOB 型項目に格納するように設定するなど、色々な組み合わせがあります。ただし、基本的な手順に違いはありません。

3. マージフォームを作成する

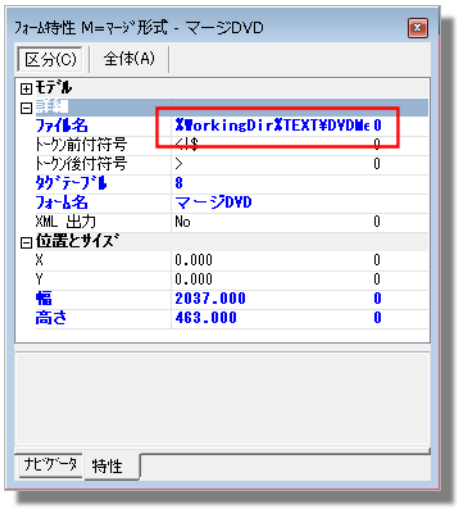
次に、マージフォームを作成します。このフォームは他のフォームと同じように**フォームエディタ**で作成します。この場合、**インタフェースタイプ**は ***M= マージ形式***に設定します。

また、他のフォームとは異なるクラスを設定する必要があります。この例では、既にグローバルな***ページヘッダ***と***フッタ***がクラス =1 で定義されているため、マージフォームのクラスは ***2*** に設定されています。



4. マージフォームの特性を設定する

次に、**マージ形式**フォームの**フォーム特性**（*Alt+Enter*）を開きます。



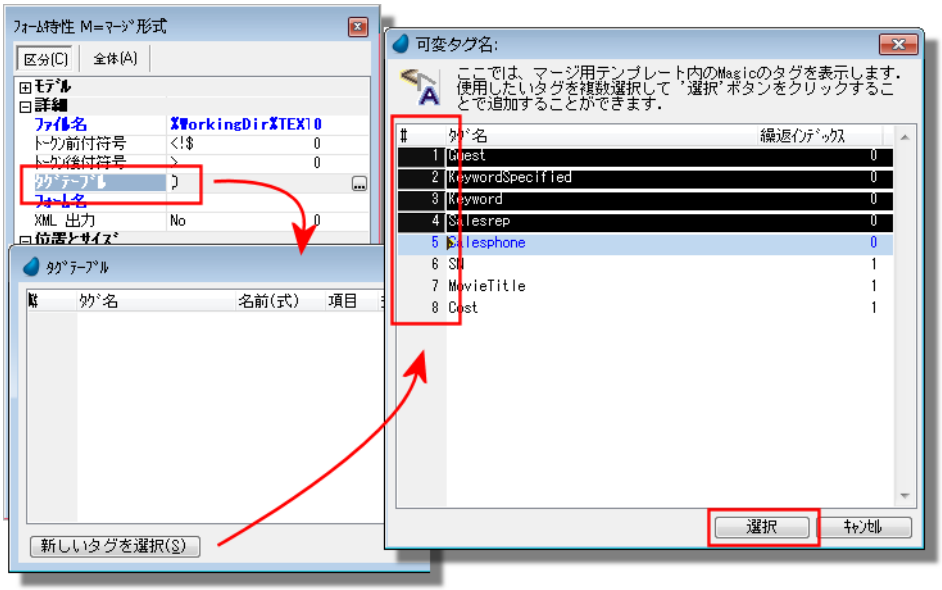
ここには、**ファイル**特性にテンプレートファイル名を設定します。テンプレートのパスに論理名を使用することで、システム環境に依存しないようにすることができます。

ここにテンプレートファイルを指定すると、**フォームエディタ**の**名前**カラムで**ズーム**することでテンプレートを編集することができます。この操作は、テンプレート名が正しく設定されているかどうかを確認することにもなります。

また、**トークン前付符号**特性と**トークン後付符号**特性はここで変更することができます。変更した場合、テンプレート内に定義されたタグは異なるものとして扱われます。例えば、**トークン前付符号**特性を *< ! $* から *<#%* に変更した場合、Cost タグは、**<!$MG_Cost>** から **<#%MG_Cost>** に変更する必要があります。保守の観点から、本当に必要でない限り、**トークン前付符号**特性と**トークン後付符号**特性を変更しないようにしてください。

ファイル名を指定したら、**タグテーブル**特性から、**ズーム**してください。

5. タグを選択する

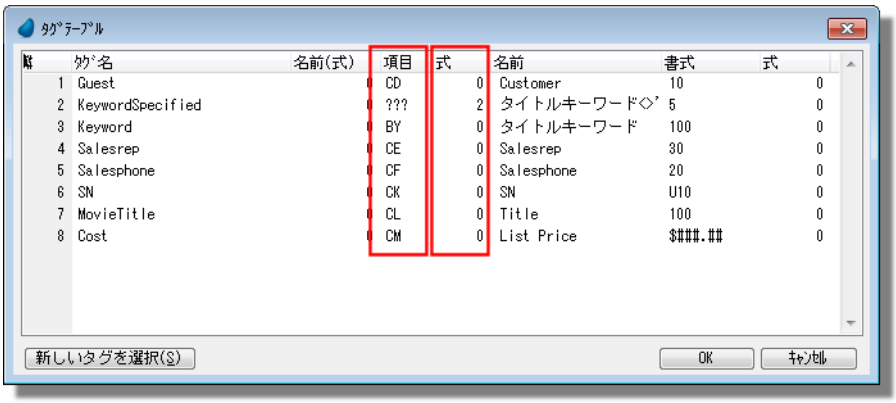


次に、タグを選択する必要があります。以下の手順で選択します。

1. **フォーム特性**の**タグテーブル**特性に移動し、**ズーム**（**F5**、または**ダブルクリック**）します。新規フォームの場合、タグテーブルは空のままです。
2. 新しいタグの選択ボタンを**クリック**し、テンプレートに定義されているタグ名のリストを表示します。
   タグのリストが表示されない場合、以下のような原因が考えられます。
   - すでにすべてのタグが選択されている。この場合は、ビープ音が鳴り、画面の下に「テンプレートファイルに新しいタグが見つかりませんでした」というメッセージが表示されます。
   - タグ名に誤りがある（**トークン前付符号**特性や**トークン後付符号**特性がテンプレートに定義されたタグ名と合っていない）。
   - テンプレートファイル名に誤りがある。ファイル名が正しい場合、**フォームエディタ**から**ズーム**することでファイルが表示されるため、この方法で確認できます。
3. 最初のカラム（行番号が表示されています）を**クリック**することで、タグを選択できます。タグを**クリック**すると黒く反転表示されます。***Ctrl+* クリック**を使用して複数のタグを選択することができます。すべてのタグが選択されたら、選択ボタンを**クリック**します。
4. 選択されたすべてのタグは**タグ**テーブルに表示され、次の手順に進むことができます。

注： タグ名を直接入力することもできますが、選択した方が簡単で確実です。

6. 各タグをデータと関連付ける



前の手順が正しく行われた場合、**タグ**テーブルにタグが定義されます。次に、各タグと値と関連付ける必要があります。値は項目または式のどちらかで指定します。

項目を指定する場合は、**項目**カラムから**ズーム**し、**項目**テーブルから選択します。

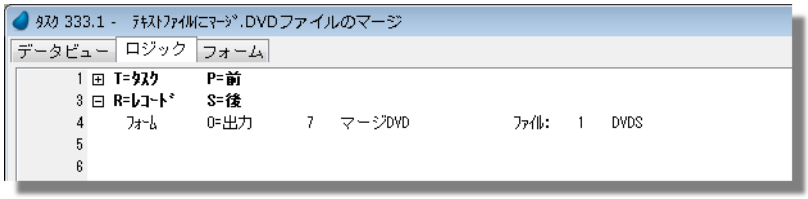
式を指定する場合は、**式**カラム（**項目**カラムのとなり）から**ズーム**します。

この作業は、帳票用にデータを選択する場合と同じではありますが、いくつか異なる点もあります。大きな違いの 1 つは、データが自動的に調整されることです。

例えば、タイトル長が 100 文字であっても、それがマージされると、100 桁の文字として出力されません。

別の違いは、マージの場合、一般にヘッダやフッターフォームがないことです。すべてのデータは 1 つのテンプレートとマージされます。MGREPEAT タグは、繰り返す要素がどこにあるかを指定します。

7. マージフォームを出力する



最後に、フォームの出力処理を定義する必要があります。通常は、処理される各レコード毎に 1 回のデータ出力が行われるため、**レコード後**で定義します。

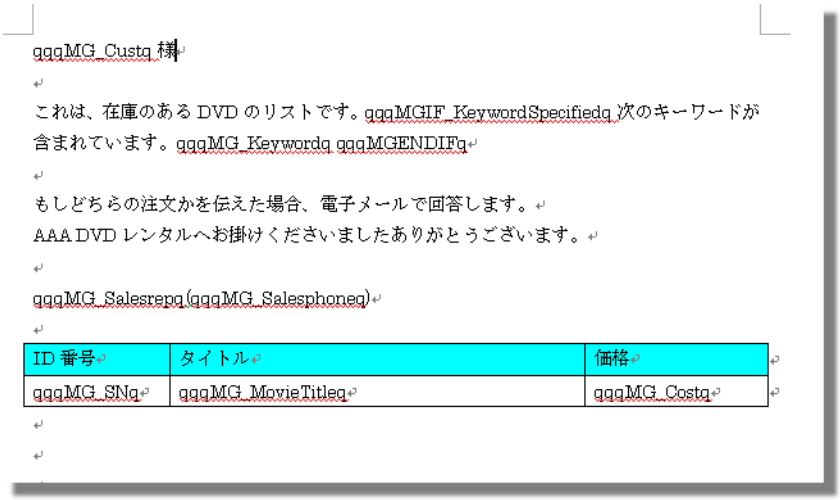
これにより以下のような結果が出力されます。

```
1
2  Frank Smit様
3
4  これは、在庫のあるDVDのリストです。
5
6  もしどちらの注文かを伝えた場合、電子メールで回答します。
7  AAA DVDレンタルへお掛けくださいましたありがとうございます。
8
9  Jill(555.555.1212)
10
11 ================================================================================
12      ID番号         タイトル                                  価格
13 ================================================================================
14      0784012717     The Boys From Brazil                        $ 69.98
15      B00003CWLF     Anna and the King                         $ 14.98
16      B00003CXCT     Star Wars Trilogy (Widescreen Edition)                  $ 69.98
17      B000052210     The X-Files - Fight the Future                $  9.98
18      B000053VB4     Mystic Pizza                    $ 14.99
19      B000077VR3     Moulin Rouge (Single Disc Edition)                $ 19.98
20      B00009AOBK     Gotcha!                   $ 14.99
21      B0000DZ3GQ     Midnight Madness                  $ 14.99
22      B0003JAONG     Cloak & Dagger                  $  9.99
23      B0006H32DY     The Palm Beach Story                $ 12.99
24 ================================================================================
```

フィールドがどのように自動的にトリミングされているかを確認してください。顧客名項目は 30 文字の長さですが、コロンがタグのすぐ隣のため、名前の後のコロンはトリミングされた名前のすぐ隣になります。

しかし、タイトルフィールドがトリミングされると、価格カラムは整列されません。価格カラムを整列させるには、テンプレート内でタブ文字を使用するか、異なる種類のフォーマット（HTML など）を使用する必要があります。

動的に Word ドキュメントを作成するには


qqqMG_Custq 様

これは、在庫のある DVD のリストです。qqqMGIF_KeywordSpecifiedq 次のキーワードが含まれています。qqqMG_Keywordq qqqMGENDIFq

もしどちらの注文かを伝えた場合、電子メールで回答します。
AAA DVD レンタルへお掛けくださいましたありがとうございます。

qqqMG_Salesrepq(qqqMG_Salesphoneq)

ID 番号	タイトル	価格
qqqMG_SNq	qqqMG_MovieTitleq	qqqMG_Costq

1. 必要な文書を Word で作成します。
2. タグを追加したい場所に、特殊な文字列（qqqMG_Custq など）を追加します。特殊文字は次の手順で Word によって変換されるため、**<!$MG_Cust>** というような書式は使用できません。
3. 文書を HTML 形式で保存し、Word を終了します。

```
 </tr>
<!$MGREPEAT>
 <tr style='mso-yfti-irow:1;mso-yfti-lastrow:yes'>
  <td width=119 valign=top style='width:89.6pt;border:solid windowtext 1.0pt;
  border-top:none;mso-border-top-alt:solid windowtext .5pt;mso-border-alt:solid windowtext .5pt;
  padding:0mm 5.4pt 0mm 5.4pt'>
  <p class=MsoNormal><span class=SpellE><span lang=EN-US><!$MG_SN></span></span></p>
  </td>
  <td width=384 valign=top style='width:288.0pt;border-top:none;border-left:
  none;border-bottom:solid windowtext 1.0pt;border-right:solid windowtext 1.0pt;
  mso-border-top-alt:solid windowtext .5pt;mso-border-left-alt:solid windowtext .5pt;
  mso-border-alt:solid windowtext .5pt;padding:0mm 5.4pt 0mm 5.4pt'>
  <p class=MsoNormal><span class=SpellE><span lang=EN-US><!$MG_MovieTitle></span></span></p>
```

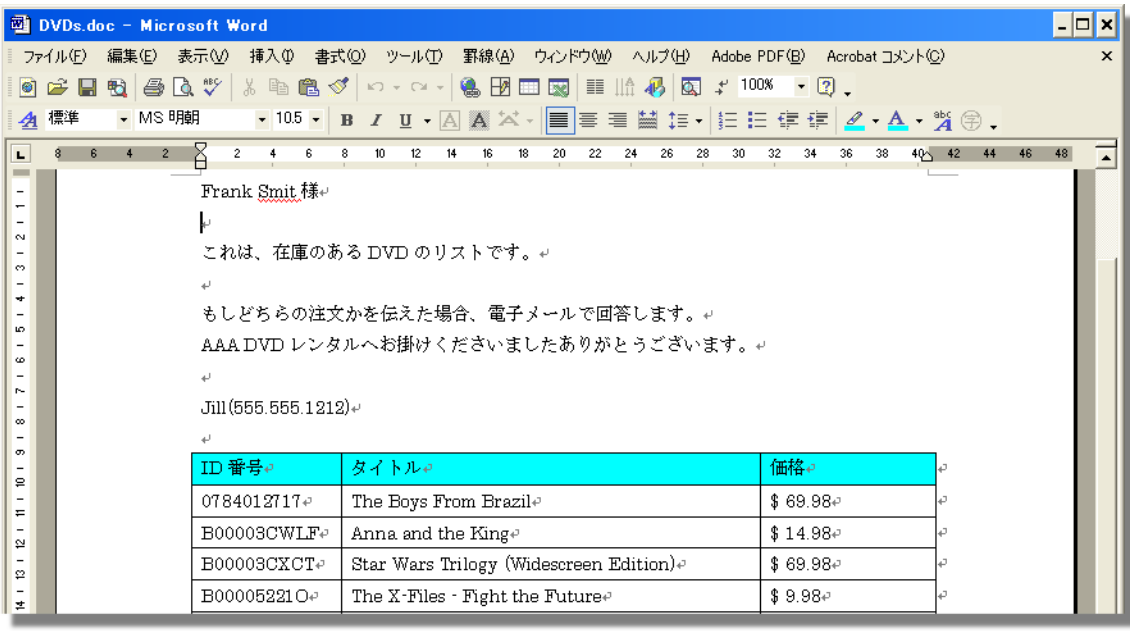
4. テキストエディタで HTML を編集します。qqqMG_Custq のような特別なタグを **<!$MG_Cust>** などの Magic タグに置き換えます。必要に応じて、MGREPEAT と MGIF を追加します。

ヒント : *qqq のようなユニークな文字列を使用した場合、検索 / 置換コマンドを使用して作業することができます。*

5. マージテンプレートとしてこの HTM ファイルを使用します。
6. 出力ファイル名の拡張子に、DOC を使用します。

これで文書が作成すると、Word の文書として Microsoft Word でオープンできるようになります。

注： この方法は、RTF 形式に対しても利用できます。



動的に Excel ドキュメントを作成するには

1. 必要な文書を Excel で作成します。
2. タグを追加したい場所に、特殊な文字列（qqqMG_Custxx など）を追加します。特殊文字は次の手順で Excel によって変換されるため、**<!$MG_Cust>** というような書式は使用できません。
3. 文書を HTML 形式で保存し、Excel を終了します。

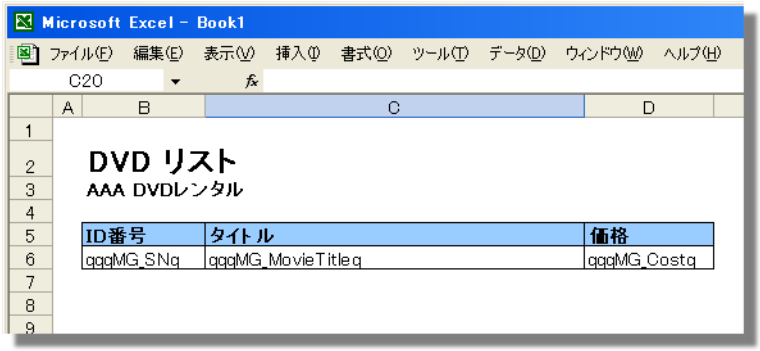
```
 </tr>
<!$MGREPEAT>
 <tr height=18 style='height:13.5pt'>
  <td height=18 style='height:13.5pt'></td>
  <td class=xl24 style='border-top:none'><!$MG_SN></td>
  <td class=xl24 style='border-top:none;border-left:none' x:str="<!$MG_MovieTitle> "><!$MG_MovieTitl
  <td class=xl24 style='border-top:none;border-left:none'><!$MG_Cost></td>
  <td></td>
 </tr>
<!$MGENDREPEAT>
 <tr height=18 style='height:13.5pt'>
  <td height=18 colspan=5 style='height:13.5pt;mso-ignore:colspan'></td>
```

4. テキストエディタで HTML を編集します。qqqMG_SNxxx のような特別なタグを **<!$MG_SN>** などの Magic タグに置き換えます。必要に応じて、MGREPEAT と MGIF を追加します。

ヒント : *qqq のようなユニークな文字列を使用した場合、検索 / 置換コマンドを使用して作業することができます。*

5. マージテンプレートとしてこの HTM ファイルを使用します。
6. 出力ファイル名の拡張子に、XLS を使用します。

これで文書が作成されると、Excel の文書として Excel でオープンできるようになります。



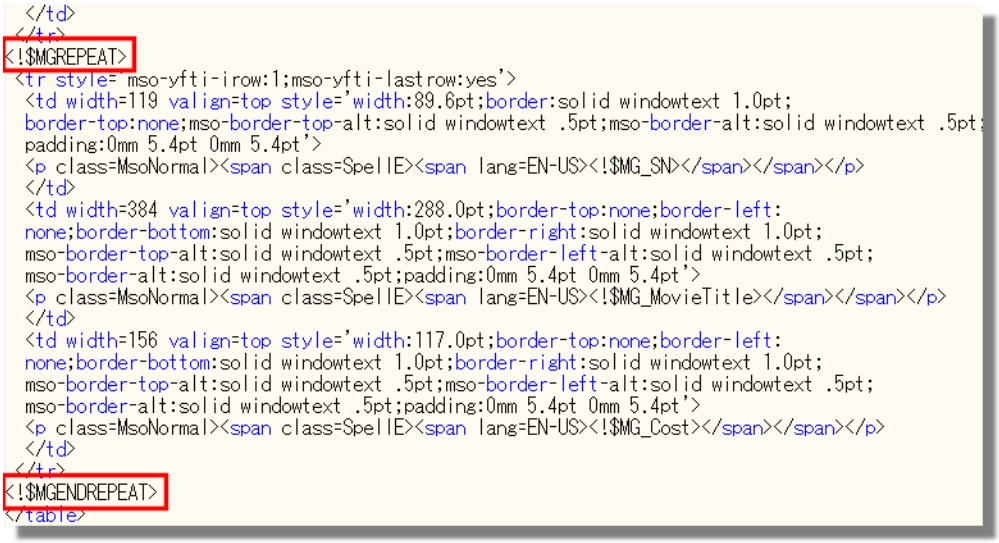
テンプレートに繰り返し可能なデータを挿入するには

```
10
11 ================================================================
12     ID番号          タイトル                          価格
13 ================================================================
14 <!$MGREPEAT>
15     <!$MG_SN>      <!$MG_MovieTitle
16 <!$MGENDREPEAT>
17 ================================
18
```

```
11 ================================================================
12     ID番号          タイトル                          価格
13 ================================================================
14     0784012717      The Boys From Brazil              $ 69.98
15     B00003CWLF      Anna and the King               $ 14.98
16     B00003CXCT      Star Wars Trilogy (Widescreen Edition)
17     B000052210      The X-Files - Fight the Future
18     B000053VB4      Mystic Pizza                 $ 14.99
19     B000077VR3      Moulin Rouge (Single Disc Edition)
20     B00009AOBK      Gotcha!                 $ 14.99
21     B0000DZ3GQ      Midnight Madness              $ 14.99
22     B0003JAONG      Cloak & Dagger               $  9.99
23     B0006H32DY      The Palm Beach Story              $ 12.99
```

繰り返し可能なデータは **<!$MGREPEAT>** タグによって処理されます。**<!$MGREPEAT>** と **<!$MGENDREPEAT>** 間にある内容は、フォームが出力されるたびに繰り返されます。従って、この例において、レコード後の中でフォーム出力処理コマンドが実行されています。フォームは、各タグ（SN、タイトル、および価格）を更新するため、これらの内容が繰り返し出力されます。これは、GUI 出力形式フォームでのテーブルコントロールの動作に似ています。

HTML テンプレートのテーブルに繰り返し可能なデータを挿入するには



```
  </td>
 </tr>
<!$MGREPEAT>
 <tr style='mso-yfti-irow:1;mso-yfti-lastrow:yes'>
  <td width=119 valign=top style='width:89.6pt;border:solid windowtext 1.0pt;
  border-top:none;mso-border-top-alt:solid windowtext .5pt;mso-border-alt:solid windowtext .5pt;
  padding:0mm 5.4pt 0mm 5.4pt'>
  <p class=MsoNormal><span class=SpellE><span lang=EN-US><!$MG_SN></span></span></p>
  </td>
  <td width=384 valign=top style='width:288.0pt;border-top:none;border-left:
  none;border-bottom:solid windowtext 1.0pt;border-right:solid windowtext 1.0pt;
  mso-border-top-alt:solid windowtext .5pt;mso-border-left-alt:solid windowtext .5pt;
  mso-border-alt:solid windowtext .5pt;padding:0mm 5.4pt 0mm 5.4pt'>
  <p class=MsoNormal><span class=SpellE><span lang=EN-US><!$MG_MovieTitle></span></span></p>
  </td>
  <td width=156 valign=top style='width:117.0pt;border-top:none;border-left:
  none;border-bottom:solid windowtext 1.0pt;border-right:solid windowtext 1.0pt;
  mso-border-top-alt:solid windowtext .5pt;mso-border-left-alt:solid windowtext .5pt;
  mso-border-alt:solid windowtext .5pt;padding:0mm 5.4pt 0mm 5.4pt'>
  <p class=MsoNormal><span class=SpellE><span lang=EN-US><!$MG_Cost></span></span></p>
  </td>
 </tr>
<!$MGENDREPEAT>
</table>
```

HTML 内でのテーブルの基本的なフォーマットは以下の通りです。

```
<table>
  <tr >
    <td> ..Header stuff ..... </td>
  </tr>
  <tr >
    <td> ..Detail line stuff ... </td>
  </tr>
</table>
```

テーブルをマージテンプレートに変換するための秘訣は、2 つの行を定義することです。一つはヘッダ用で、もう一つは MGREPEAT エリア用です。

最も簡単な方法は、現在使用しているツール（Expression Web や Excel、Word）で HTML を編集することです。この 2 つの行以外を削除します。

そして、テキストエディタを使用して、**<!$MGREPEAT>** と **<!$MGENDREPEAT>** で最後のテーブル行を囲みます。これにより以下のようになります。

```
<table>
  <tr >
    <td> ..Header stuff ..... </td>
  </tr>
  <!$MGREPEAT>
  <tr >
    <td> ..Detail line stuff ... </td>
  </tr>
  <!$MGENDREPEAT>
</table>
```

テンプレートへのデータ挿入を調整するには

一定の条件が満たした場合だけ、データを挿入したい場合があります。このような場合、MGIF タグを使用します。MGIF タグは条件付きのデータを取り囲むように定義します。例えば以下のように定義します。

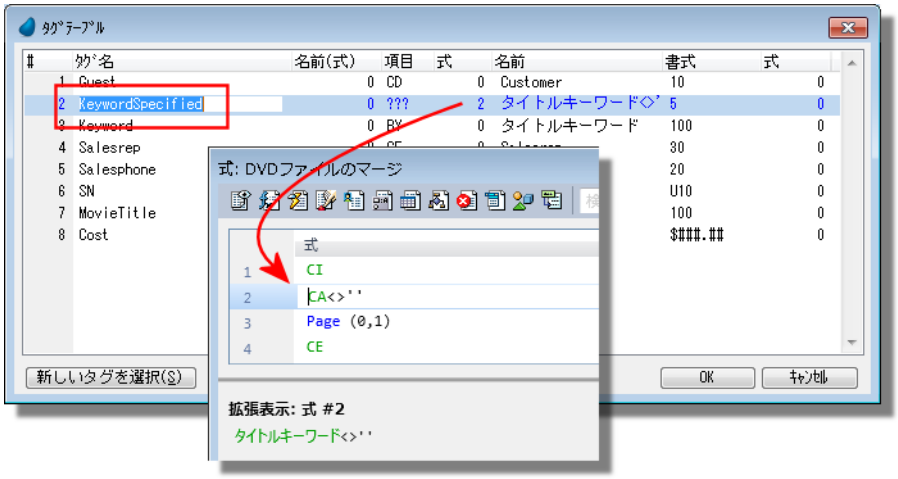
> これは、在庫のある DVD のリストです。<!$MGIF_KeywordSpecified> 次のキーワードが含まれています <!$MG_Keyword><!$MGENDIF>。

KeywordSpecified タグが True の場合、テキスト「次のキーワードが含まれています <!$MG_Keyword>」がテキスト内に含まれ、False の場合は無視され、以下のテキストとして扱われます。

> 「これは、在庫のある DVD のリストです。」

条件を指定する

Magic xpa で、KeywordSpecified が True かどうかを判定するデータを定義する必要があります。これはタグテーブルで行います。



タグ名は、MGIF タグを使用します。MGIF タグは以下のテキストを読み込みます。

```
<!$MGIF_KeywordSpecified>
```

式は、論理値として扱います。これにより、エンドユーザがキーワードを入力したかどうかを確認することができます。キーワードが入力された場合、項目「*CA*」は空白にならず、式は True として評価されます。

テンプレートに置き換え可能なトークンを設定するには

置き換え可能なトークンの最も簡単な設定方法は以下の通りです。

1. よく利用しているエディタ（例えば Dreamweaver や Excel または Word）を使用して、テンプレートを編集します。
2. 置き換え可能なトークンを設定したい場所に、qqqMG_Custxx のような簡単に分かる文字列を入力します。
3. Excel か Word を使用している場合、結果を HTML 形式で保存します。
4. 文字列を **<!$MG_Cust>** などのトークンと変更します。

この例は、「動的に Word ドキュメントを作成するには」（527 ページ）を参照してください。

HTML タグとマージのトークンを区別するには

```
<!$MGREPEAT>
 <tr height=18 style='height:13.5pt'>
  <td height=18 style='height:13.5pt'></td>
  <td class=xl24 style='border-top:none'><!$MG_SN></td>
  <td class=xl24 style='border-top:none;border-left:none' x:str="<!$MG_MovieTitle> "><!$MG_MovieTitle>
  <td class=xl24 style='border-top:none;border-left:none'><!$MG_Cost></td>
  <td></td>
 </tr>
<!$MGENDREPEAT>
```

HTML タグとマージトークンは、同じように前後を **< >** によって囲まれています。しかし、マージトークンは特定の文字列（通常は、**<!$MG_**）から始まります。**フォーム特性**の値を変更することで最初の 3 文字を指定することができますが、MG_ は Magic xpa で使用する固定の文字として設定されています。

従って、上記のコードでは、マージトークンがすべて **<!$MG** で始まるため、**<tr>** と **<td>** などの HTML タグとマージトークンを区別することができます。

複数のタスクで 1 つのドキュメントにデータをマージするには

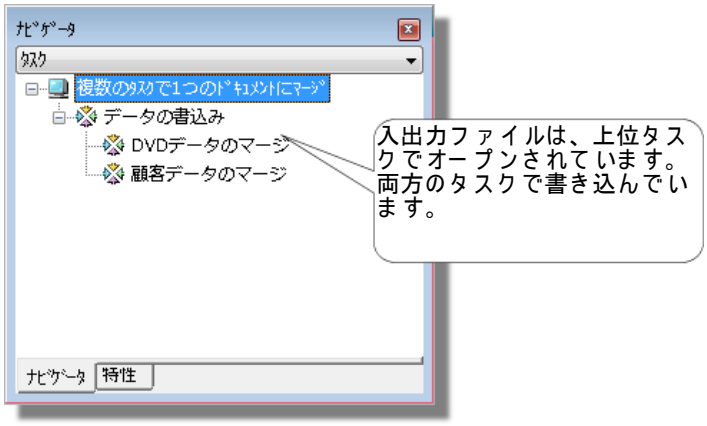
この場合、以下のようなタスクの定義方法によって異なる対応が必要です。

- 各タスクが同じプログラムに定義されており、同じタスクのサブタスクとなっている場合
- タスクが異なるプログラムで定義されている場合

これらについて以下で説明します。

注： この方法は、帳票を印刷するために使用された方法と同じです。第 21 章：「複数のプログラムで同じ入出力ファイルを使用して出力するには」（505 ページ）を参照してください。

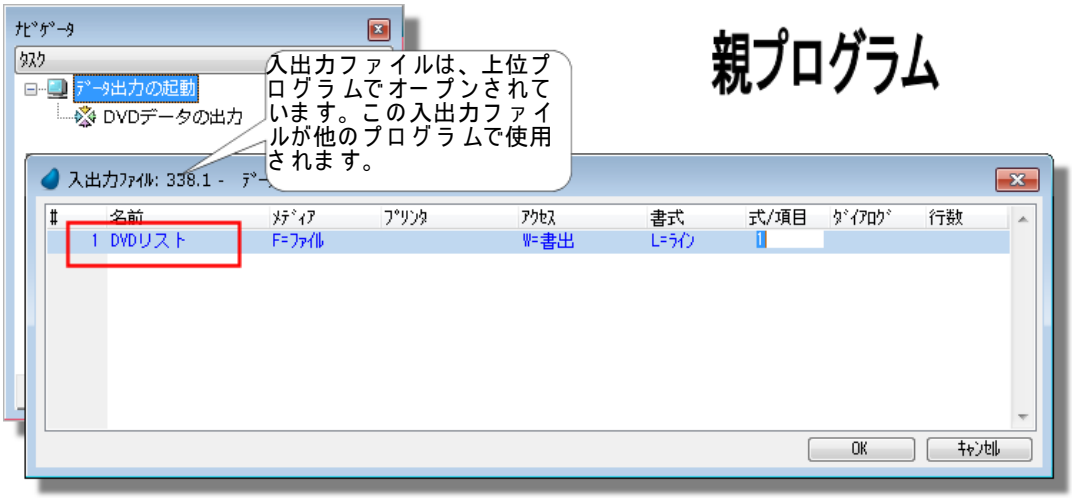
同じプログラム内の２つのタスクからのデータをマージする



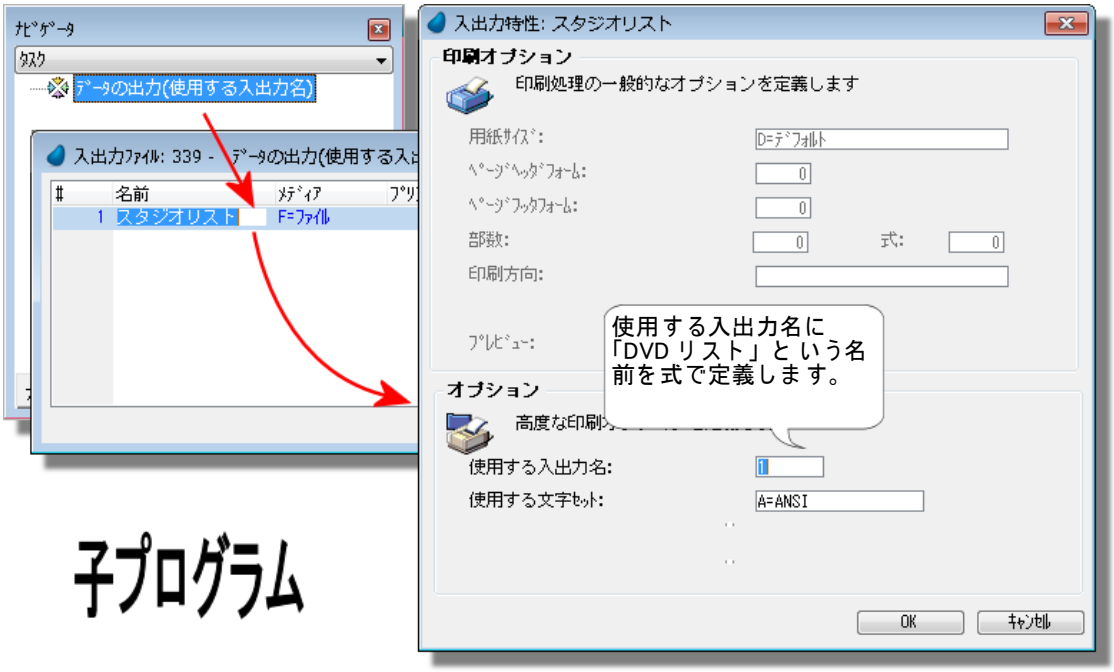
1 つのドキュメントにデータをマージさせる必要がある２つのタスクが同じプログラム内に定義されている場合、帳票処理と同じように対応することができます。つまり、親タスクで定義された入出力ファイルを２つのタスクで共有して使用することです。

２つの異なるプログラムのデータをマージする

２つの異なるプログラムでデータをマージする場合は、**入出力特性**の**使用する入出力名**特性を使用します。以下に示すように、親タスクで入出力名を設定します。



これで、同じファイルやリクエスタに送られる出力内容を持つプログラムで、この同じ名前を使用することができます。



子プログラムは、異なるファイル名が指定されていても、親プログラムでオープンされたファイルに出力します。

注： 両方のプログラムで同じメディアタイプを指定するようにしてください。一方のプログラムでファイルを指定し、他方がプリンタになっている場合、構文チェックではエラーにはなりませんが、実行時に一貫性のない結果となります。

テンプレートを使用して、データをグループ化して出力するには

HTML テンプレートを使用してデータを出力し、データをグループ化したい場合、最もよい方法は HTML テーブルを使用することです。1 つのファイル内に複数の HTML テーブルを定義し、スタイル定義を行うことで見栄えよく出力することができます。

例えば、1 つの HTML フォーム内でグループ分けされ、２つの個別のリストがあります。

S/N	タイトル
B00003CWT6	The Lord of the Rings - The Fellowship of the Ring (Widescreen Edition)
B00005JKZV	The Lord of the Rings - The Two Towers (Widescreen Edition)
B00005JKZY	The Lord of the Rings - The Return of the King (Widescreen Edition)
B000067DNF	The Lord of the Rings - The Fellowship of the Ring (Platinum Series Special Extended Edition)

スタジオコード	スタジオ名
S001	Twentieth Century Fox Home Video
S002	Buena Vista Home Video
S003	Universal Studios
S004	Paramount
S005	Warner Home Video
S006	New Line Home Entertainment

２つの異なるタスクが、同じファイルにデータを出力するために使用されます（「複数のタスクで 1 つのドキュメントにデータをマージするには」（535 ページ））。

ここで示されるように、使用されたテンプレートは、MGREPEAT タグを使用して HTML テーブルを作成しています。

最初のタスクでは、DVD データを出力しています。ここでは、最初の 2 つのタグ（MG_SN、および MG_Title）を参照しているだけです。

２番目のタスクでは、スタジオデータを出力しています。ここでは、MG_Studio と MG_StudioName の行を書き込んでいます。

```
<table class="MGW_TableControl">
  <tr class="MGW_TableHeader">
    <td ><b>S/N</b></td>
    <td ><b>タイトル</b></td>
  </tr>
   <!$MGREPEAT>
  <tr>
    <td ><!$MG_DVD_SN></td>
    <td ><!$MG_TITLE></td>
  </tr>
   <!$MGENDREPEAT>
</table>
<p>□</p>
<table  class="MGW_TableControl">
  <tr class="MGW_TableHeader">
    <td ><b>スタジオコード</b></td>
    <td ><b>スタジオ名</b></td>
  </tr>
   <!$MGREPEAT>
  <tr>
    <td> <!$MG_STUDIO_CODE></td>
    <td ><!$MG_STUDIO_NAME></td>
  </tr>
   <!$MGENDREPEAT>
</table>
```

スタジオ：S003 Universal Studios

S/N	タイトル
B00003CWLF	Anna and the King
B00009AOBK	Gotcha!
B0003JAONG	Cloak & Dagger
B0006H32DY	The Palm Beach Story

スタジオ：S004 Paramount

S/N	タイトル
6305537321	Breakfast at Tiffany's
B00003CXCG	Sabrina
B00005ALMI	Paris When It Sizzles
B00005JKFA	Better Off Dead

また 1 つのデータソースにあるデータをグループ化するようなことも可能です。この例では、スタジオコードにもとづいて DVD データをグループ化しています。

ここでは、個別の HTML テーブルが各グループ毎に作成されています。

これで、どのようにして作成されたか理解できると思います。

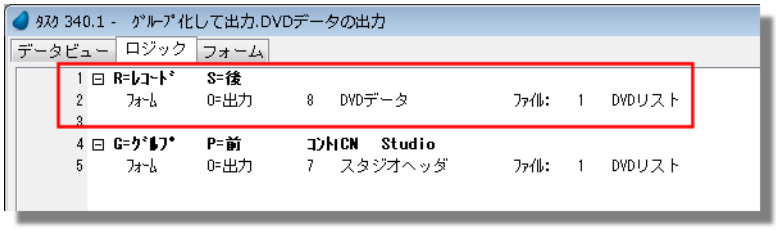
1. テンプレートを設定する

```
<FORM Name="Studios">
  <!$MGREPEAT>
  <H3> スタジオ： <!$MG_STUDIO_CODE> <!$MG_STUDIO_NAME> </H3>
  <table class="MGW_TableControl">
    <tr class="MGW_TableHeader">
      <td >S/N</td>
      <td >タイトル</td>
    </tr>
    <!$MGREPEAT>
    <tr>
      <td ><!$MG_DVD_SN></td>
      <td ><!$MG_TITLE></td>
    </tr>
    <!$MGENDREPEAT>
  </table>
<!$MGENDREPEAT>
```

テンプレートはネストされた MGREPEAT タグを使用しています。内側の MGREPEAT は、帳票の場合と同じように詳細行を作成するものです。MG_DVD_SN と MG_TITLE の２つのタグは、**レコード後**で出力されるため、1 レコードあたり 1 行が出力されます。

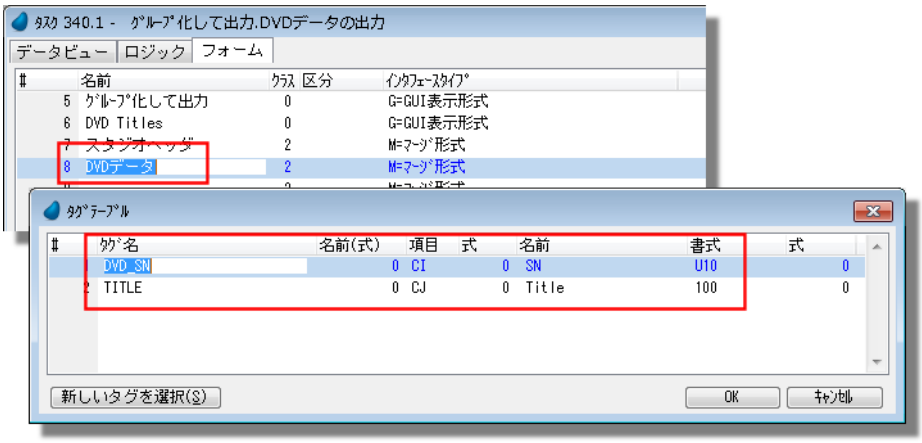
外側の MGREPEAT はスタジオ名を大文字で出力し、HTML テーブルとテーブルヘッダーの定義を処理します。

2. 詳細行を書き込む



帳票の出力処理と同じように、明細行は**レコード後**で出力されます。

しかし、出力されるフォーム（DVD データ）は、内部の MGREPEAT タグのみ参照します。従って、**フォーム出力**処理コマンドは 1 レコード分の DVD データフォームのみを書き込みます。

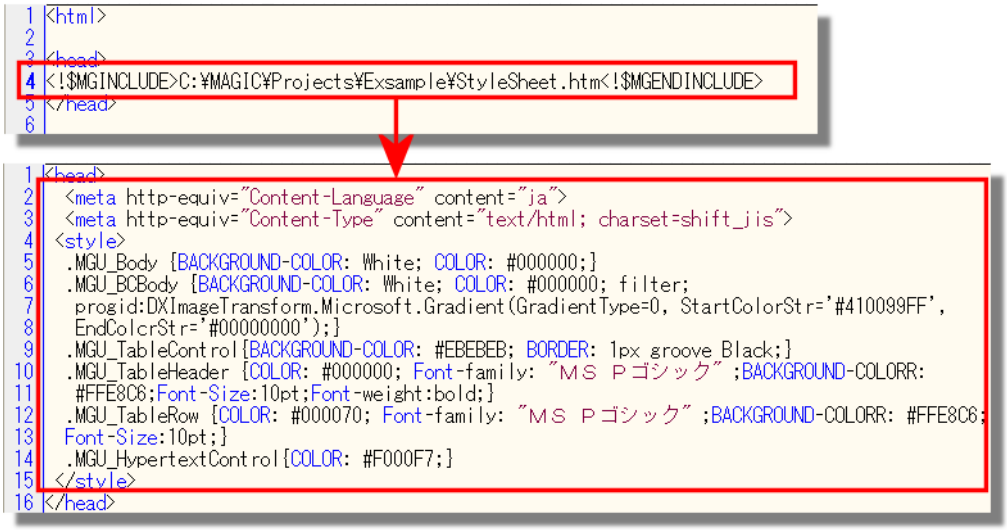


3. ヘッダを書き込む

ヘッダは、グループ後ロジックユニットで出力します。参照する項目の内容が変更された場合のみ、このロジックユニットは実行されます。データビューで Studio インデックスを使用しているため、スタジオ毎にスタジオヘッダが出力されます。GUI 出力フォームで帳票を出力する場合と同じように動作します。

スタジオヘッダのフォームは、外側の MGREPEAT タグ内のタグのみ参照します。従って、フォーム出力処理コマンドが実行されると、新しい **<H3>** ヘッダと HTML テーブルが書き込まれます。

テンプレートにファイルを埋め込むには



<!$MGINCLUDE>** タグを使用することで、ファイル全体を埋め込むことができます。この例では、実行時にファイルの中にスタイルシート（StyleSheet.htm）を埋め込むために、**<!$MGINCLUDE> を使用しています。

<!$MGINCLUDE> の構文

<!$MGINCLUDE> の構文は以下の通りです。

<!$MGINCLUDE>***<file name>***<!$MGENDINCLUDE>

<file name> には、埋め込むファイルの名前を定義します。ファイル名のためにタグを使用することができます。ファイルが完全にマージされた後で、埋め込み処理が実行されます。

マージ Web アプリケーションを作成するには

ここでは、マージ機能を利用した Web アプリケーションについて説明いたします。Magic xpa は、マージ機能使用して Web アプリケーションを開発することができます。

基本的なマージ Web アプリケーション

マージ形式を使用したプログラムの例として、商品一覧というデータテーブルの内容を表示する簡単な Web プログラムを紹介します。このプログラムは、「商品番号」をパラメータとして渡されるとその番号以降を表示します。

表示するデータソースの内容

データソースのカラムは以下の通りです。

名前	型	書式
商品番号	N= 数値	5Z
商品名	A= 文字列	20
単価	N= 数値	N7CZ

HTML ファイルを作成する

HTML オーサリングツールを使用して HTML ファイルを作成します。このファイルに Magic xpa のデータがマージされます。

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=shift_jis">
   </head>
   <body>
      <p align="center"><font size="6"> 商品一覧 </font></p>
      <hr>
         <div align="center">
            <table border="1">
               <tr>
                  <td> 商品 #</td>
                  <td> 商品名 </td>
                  <td> 単価 </td>
               </tr>
               <!$MGREPEAT>
                  <tr>
                     <td><!$MG_ID></td>
                     <td><!$MG_Name></td>
                     <td><!$MG_Price></td>
                  </tr>
               <!$MGENDREPEAT>
            </table>
         </div>
   </body>
</html>
```

マージタグ

データ置換タグ

書式 ：<!$MG_ID>

ID は任意の項目名です。この部分に Magic xpa がデータを埋め込みます。タグテーブルには ID のみが表示されます。

繰り返しタグ

書式 ：<!$MGREPEAT> ～ <!$MGENDREPEAT>

<!$MGREPEAT><!$MGENDREPEAT> に囲まれた部分が繰り返されます。このタグは、テーブルを一覧形式で表示させる場合に使用します。タグテーブルには表示されません。

条件タグ

書式 ：<!$MGIF_Active? > ～ <!$MGELSE> ～ <!$MGENDIF>

この記述で、「IF then else endif 」の構文を実現します。**Active?** は任意の名称で、「True」かどうかの判定条件として扱われます。*タグ*テーブルには **Active?** が表示され、論理型の値を割り当てます。

マージプログラムを作成する

HTML ファイルにテーブルのデータをマージするプログラムを作成します。

タスク特性

- タスクタイプ……B= バッチ
- メインテーブル……商品一覧テーブル

フォームテーブル

- フォーム形式……HTML マージ形式
- HTML ファイル……作成した HTML ファイルを指定
- トークン前付符号……<!$
- トークン後付符号……>

タグテーブル

タグテーブルには、HTML ファイルに定義されたタグ（**!$MG_** の後に記述された文字列）とデータを割り当てます。

- ID……カラム「商品番号」
- Name……カラム「商品名」
- Price……カラム「単価」

入出力テーブル

- メディア……リクエスタ

タスクを定義する

データビューエディタ

1. パラメータ項目「変数：商品番号」を定義します。書式は、テーブルのカラムの 「商品番号」と同じです。
2. メインソースののカラムを全て定義します。
3. カラム「商品番号」の範囲：最大値特性に「変数：商品番号」を式エディタで割り当てます。

レコード後

以下の内容でフォーム出力処理コマンドを定義します。

- 内容……マージフォーム
- ファイル……入出力テーブルに定義されたリクエスタ

マージプログラムを実行する

マージプログラムは、開発モードで実行確認することができません。実行モードにして、Web ブラウザより呼び出します。呼び出しには、Web ブラウザから URL を入力する方法と、他の Web ページから Form タグで指定する方法の２種類があります。

URL で呼び出す

Web ブラウザより以下のような URL を入力して呼び出します。

```
http:// [サーバー名] / [スクリプトパス] /mgrqispi.dll?APPNAME= [アプリケーション名]
&PRGNAME= [公開プログラム名] &ARGUMENTS= [パラメータ]
```

例えば以下のように指定します。

```
http://localhost/Magic3xScripts/mgrqispi.dll?APPNAME=MergeProg&PRGNAME=Test&ARGUMENTS=-
N1013
```

これによって、プログラムが呼び出され、数値「1013」がパラメータとして渡されます。

JavaScript の利用

URL での呼び出し方法を JavaScript で関数化することができます。この場合、事前に Script タグ内で関数を定義する必要があります。

```
<HTML>
......
<script>
function mgcall(prog,arg)
{
url = 'http://localhost/Magic3xScripts/mgrqispi.dll?APPNAME=MergeProg&PRGNAME=' + prog+
'&ARGUMENTS=' + arg ;
window.open (url,'MainFrame');
}
</script>
.......
<a href="javascript:mgcall('Test','-N1013')">MergeSample</a></font></p>
```

Form タグの送信で呼び出す

別の Web ページ（HTML ファイル）に Submit ボタンを配置し、ボタンがクリックされた時に呼び出すようにすることもできます。

```
<form action="http://localhost/Magic3xScripts/mgrqispi.dll" method="post">
<input type="submit" value=" 送信 " name="B1">
<input type="hidden" name="APPNAME"value="MergeProg">
<input type="hidden" name="PRGNAME"value="Test">
<input type="hidden" name="ARGUMENTS"value="-N1013">
</form>
```

これらの内容をマージ機能で動的に変更することも可能です。

Cookie を使用するには

Cookie は、Web ブラウザを実行している PC 上にファイルとして情報を保存するものです。Cookie を使用することで以下のような処理を実現させることができます。

- 訪問者がそのページに何回訪れたか記録して表示する。
- 訪問者の好みの表示方法を記録しておき、次回訪問時にそのモードで表示する。
- 掲示板やチャットで入力したユーザー名を記録しておき、次回訪問時にユーザー名の入力を省略する。
- ログインによるセッションを確立する。

ここでは、Web アプリケーションで Cookie に情報を書き込む方法と Cookie から情報を読み込む方法を説明します。

Cookie の書き込み

マージプログラムの場合、HTML ファイルに以下のような方法で Cookie の書き込み処理を記述します。

- HTML ファイルの先頭に書き込み処理を記述する。
- META タグを使用する。
- JavaScript を使用する。
- RQHTTPHEADER 関数を使用する。

HTML ファイルの先頭に 書き込み処理を記述する

HTML ファイルの先頭に以下の記述を追加します。

```
<!$MG_CookiesString>
<html>
…
</html>
```

この場合、マージ形式フォームのタグテーブルに以下のように設定します。

- タグ名……CookiesString
- 式……'set-cookie: NAME=VALUE; expires=Fri, 05-Mar-2010 04:00:00 GMT;'

META タグを使用する

META タグの中に書き込み処理を記述します。

```
<html>
<head>
<meta http-equiv="set-cookie" content="<!$MG_CookiesString>">
…
</html>
```

この場合、マージ形式フォームのタグテーブルに以下のように設定します。

- タグ名……CookiesString
- 式……'NAME=VALUE; expires=Fri, 05-Mar-2010 04:00:00 GMT;'

JavaScript を使用する

Scripts タグの中に JavaScript 形式で記述します。

```
<html>
…
<script type="text/javascript">
document.cookie = "<!$MG_CookiesString>";
</script>
…
<body>
```

- タグ名……CookiesString
- 式……'NAME=VALUE; expires=Fri, 05-Mar-2010 04:00:00 GMT;'

RqHTTPHeader 関数を使用する

タスク前にアクション処理コマンドで **RqHTTPHeader** 関数を以下ように実行します。

```
RqHTTPHeader ('set-cookie: NAME=VALUE; expires=Fri, 05-Mar-2010 04:00:00GMT;')
```

Cookie の読み込み

書き込まれた Cookie の情報を読み込むには、以下の２つの方法があります。

- **GetParam** () 関数を使用する。
- HTTP ヘッダ情報から取り出す。

GetParam 関数を使用する

Cookie に設定された内容は **GetParam** () 関数を使用することで取得することができます。例えば、以下のように Cookie が書き込まれた場合。

```
Set-Cookie: NAME=Tanaka;TIME=10:12:00; expiers=Fri, 31-Dec-2030 23:59:59;
```

GetParam(*NAME*) では、「Tanaka」が返ります。

注： Cookie の書き込みの際に、日付制限（expires）を指定しないとブラウザ終了時に Cookie の情報が削除されます。

HTTP ヘッダ情報から取り出す

また、HTTP ヘッダ情報を取り出すことで、まとめて取得することもできます。

Mgreq.ini に以下の記述を加えます。

```
HttpVarts = HTTP_COOKIE
```

この状態で、**GetParam(*HTTP_COOKIE*)** を実行すると「NAME=tanaka;TIME=10:12:00」が返ります。

セッション管理を行うには

セッション管理の必要性

HTTP プロトコルは Web サーバーとの接続を保持しません。このため、アプリケーションによっては、ブラウザからの呼び出しが同じユーザからなのかどうかをその都度チェックする必要があります。この処理をセッション管理といいます。

マージを使用した Web アプリケーションでは、セッション管理をアプリケーション側でサポートする必要があります。

セッション管理の方法

セッション管理は、セッション ID を使用して管理します。セッション ID を次のページに渡し、データベースと照合することでユーザを識別します。

セッション ID を生成する

Web サーバー側でセッション ID を発行する際、ランダムで容易に推測できない値にしてください。規則性を持った（例えば 0001、0002、0003 のような）セッション ID を発行すると、他のユーザによるなりすましが発生する危険性があります。完全にユニークなセッション ID を生成し、さらにセッション ID を暗号化するように検討してください。

例えば、「最終アクセス日時」と「ランダム値」、「ホストの IP アドレス」を組み合わせた文字列を暗号化するなどして算出します。

セッション ID を実装する

セッション ID を実装する方法として、以下の２つがあります。

- Cookie を使用する
- Hiden フィールドに埋め込む。

Cookie を使用する

Cookie にセッション ID を書き込むことで、同一 PC 上のセッション ID を管理することができます。

Hiden フィールドに埋め込む

入力フォームを Web サーバーに送信する際に利用する FORM タグの hidden フィールドにセッション ID を持たせることで、セッション管理を実現できます。Cookie を使用できない場合はこちらを利用します。

セッション ID を保存する

セッション ID は、データベースに保存します。データベースには以下の情報を保存します。

- セッション ID
- ログイン日時
- 最終更新日時
- ユーザー ID

その他の項目はアプリケーションの要求に応じて管理します。

コンテキスト管理を行うには

コンテキストオブジェクトを使用すると、アプリケーション情報を保管したり、アプリケーションのさまざまなコンポーネント間で情報を共有ですることができます。

たとえば、アプリケーションが複数の HTML ページなどで構成されているとします。これらのアプリケーションコンポーネント間でのやり取りを可能にするために、アプリケーションコンテキストオブジェクトを使用して、その情報を保管したり、取り出したりすることができます。

Magic xpa では以下の情報をコンテキスト毎に保持することができます。

- メモリテーブル
- グローバル値 （**GetParam()** 関数で取得できます。）
- **IniPut()** 関数で設定した値
- メインプログラムで定義されたグローバル変数の内容

コンテキスト管理の有効化

ブラウザクライアントでは、コンテキスト管理を Magic xpa 側の機能で行っていますが。マージアプリケーションの場合は、アプリケーション側で以下のような設定が必要になります。

タスク特性

拡張タブのコンテキストの保持を ***Yes***（または、「True」になる式）に設定します。これにより、バッチタスクが終了してもコンテキストは破棄されなくなります。

コンテキスト ID の取得

一番最初に自分のコンテクスト ID を取得する必要があります。このとき、CtxGetdId() 関数を実行することで取得できます。

プログラム起動時の URL

プログラムを実行させるための URL にコンテキスト ID（CTX=）を明示的に入れるようにします。これにより、このリクエストは指定されたコンテキストで実行されます。

（例）

```
http://localhost/Magic3xScripts/
mgrqispi.dll?appname=Samples&prgname=Test&CTX=123456789123456789
```

または、

```
<FORM Name="Items" action="http://localhost/Magic3xScripts/mgrqispi.dll">
<input type="Hidden" name="APPNAME" value="Samples">
<input type="Hidden" name="PRGNAME" value="Test">
<input type="Hidden" name="CTX" value="123456789123456789">
<input type=submit value=" 送信 "
</FORM>
```

コンテキスト管理プログラム

以下の図は、コンテキスト管理を行った場合のプログラムのフローを表しています。「ログインチェックプログラム」でコンテキスト保持を Yes にすることで以降の処理中の同じコンテキストを使用することができます。これにより、プログラム間のパラメータは、コンテキスト ID のみとなり。

グローバル値やグローバル変数、メモリテーブルに値を設定することで、コンテキスト内のデータの一貫性が保たれます。

クライアントの認証

コンテキストにクライアントの認証情報がない場合、クライアント（ユーザ）が異なっていても、リクエストに指定されたコンテキスト ID さえ合っていれば（割り込みして）処理を継続させることができます。

このようなことを防ぐためには、一連の処理が同じクライアントで行われたことを識別する情報が必要になります。そのための情報として HTML 環境変数を利用します。

HTML 環境変数

HTML 環境変数とは、Web サーバやリクエストを送信したクライアントの情報を Web サーバが設定する環境変数です。この値をもとにクライアントを識別することができます。クライアントに関する環境変数には以下のものがあります。

注： 「REMOTE_USER」は、エイリアスの［匿名アクセス］が有効になっている場合は取得できません。

Magic プログラムで HTML 環境変数の内容を取得するには、**GetParam()** 関数を使用します。

認証項目を指定する

クライアントの認証項目の指定は、インターネットリクエスタが参照する（同じディレクトリ内にある）MGREQ.INI ファイルで行います。

設定例

```
[REQUESTER_ENV]
…
HttpSigVars = REMOTE_ADDR
…
```

ISAPI 用のインターネットリクエスタの場合、MGREQ.INI の変更を有効にするには、IIS を再起動する必要があります。

確認

一方のクライアントの Web ブラウザからアプリケーションを呼び出します。

```
http://localhost/Magic3xScripts/mgrqispi.dll?appname=Samples&prgname=Test
```

この時のコンテキスト ID を「123456789123456789」とします。別のクライアントの Web ブラウザから同じサーバの同じアプリケーションをコンテキスト ID を指定して呼び出します。

```
http://localhost/Magic3xScripts/
mgrqispi.dll?appname=Samples&prgname=Test&CTX=123456789123456789
```

第 23 章 : メッセージング

メッセージを MSMQ に送るには

MSMQ コンポーネントを使用することで、Magic プログラム内で簡単にメッセージを MSMQ に送ることができます。メッセージを送るために、以下の 2 つの基本的なオプションが用意されています。

- ３つのプログラム（**Open Queue**、**Send Message**、**Close Queue**）を使用することができます。送信するメッセージがたくさんある場合、これらを使用することで効率よく処理ができます。
- **Quick Send** を使用することで、上記の処理を 1 つのプログラムを呼び出すことで実行させることができます。

ここでは、両方の方法について説明します。

必要条件 : 利用する PC にあらかじめ MSMQ をインストールしておく必要があります。また、Magic xpa の MSMQ コンポーネントがアプリケーションで利用できるように設定する必要があります。「PC に MSMQ を設定するには」（561 ページ）を参照してください。

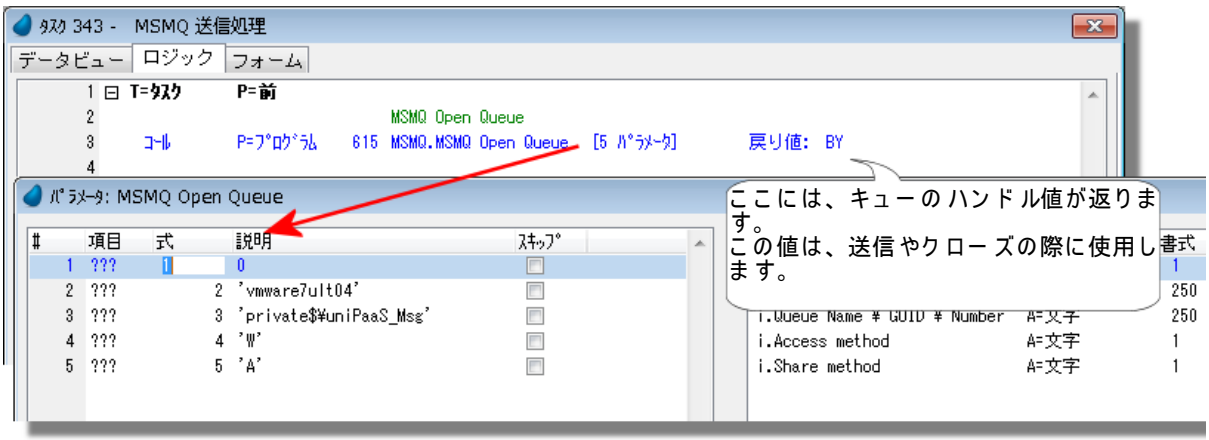
Open/Send/Close を使用する

最初の方法は、以下の 3 つの MSMQ プログラムを使用します。

- Open Queue：送信キューをオープンし、そのハンドル値を返します。返されたハンドル値を使用してメッセージを送信します。
- Send Message：キューのハンドル値を指定してメッセージを送信します。
- Close Queue：キューをクローズします。

これらの 3 つのプログラムに関する詳細情報は、Magic xpa の『リファレンスヘルプ』を参照してください。ここでは、簡単な例を紹介します。

1. メッセージキューをオープンする



メッセージングキューを使用する前に、オープンする必要があります。キューをオープンする際に、そのキューは送信用なのか受信用なのかを指定します。同時に両方の処理が必要な場合、2 つの異なるキューを用意する必要があります。

同様に、各宛先毎にユニークなキューをオープンする必要があります。この例で示すように、1 台の特定の PC を表すアドレスや具体的な IP アドレスを指定してキューをオープンするか、パブリック、または専用（プライベート）キューを使用することができます。

最初の 3 つのパラメータはキューを指定します。

キューには様々なタイプがあり、指定される PC のアドレスとキュー名は、どんな種類のキューにアクセスするかに依存します。

例：

キューの書式名	PC のアドレス	キュー名
0: PC のホスト名	Windows に設定されているコンピュータ名	Windows の MSMQ コンポーネントに設定されたキュー名
1: IP アドレス	IP アドレス	
2: パブリック名	不要	キューの GUID
3: プライベート名	キュー番号	キュー番号

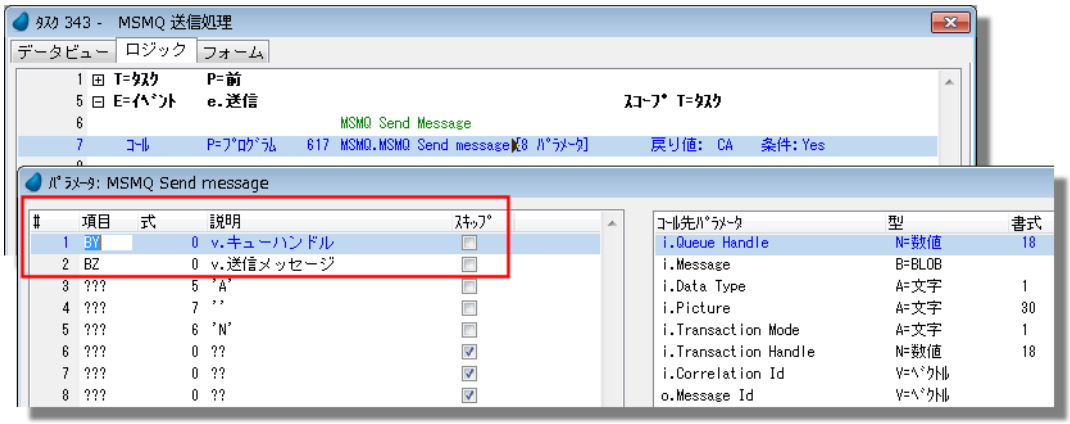
4 番目のパラメータはアクセス方式を指定します。もしそれが送信用のキューの場合、「書込」を表す ***W*** を指定します。他のプログラムとキューの書込を共有させるため、５番目のパラメータでは共有指定として ***A*** を設定します。

Open Queue プログラムがキューのオープンに成功したら、正の整数（キューのハンドル値）を返します。この値は、メッセージの送信やキューをクローズする際に使用するため、記憶しておく必要があります。

Open Queue プログラムが処理に失敗した場合、それは負の整数を返します。エラーメッセージを取得する場合は、この値を使用します。**MSMQ.PublicError** イベントを使用することでエラーメッセージを取得することができます（「メッセージングのエラーを捕捉するには」（559 ページ）を参照）。

ヒント : ***この例では、読みやすさを考慮してコンピュータ名を直接設定していますが、OSEnvGet('COMPUTERNAME') 関数を使用して実行中の PC のコンピュータ名を取得したり、データソースにコンピュータ名や IP アドレスを格納して使用してください。***

2. メッセージを送信する

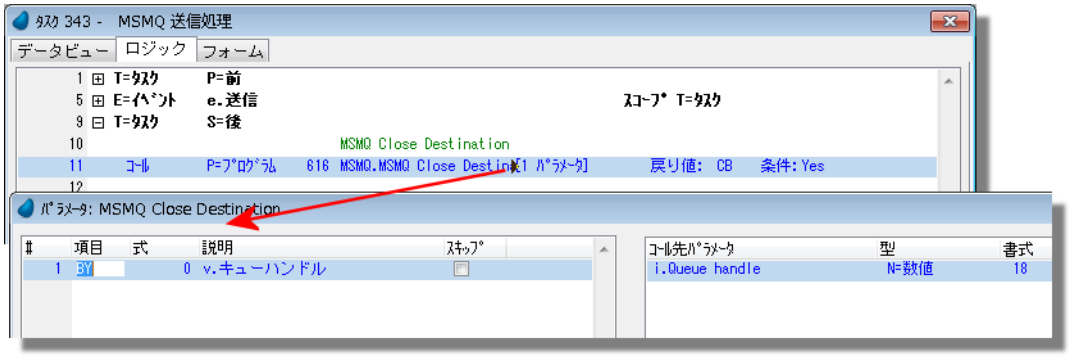


簡単なメッセージを送信するために、以下のパラメータを使用します。

- ***Queue handle***：キューをオープンした際に取得されたハンドル値
- ***Message***：送信したいメッセージ。BLOB 型のデータを指定します。
- ***Data type***：簡単なテキストメッセージとして、***A*** を指定します。
- ***Picture***：このパラメータは、数値データの場合のみ必要です。
- ***Transaction mode***：トランザクションを使用していないため、***N*** を指定しています。
- ***Transaction handle***：トランザクションを使用していないため必要ありません。

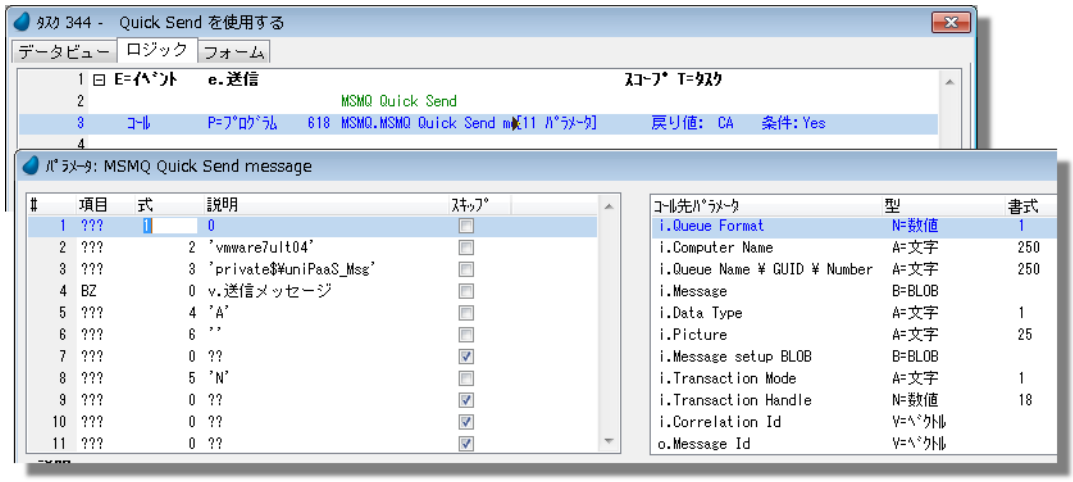
再度キューをオープンし直すことなく複数のメッセージを送信することができます。

3. キューをクローズする



メッセージの送信後にキューをクローズします。キューをオープンした際に取得したハンドル値を使用します。

Quick Send を使用する



Quick Send プログラムを使用することで、キューのオープン、送信、キューのクローズといった一連の手順を実行することなく、簡単にメッセージを送信することができます。1 つのメッセージを送信するだけであれば、こちらの方がより簡単に行うことができます。

Quick Send プログラムは、前述の 3 つのプログラムの処理を実行させるため、これらと同じぐらいの多くのパラメータを使用しています。ただし、簡単なテキストメッセージを送信する場合は、これらのパラメータの全部を指定する必要はありません。

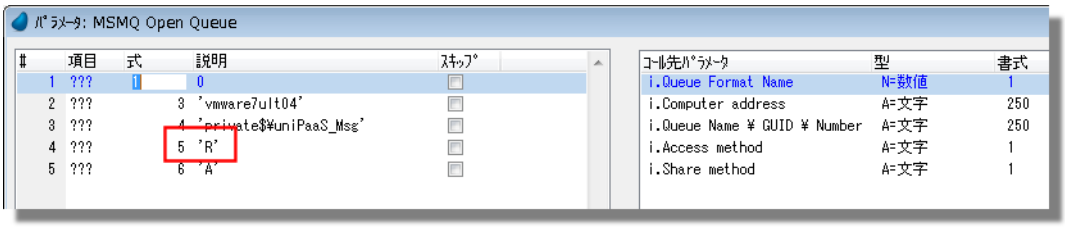
MSMQ メッセージを受信するには

MSMQ メッセージを受信するには、コンポーネントから 3 つの MSMQ プログラムを呼び出す必要があります。

- Open Queue：受信キューをオープンし、そのハンドル値を返します。返されたハンドル値を使用してメッセージを受信します。
- Read Message：キューのハンドル値を指定してメッセージを受信します。
- Close Queue：キューをクローズします。

これらの 3 つのプログラムに関する詳細情報は、Magic xpa の『リファレンスヘルプ』を参照してください。ここでは、簡単な例を紹介します。

1. メッセージキューをオープンする

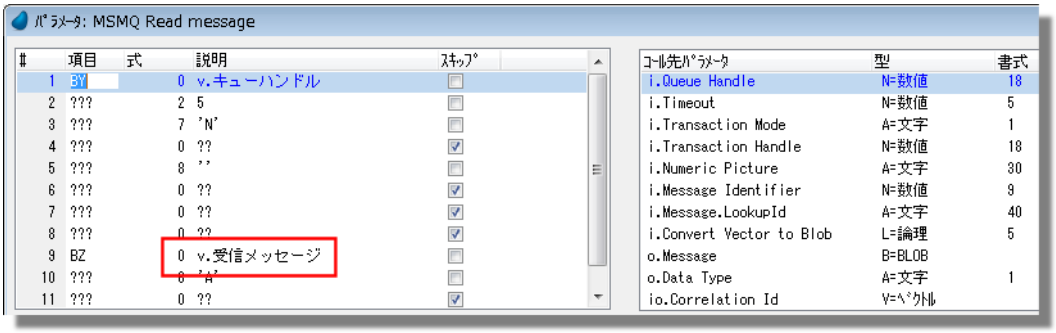


メッセージキューをオープンするには、MSMQ.Open Queue プログラム呼び出します。4 番目のパラメータで ***R*** を指定することでキューを受信用としてオープンすることができます。この指定を行うことで、読み込まれたメッセージはキューから削除されます。***V*** と指定した場合は、メッセージを読み込んでもキューからは削除されません。

これ以外のパラメータの説明は、「1. メッセージキューをオープンする」（549 ページ）を参照してください。

Open Queue プログラムがキューのオープンに成功したら、正の整数（キューのハンドル値）を返します。この値は、メッセージの受信やキューをクローズする際の最初のパラメータとして使用されます。

2. メッセージを読み込む



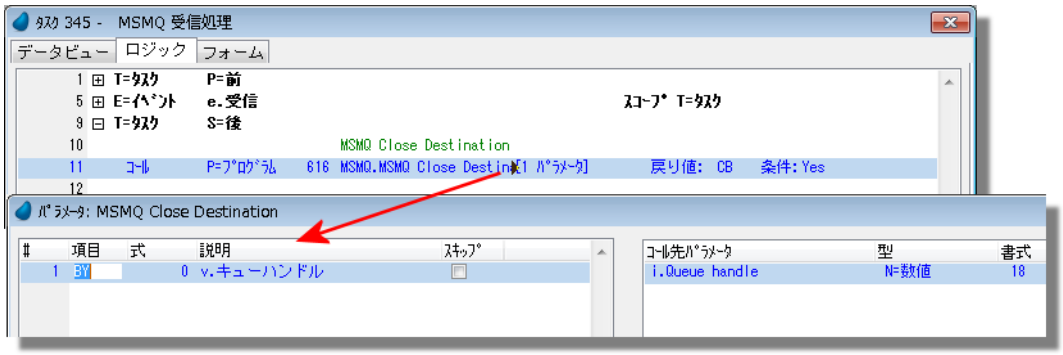
キューがオープンされると、MSMQ.Read messages プログラムを使用して、メッセージを受信することができます。プログラムが呼び出されるたびに、1 つのメッセージがメッセージキューから読み込まれ、9 番目のパラメータで指定される BLOB 項目にコピーされます。

この例のように、トランザクションを使用しない簡単なテキストメッセージの場合、パラメータのほとんどを空白（またはスキップ）にしておくことができます。

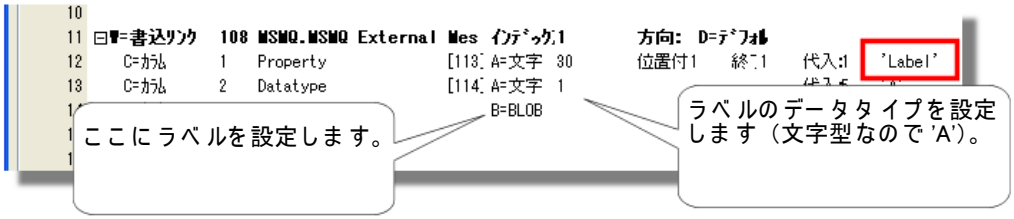
Timeout パラメータは、キューのメッセージを待つ時間を示します。メッセージがキューに送信されるまで永久に待つようにする場合、このパラメータに -1 を設定します。これで、メッセージが到着するまで、タスクの処理が停止します。これはリスナー用のタスクをどのように実現するかを指定することになります。

3. キューをクローズする

メッセージを受信したら、キューをクローズする必要があります。その際、キューをオープンした時に取得されたハンドル値を使用します。



送信メッセージにラベルを設定するには



MSMQ メッセージは、Microsoft によって定義された一連の特性を持っています。メッセージを送信する際に、Magic xpa の MSMQ コンポーネントはこれらの特性を設定する必要があります。特性は、コンポーネント内に設定する必要があります。MSMQ.External Message Set を使用することでこれらの処理を行うことができます。

MSMQ のラベル特性を設定する

1. データビューエディタで書込リンクコマンドを定義します。
2. **Property** カラムで文字列「Label」を使用してリンク定義を設定します。**代入**特性にも同じ値を設定します。
3. **Datatype** カラムを、設定するラベルのデータタイプ（通常は文字型のため、A）で更新します。
4. **Data** カラムを、設定するラベル値で更新します。

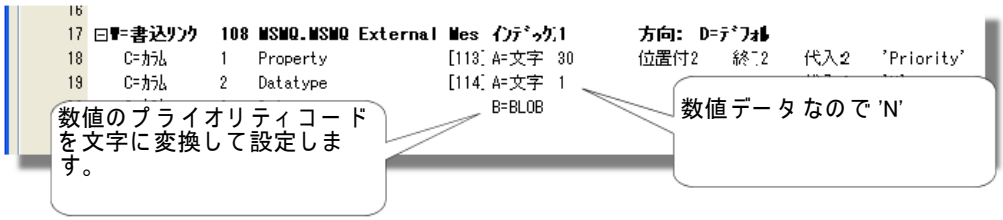
テーブルをコンポーネントに送る

次に、すべての特性を持つテーブルをコンポーネントに送る必要があります。この方法については、「外部メッセージテーブルをメッセージセットアッププログラム に送るには」（558 ページ）を参照してください。

送信メッセージにプライオリティを設定するには

MSMQ メッセージは、Microsoft によって定義された一連の特性を持っています。メッセージを送信する際に、Magic xpa の MSMQ コンポーネントはこれらの特性を設定する必要があります。特性は、コンポーネント内に設定する必要があります。MSMQ.External Message Set を使用することでこれらの処理を行うことができます。

MSMQ のプライオリティ特性を設定する

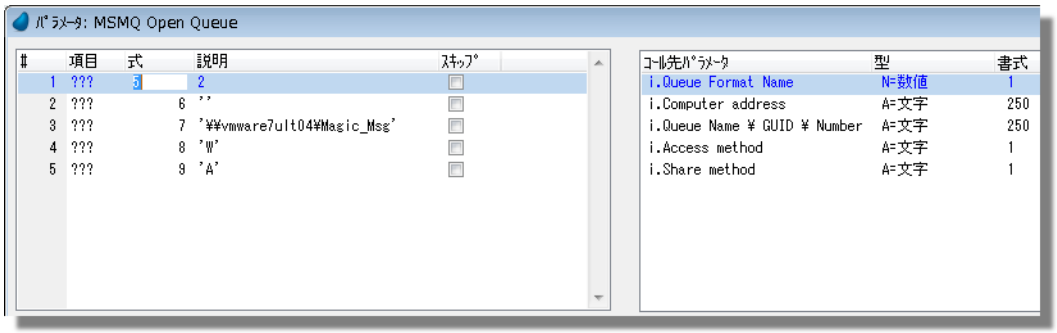


1. データビューエディタで書込リンクコマンドを定義します。
2. **Property** カラムで文字列「Priority」を使用してリンク定義を設定します。**代入**特性にも同じ値を設定します。
3. **Datatype** カラムを、設定するラベルのデータタイプ「N」で更新します。
4. **Data** カラムを、（文字列書式の）プライオリティコードで更新します。必要であれば、**Str()** 関数を使用してコードを文字列に変換することができます。

テーブルをコンポーネントに送る

次に、すべての特性を持つテーブルをコンポーネントに送る必要があります。この方法については、「外部メッセージテーブルをメッセージセットアッププログラム に送るには」（558 ページ）を参照してください。

MSMQ のパブリックキューにアクセスするには



MSMQ のパブリックキューにアクセスしたい場合、以下を使用してキューをオープンする必要があります。

- ***Queue Format Name***：***2***（パブリック）
- ***Computer Address***：**''**（空白）
- ***Queue Name***：キューの GUID

指定された GUID を持つパブリックキューがネットワーク上に存在する場合、送信されたメッセージはそのキューに書き込まれます。

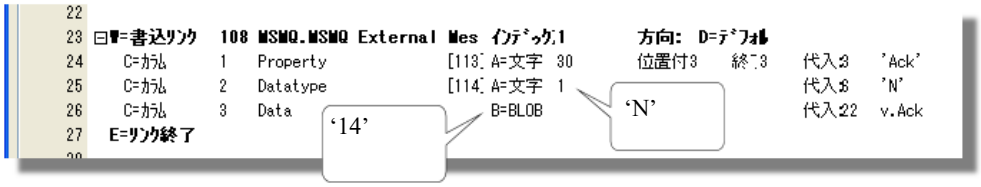
注： パブリックキューにアクセスするには、MSMQ をディレクトリモードでインストールする必要があります。

MSMQ に送られたメッセージが読み込まれたことを確認するには

MSMQ オブジェクトには、メッセージがキューに到達したりするなどの状態を確認することが可能なシステムが含まれています。

MSMQ メッセージは、Microsoft によって定義された一連の特性を持っています。メッセージを送信する際に、Magic xpa の MSMQ コンポーネントはこれらの特性を設定する必要があります。特性は、コンポーネント内に設定する必要があります。MSMQ.External Message Set を使用することでこれらの処理を行うことができます。

MSMQ の Ack 特性を設定する



1. データビューエディタで書込リンクコマンドを定義します。
2. **Property** カラムで文字列「Ack」を使用してリンク定義を設定します。**代入**特性にも同じ値を設定します。
3. **Datatype** カラムを、設定するラベルのデータタイプ「N」で更新します。
4. **Data** カラムを、（文字列書式の）Ack コードで更新します。必要であれば、**Str()** 関数を使用してコードを文字列に変換することができます。

MSMQ の Ack コードには、様々なものがあります。詳細は、『リファレンスヘルプ』を参照してください。この例では、14 の Ack コードを指定しています。これはキュー内のメッセージの受信状態（時間内に受信できた場合は肯定応答、それ以外は否定応答）にもとづいて Ack を送信する指定です。

MSMQ の AdminPath 特性を設定する

Ack メッセージは、MSMQ の管理キューに送られます。管理キューは、MSMQ の **AdminPath** 特性で設定します。設定は、同じように MSMQ.External Message Set を使用して行います。

1. データビューエディタで書込リンクコマンドを定義します。
2. **Property** カラムで文字列「Adminpath」を使用してリンク定義を設定します。代入特性にも同じ値を設定します。
3. **Datatype** カラムを、設定するラベルのデータタイプ「A」で更新します。
4. **Data** カラムを、管理キューのパスを表す文字列で更新します。パスは以下のとおりです。

```
<ServerName>¥<QueueType>$¥<QueueName>
例：serverPC¥Private$¥Magic
```

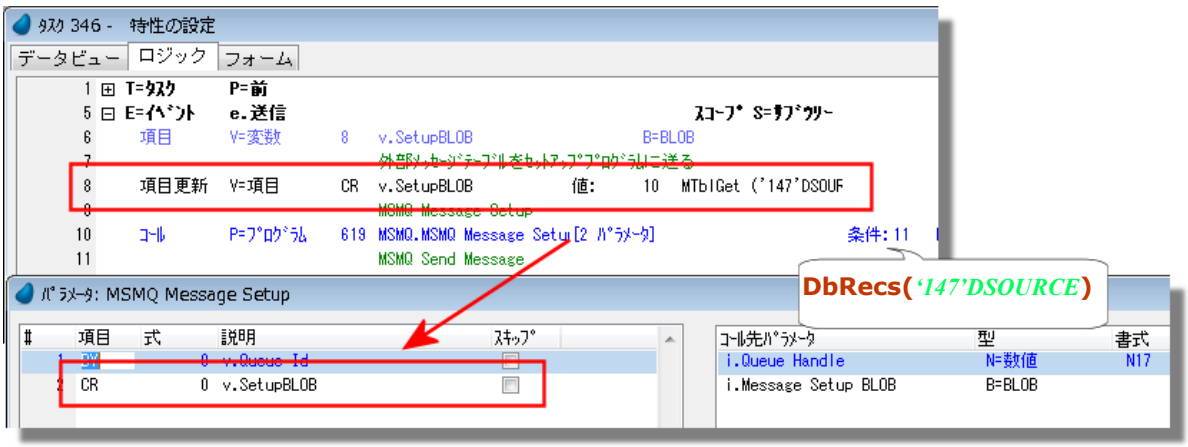
注： 管理キューは、トランザクションキューでなければどのキューも指定可能です。

テーブルをコンポーネントに送る

次に、すべての特性を持つテーブルをコンポーネントに送る必要があります。この方法については、「外部メッセージテーブルをメッセージセットアッププログラム に送るには」（558 ページ）を参照してください。

これで、キュー内のメッセージが受信されると、指定された Ack のタイプにもとづいて MSMQ は応答を返します。アプリケーションは、管理キューを読み込み、必要に応じて応答を返す必要があります。

外部メッセージテーブルをメッセージセットアッププログラム に送るには



MSMQ.External Message Set テーブルに MSMQ の各特性値を設定したら、この内容を有効にするには、このテーブルをコンポーネントに送る必要があります。**MSMQ.External Message Set** テーブルをコンポーネントに送るには、BLOB 項目内にテーブルをパッケージ化し、コンポーネントに送ります。これを行うには、以下の手順を実行します。

1. **MTblGet()** 関数を使用して、BLOB 項目内にテーブルをパッケージ化します。
2. MSMQ.Message Setup プログラムを呼び出します。２番目のパラメータには送信する BLOB 項目を設定します。
3. 常にテーブルを設定するわけではない場合は、セットアップテーブル内にレコードがあるかどうかにもとづいて MSMQ.Message Setup プログラムの呼び出しを調節してください。

これで、メッセージが送信される、**ラベル**特性が設定されます。

メッセージングのエラーを捕捉するには

MSMQ プログラムを使用する場合、エラーが発生するす可能性を考慮する必要があります。エラーが発生すると、各プログラムは、エラー内容を確認するためのエラー番号を戻り値として返します。また、自動的に MSMQ エラーを捕捉するためにイベントロジックユニットを使用することもできます。このロジックユニットを追加するには、以下の手順を実行します。

1. MSMQ エラーを処理したいタスクを開きます。定義する場所は、以下の３つのうちどれかになります。
- メインプログラム。**スコープ**特性は、***G= グローバル***に設定します。
- タスクツリー上の上位タスク。**スコープ**特性は、***S= サブツリー***に設定します。
- MSMQ プログラムを呼び出すタスク。**スコープ**特性は、***T= タスク***に設定します。

2. **ロジック**タブを**クリック**します。
3. **Ctrl+H**（**編集→ヘッダ行作成**）を押下して、新しいヘッダ行を作成します。
4. **E** を入力して***イベント***を選択します。**イベント**ダイアログが表示されます。
5. **イベントタイプ**で ***U= ユーザ***を選択します。
6. **イベント**から**ズーム**して ***MSMQ.Public error*** を選択します。
7. **OK** を**クリック**します。「このイベントに対応するパラメータを作成しますか？」という確認のメッセージボックスが表示されます。**Yes** を**クリック**します。
8. **イベント**ロジックユニットにパラメータ項目が作成されます。
9. **#1** の内容に応じて**スコープ**特性を設定します。
10. 作成されたパラメータ項目は、必要に応じてエラーの処理で使用します。この例では、ユーザ用のエラーメッセージを表示させるために、メッセージテキスト（pO.Error message）を使用しています。

このイベントには、以下の４つのパラメータが作成されます。

パラメータ名	内容
pO.Messaging system	• M …… MSMQ • J …… JMS • W …… WebSphere MQ
pO.Error code	エラーコード
pO.Validation Error?	• True ……エラーが MSMQ コンポーネントの確認機能で発生したも • False ……実行時に発生したエラーの場合
pO.Error message	エラーメッセージ

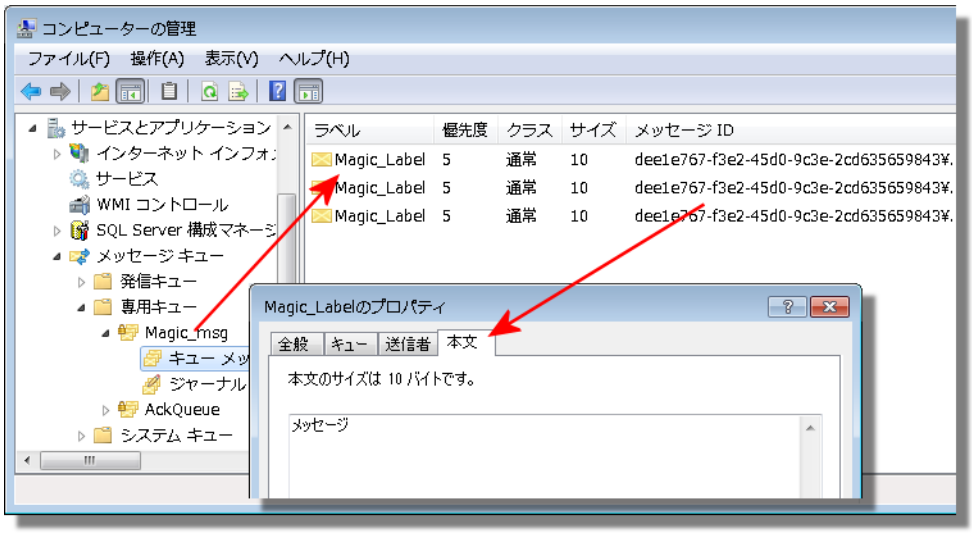
注： ここですべてのエラーメッセージを捕捉した場合、それらはメッセージエラーログにも書き込まれます。このログは、論理名の MessagingComponentDir で指定された場所に作成されます。

MSMQ アプリケーションをデバッグするには

メッセージングは複数のアプリケーションとサーバが関連するため、MSMQ をデバッグすることは Magic xpa のアプリケーションのみの場合より複雑になります。MSMQ のデバッグには以下のようなツールを使用することができます。

- デバッガ（第 28 章 :「デバッガを使用してデバッグするには」（613 ページ）を参照してください）は、Magic xpa の内部処理を確認する場合に非常に有効です。
- MSMQ のエラーは、ログファイルに書き込まれます。ログファイルの位置とファイル名は、論理名 MessagingErrorLogFile で設定されます。デフォルトは、%MessagingComponentDir%MG_message_err.log です。
- Windows 上で MSMQ のメッセージを参照することができます。以下で説明します。

Windows 上でメッセージを確認する



1. コンピュータの管理（スタート→設定→コントロールパネル→管理ツール→コンピュータの管理）ウィンドウを開きます。
2. サービスとアプリケーションのツリーノードを開きます。
3. メッセージングキューを開きます。これで登録されているメッセージキューが表示されます。
4. メッセージを送信したキューに移動します。キューメッセージセクションで、キューから読み込み処理が実行されていないメッセージを参照することができます。
5. 参照したいメッセージにカーソルを置き、右クリックでコンテキストメニューを開き、プロパティを選択します。上図のようにメッセージの内容やの他の特性を参照することができます。

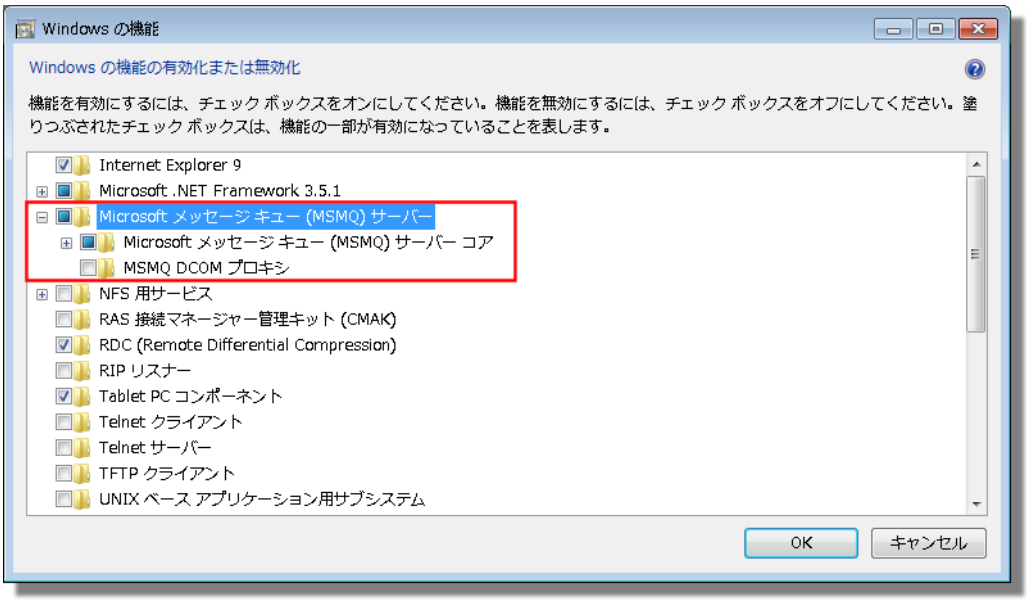
注： コンピュータの管理ウィンドウが開いている状態で操作→最新の情報に更新を選択すると、新しいメッセージを表示させることができます。

PC に MSMQ を設定するには

MSMQ は、Windows に標準で装備されているメッセージングシステムです。ただし、デフォルトではインストールされません。Windows ユーザが MSMQ コンポーネントを選択する必要があります。どのようにインストールするかを説明します。

注： コンポーネントの名前やダイアログのイメージは、OS によって異なる場合があります。

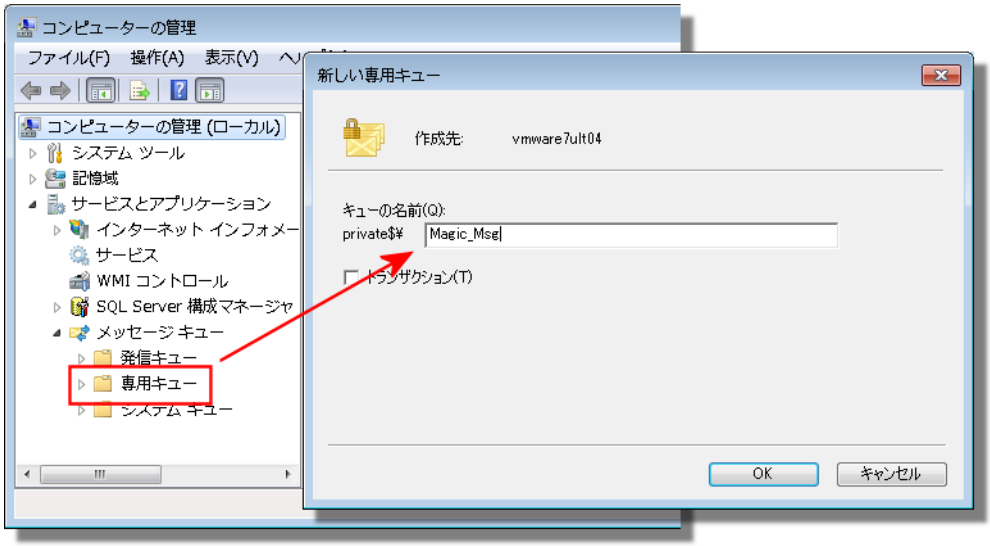
MSMQ をインストールする



MSMQ を利用する前に、あなたの PC にインストールする必要があります。以下の手順で実行します。（注：Windows 7 を前提に説明しています。）

1. Windows のメニューから**スタート→コントロールパネル**を選択します。
2. **プログラムと機能**を**クリック**します。
3. **Windows の機能の有効化または無効化**を**クリック**します。
4. **Microsoft メッセージキュー（MSMQ）サーバー**を選択し、チェックボックスをチェックします。
5. OK を**クリック**すると設定処理が実行されます。

キューを追加する



1. **コンピュータの管理（コントロールパネル→管理ツール→コンピュータの管理）**ウィンドウを開きます。
2. **サービスとアプリケーション→メッセージキュー**を開きます。キューを**パブリック**、または**専用**キューとして追加することができます。パブリックキューはドメインに参加している PC でのみ利用できます。

3. キューを追加するには、コンテキストメニューから**新規→専用キュー**（必要であれば**パブリックキュー**）を選択します。
4. **新しい専用キュー**ダイアログが表示されます。キューに名前を定義します。この例では、$¥Magic_Msg になっています。この名前は、Magic xpa でキューをオープンする際に使用されます。
5. 必要であれば、**トランザクション**のチェックボックスをチェックします。MSMQ のバージョンによっては、トランザクションでないメッセージをトランザクションキューに送信することを許可していないものもあります。
6. OK を**クリック**すると、キューが作成されます。

メッセージングコンポーネントをインストールする

Magic xpa のインストールの際に、カスタムインストールで**メッセージング**コンポーネントを選択するとインストールされます。インストールされていない場合は、インストール処理を実行し直し、**追加 / 削除**オプションで選択します。

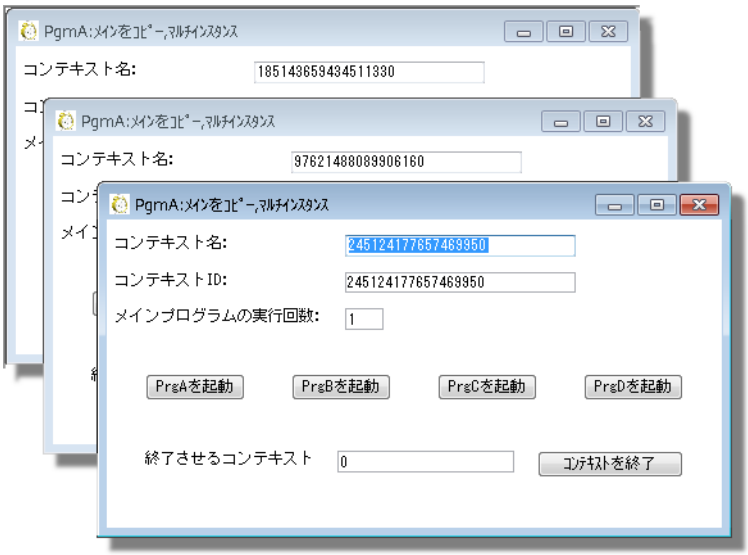
メッセージングコンポーネントを利用する上で、２つの論理名を使用されます。

- MessagingComponentDir ……これはメッセージングコンポーネントが格納されているディレクトリを指定します。コンポーネントは ***.ecf*** ファイルで、メッセージングを利用する上で使用されるプログラムやハンドラが含まれています。***.eci*** ファイルも含まれており、このコンポーネントをアプリケーションに追加することができます。.
- MessagingErrorLogFile ……これはエラーログファイルの位置とファイル名を指定します。

コンポーネントの設定については、第 15 章 :「プロジェクトにコンポーネントを読み込むには」（362 ページ）を参照してください。

第 24 章 : マルチタスク

複数の対話型タスクを同時に実行させるには



Magic プログラムは、デフォルトでは 1 度に 1 つのランタイムツリーを実行させることができます。つまり、メニューからプログラムを開くと、実行中の他のランタイムツリーは自動的に終了します。プログラムは他のタスクを呼び出しますが、このタスクは階層的なツリー構造内で動作します。グローバルイベント使用することで、階層上に存在しないタスクを起動させることができます。これらのタスクはメインタスクと一緒に同時に実行させることができますが、同じプログラムをマルチウィンドウでオープンさせたままにすることはできません。

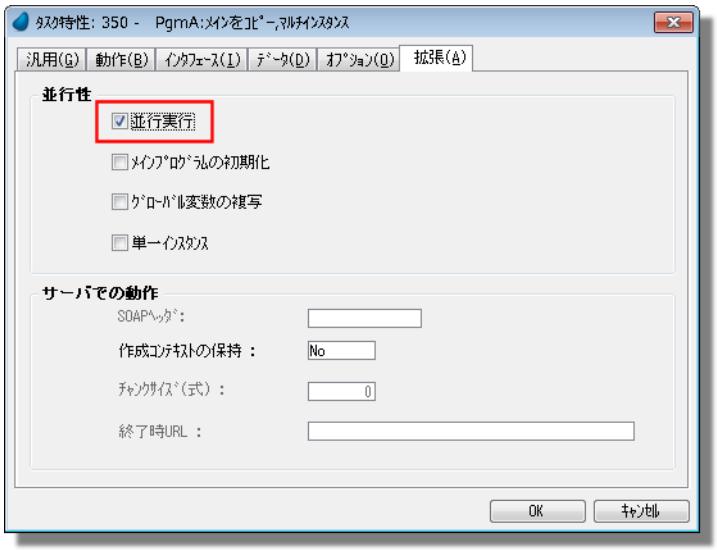
しかし、プログラムを並行モードで動作させることができます。プログラムが上の図のように並行に実行している場合、各プログラムは独自の階層ツリーで実行させたり、より多くの並列プログラムを呼び出すことができます。

各並列プログラムは、ユニークな**コンテキスト ID** を持っています。**コンテキスト ID** は、自動的に割り当てられる文字列で表された 18 桁以内の番号です。必要であれば変更することができます（「現在のコンテキストに異なる名前を設定するには」（570 ページ）を参照してください）。

この例では、Pgm A が 3 回呼び出されています。メニューや別のプログラムから呼び出されたかどうかにかかわらず、プログラムはユニークな**コンテキスト ID** を持っており、他のウィンドウと無関係に動作します。

.

タスクを並行に実行させる



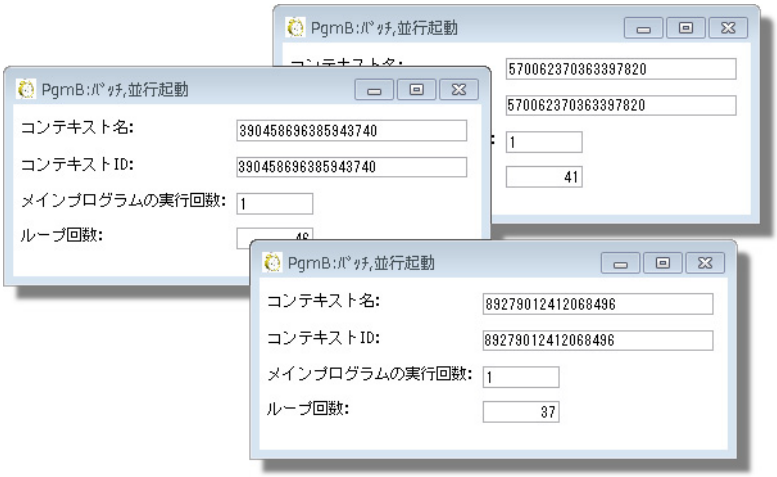
タスクを並行に実行させるには以下の手順を実行します。

1. **タスク特性**（*Ctrl+P*）を開きます。
2. **拡張**タブを**クリック**します。
3. **並行実行**のボックスをチェックします。

これで、プログラムは並行に実行されます。

プログラムが並行に実行するように設定された場合、どのように実行するかを定義するためのオプションを指定することもできます。詳細は、「並行タスクの初期設定をコントロールするには」（575 ページ）を参照してください。

並行処理のバッチプログラムを実行させるには



バッチタスクは、対話型タスクと同じ方法で実行させることができます。並行に実行する各バッチタスクは、独自のコンテキストを取得し、タスクが終了するとクローズします。

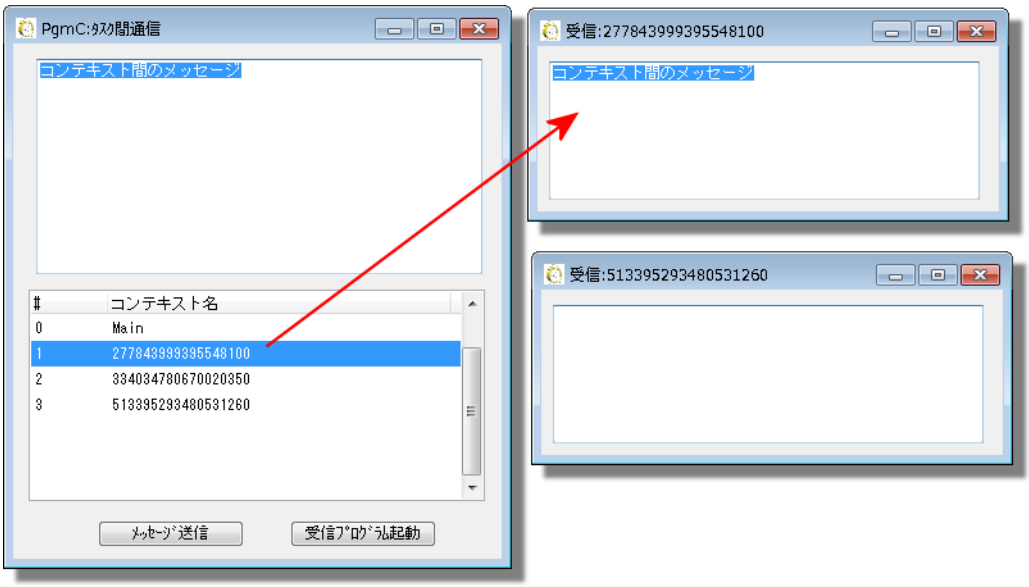
バッチタスクを並行に実行させる

バッチタスクを並行に実行させるには、以下の手順を実行します。

1. **タスク特性**（*Ctrl+P*）を開きます。
2. **拡張**タブを**クリック**します。
3. **並行実行**のボックスをチェックします。

これで、プログラムは並行に実行されます。

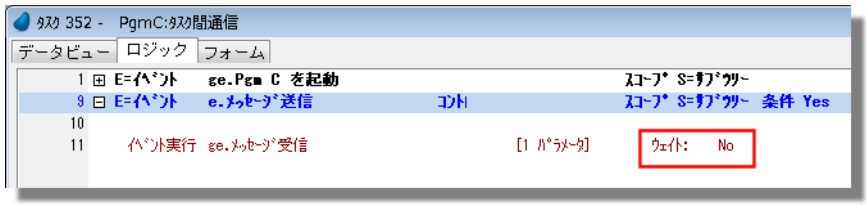
並行実行しているタスク間の通信を管理するには



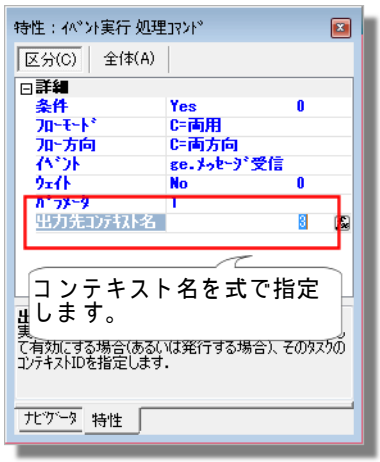
並行タスクは、個別のメモリ空間で実行しているため、それらの間で直接パラメータを渡したり、**メインプログラム**に定義されたグローバル変数を共有することができません。従って、並行タスクを「呼び出す」には、イベントをタスクに送る必要があります。イベントは、並列タスクにデータを渡すためのパラメータを定義することができます。ここでは、その方法について説明します。

必要条件： 通信するプログラムのコンテキスト ID を知る必要があります。コンテキスト ID を取得する方法は、「呼び出された並列プログラムのコンテキスト ID を取得するには」（572 ページ）を参照してください。

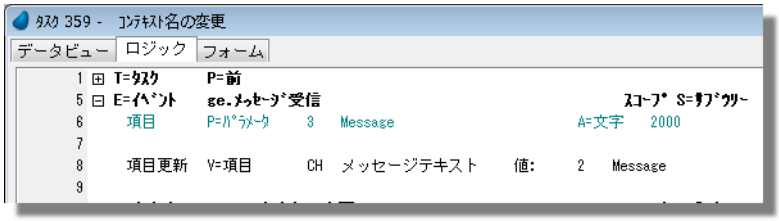
1. プログラム間で渡す必要があるパラメータを持つグローバルイベントを定義します。この例では、**ge. メッセージ受信**と呼ばれるイベントが定義されています。このイベントには、1 つのパラメータ（渡したいメッセージ）が定義されています。**ウェイト**特性は ***No*** に設定する必要があります。
2. 最初のプログラムでは、ここに示されているように、パラメータ渡してイベントを実行します。**ウェイト**特性は ***No*** に設定しなければなりません。No に設定しないとエラーが発生します。



3. **イベント特性**で、**出力先コンテキスト名**特性に、***通信するコンテキスト名***を設定します。ほとんどの場合、1493034580378080 というような Magic xpa で生成された文字型の数値となります。しかし、Main と呼ばれるメインのコンテキストもあるため、必要であればコンテキストに名前を設定することもできます。



これでイベントが発行されると、指定されたコンテキストでイベントが処理されることになります。この例では、メッセージの受信タスクで、**ge. メッセージ受信**イベントが処理されています。

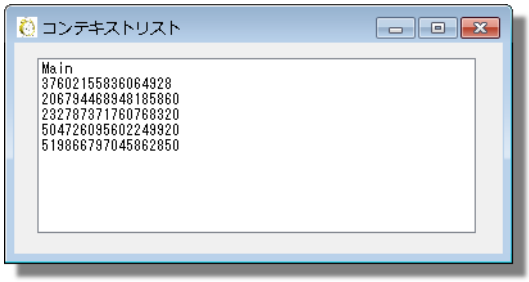


指定されたコンテキストでは、発生したイベントを受け取り、適切に処理を実行します。イベントを受け取ったタスクは、イベントの発生元を認識することはありませんし、その必要もありません。この例では、受け取ったメッセージを表示するだけです。

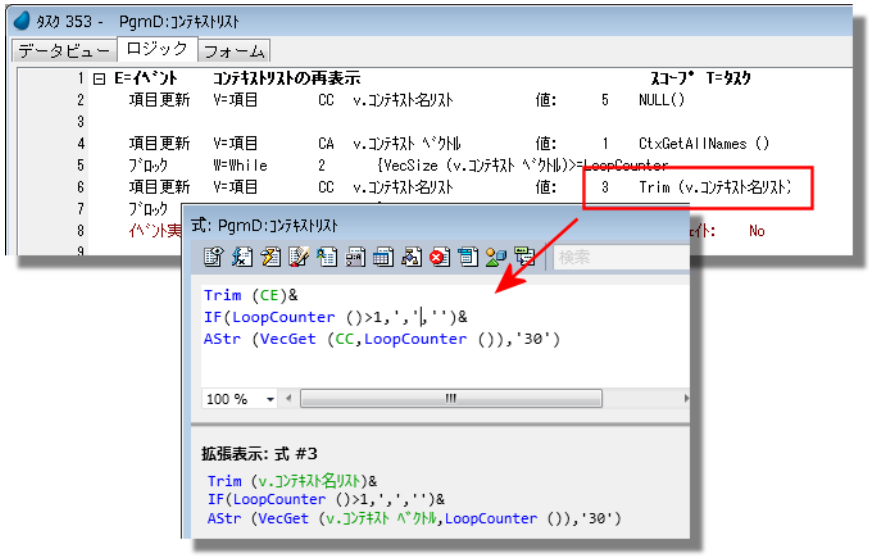
しかし、このタスクが送信元のプログラムにデータを送り返す必要がある場合、送信元プログラムのコンテキスト名を知る必要があります。このような場合は、以下のようにプログラムを修正してください。

1. 送信元のコンテキスト名をパラメータとして追加してイベントを発行します。
2. 受信先では、コンテキスト名を受け取れるようにパラメータ項目を追加します。
3. イベント処理のロジックユニット内で**イベント実行**処理コマンドを定義し、受け取ったコンテキスト名を**出力先コンテキスト**特性に指定します。また、**パラメータ**特性に返したいデータを指定します。
4. 送信元のタスクにこのイベントを受け取る**イベント**ロジックユニットを作成し、パラメータ項目を定義します。

実行中のコンテキストのリストを取得するには



デバッガを有効にしている場合、デバッガの実行コンテキストペインに既存のコンテキストがすべて表示されます。しかし、実行環境で実行中のコンテキストのリストを表示させたい場合は、**CtxGetAllNames()** 関数を利用することで実現できます。この関数は、コンテキスト名が格納されたベクトルデータを返します。Magic xpa のベクトル関数を使用することでこのデータを処理することができます。



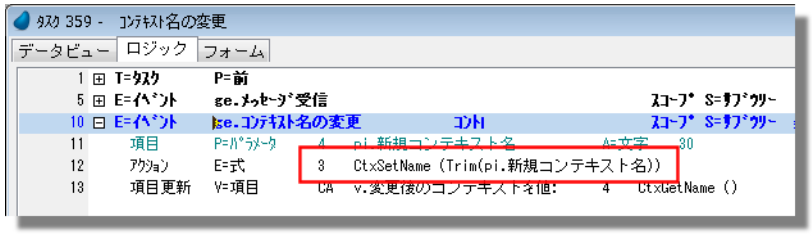
画面にコンテキストを表示させたい場合、ここに示したように、取得したデータをカンマ区切りで結合させることでリスト表示させることができます。

現在のコンテキスト名を取得するには

コンテキスト間で通信する場合、送信元のプログラムを相手先が認識できるようにするため、自身のコンテキスト名を知る必要があります。

コンテキスト名を取得するには **CtxGetName()** 関数を使用します。パラメータはありません。この関数は文字列（最長 128 文字）を返します。Magic xpa によって割り当てられたコンテキスト名は 16 文字の文字列ですが、プログラムによって最高 128 文字までの長さのコンテキスト名を指定することができます。

現在のコンテキストに異なる名前を設定するには



コンテキストに名前を設定する必要がある場合は、**CtxSetName()** 関数を使用します。名前は静的で認識しやすいため、コンテキストに名前を設定することは、単一インスタンスのコンテキストを扱う場合に便利です。

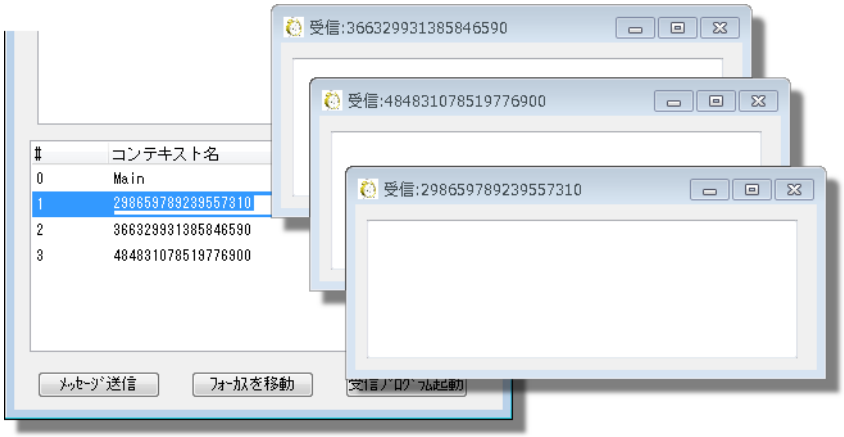
関数の構文は以下の通りです。

CtxSetName(*new name*)

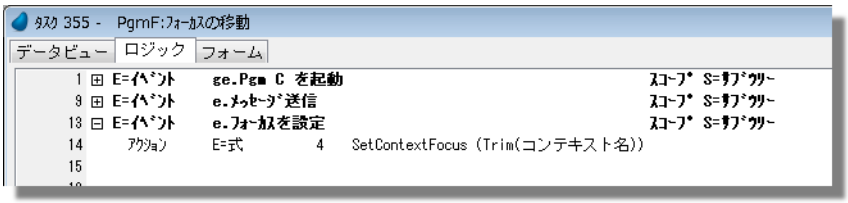
パラメータ：

- *new name*：指定するコンテキスト名です。最高 128 文字の文字列が指定できます。

任意のコンテキストにフォーカスを移すには



ユーザが対話型のコンテキストにアクセスしている場合は、任意のコンテキストをクリックすることができます。しかし、**SetContextFocus()** 関数を使用することでプログラムによって任意のコンテキストにフォーカスを移すことができます。



関数の構文は以下の通りです。

SetContextFocus(*name*)

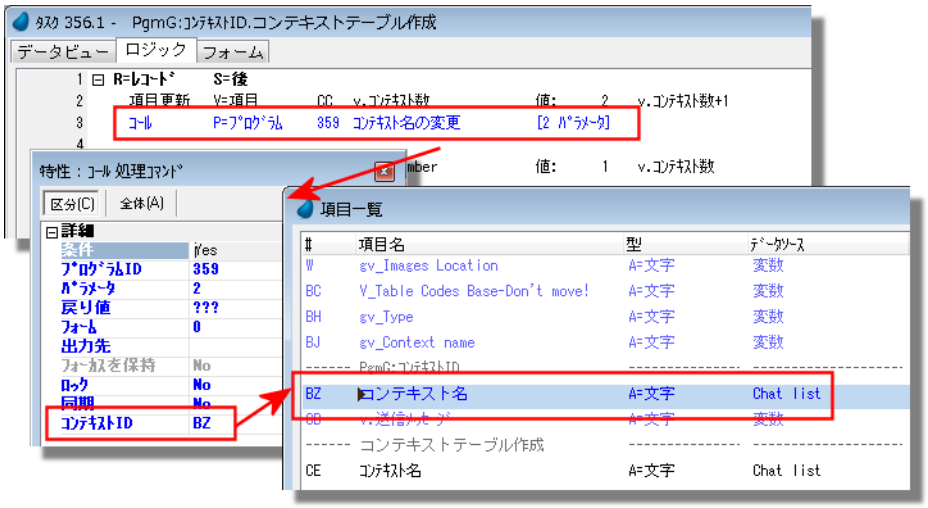
パラメータ：

- ***name***：コンテキスト名を表す文字列です。**Trim()** 関数を使用して余分な空白を削除するようにする必要があります。

関数が実行され、指定されたコンテキストが存在していれば、そのコンテキストにフォーカスが移ります。

コンテキストが存在しており、フォーカスがそのコンテキストに切り替わった場合、関数は True は返します。それ以外は、False が返ります。

呼び出された並列プログラムのコンテキスト ID を取得するには



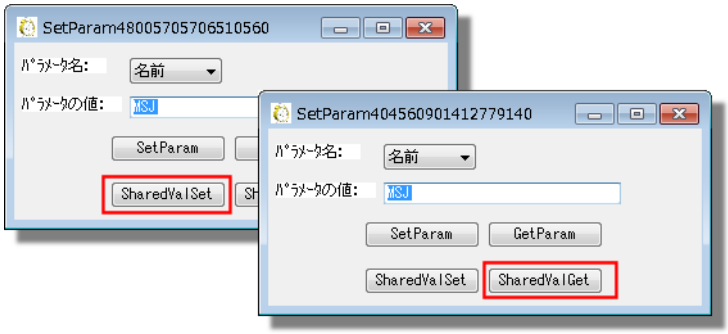
並行プログラムを呼び出す場合、**コール**処理コマンドの**コンテキスト ID** 特性に項目を指定することで簡単に呼び出されたプログラムの**コンテキスト ID** を保存することができます。設定するには以下の手順で実行します。

1. 並行プログラムを呼び出す、**コールプログラム**処理コマンドの定義行に移動します。
2. *Alt+Enter* を押下して**コール特性**を開きます。
3. **コンテキスト ID** 特性に移動します。
4. ここから**ズーム**して、返される**コンテキスト ID** を保持するための文字型項目を選択します。

これで、プログラムが呼び出されると、項目に**コンテキスト ID** の値が格納されます。

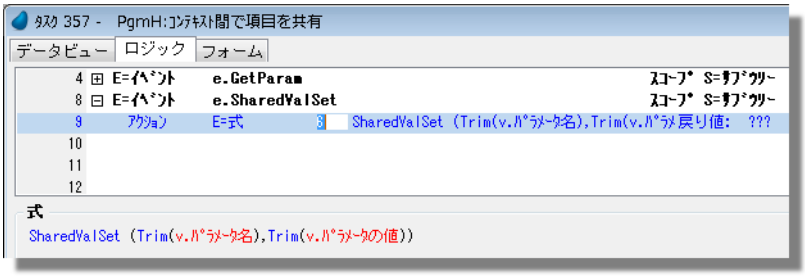
注： 呼び出されたプログラムには、呼び出し元プログラムの**コンテキスト ID** を認識するための機能は持っていません。従って、必要であればパラメータとして自分の**コンテキスト ID** を渡してください。

コンテキスト間で項目を共有するには



単一コンテキスト環境では、通常メインプログラムや **SetParam()** 関数を使用することでプログラム間のデータ共有を実現することができます。しかし、これらはどちらもコンテキスト間では利用できません。上の例では、各ウィンドウが **SetParam()** と **GetParam()** を使用して、自身のコンテキスト内に値を格納することができます。しかし、複数のコンテキスト間で値を共有させるには、**SharedValSet()** と **SharedValGet()** 関数を使用する必要があります。

SharedValSet() 関数を使用する



関数の構文は以下の通りです。

SharedValSet(*name,value*)

パラメータ：

- ***name***：項目名です。上記の例のように項目に格納して使用したり、直接項目名を指定することができます。
- ***value***：項目に格納される値です。データの型は任意ですが、データを取得する際に同じ型になるように開発者の責任で定義する必要があります。例えば、日付データを格納した場合、別のプログラムでは日付型データとして取り出す必要があります。

この関数は、常に True を返します。

SharedValGet() 関数を使用する

関数の構文は以下の通りです。

SharedValGet(*name*)

パラメータ：

- ***name***：項目名です。上記の例のように項目に格納して使用したり、直接項目名を指定することができます。

この関数は、Magic xpa のメモリに格納されている項目の値が返ります。

ヒント：*共有された値は、デバッガの変数リストの最上位に表示されます。*

コンテキスト間でメモリテーブルを共有するには

コンテキスト間で直接メモリテーブルを共有することはできません。しかし、BLOB 項目にメモリテーブルを格納し、**SharedVal()** 関数を使用して BLOB 項目を格納することで共有することが可能になります。

共用メモリにメモリテーブルを格納する例を説明します。

1. 最初に、**MTblGet(*'101'DSOURCE、''*)** を使用して、データソース **#105**（メモリテーブル）を BLOB 項目（b.TableBlob）に格納します。
2. 次に、**SharedValSet(*'ContextTbl'、b.TableBlob*)** を使用して BLOB 項目を共用メモリ（ContextTbl）に格納します。

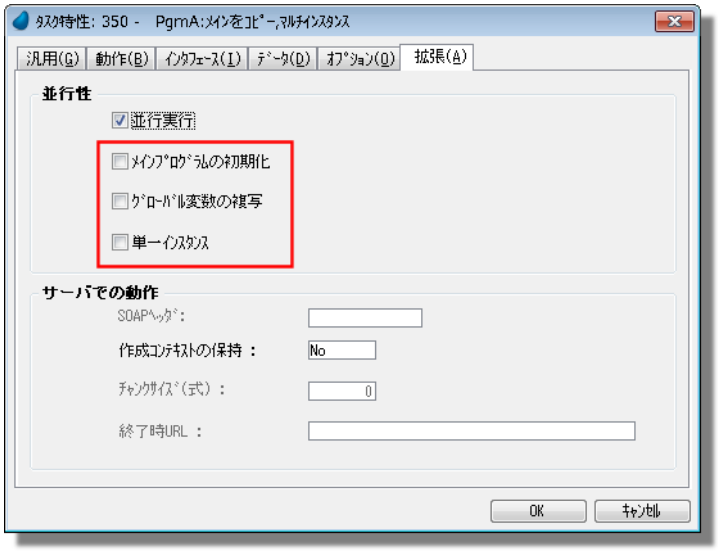
逆の処理は以下のように実行します。

1. 最初に、**SharedValGet(*'ContextTbl'*)** を使用して、共用メモリから BLOB 項目を取得します。
2. 次に、**MTblSet(*b.TableBlob,'105'DSOURCE、'',1*)** を使用して BLOB 項目をデータソース **#105**（メモリテーブル）に展開します。

MTbl() 関数の詳細は、『リファレンスヘルプ』を参照してください。

SharedVal() 関数の使用方法については、「コンテキスト間で項目を共有するには」（573 ページ）を参照してください。

並行タスクの初期設定をコントロールするには



タスク特性で**並行実行**特性をチェックすると、他の３つのオプションが利用可能になります。３つのオプションのうち２つは並行プログラムの初期設定をコントロールするために使用できます。

メインプログラムの初期化

この特性がチェックされた場合、並行プログラムが実行される前に、メインプログラムが再度実行されます。メインプログラムのタスク前が実行され、変数項目の値は新しいコンテキストで初期化されます。

この特性がチェックされない場合、メインプログラム内の項目の値はオリジナルのコンテキストでの内容がコピーされます。その後、新しいコンテキスト内で発生した変更内容はそのコンテキストでのみ有効であり、オリジナルのコンテキストのメインプログラムの内容には影響しません。

グローバル変数の複写

この特性がチェックされた場合、既存のグローバルパラメータのコピーが作成されます。新しいコンテキストはグローバルパラメータのスナップショットを取得します。この場合、グローバルパラメータが更新された場合、新しいコンテキストにコピーされたパラメータが更新対象となります。

この特性がチェックされない場合、新しいコンテキストのグローバルパラメータは初期化されます。

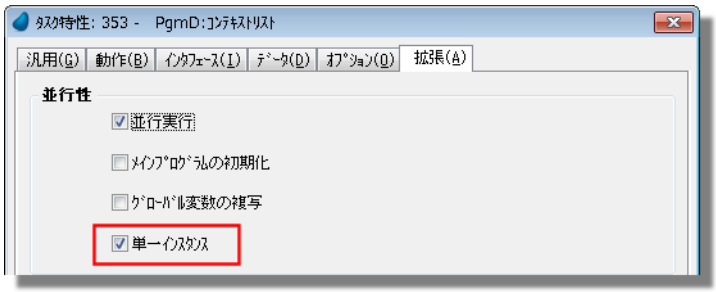
これらのグローバルパラメータは、**SetParam()** 関数で扱うことのできる項目です。**SharedValSet()** と **SharedValGet()** 関数を使用することで、コンテキスト間で常にパラメータを共有することができます。

プログラムを単一インスタンスで実行させるには

並行タスクには、２種類の基本的なタイプがあります。

最初のタイプは、同時に複数の異なるウィンドウをオープンします。例えば、同時に複数の異なるドキュメントを処理するために複数のチャットウィンドウでこれらをオープンするような場合に使用します。これは、Magic xpa の並行プログラムのデフォルトタイプとなります。

２番目のタイプは、常に同じウィンドウで処理する場合です。例えば、ドキュメントを開くメニューやチャットウィンドウを開くリスト等がこのケースに当たります。このタイプの並行プログラムは、**単一インスタンス**プログラムと呼ばれます。



プログラムを単一インスタンスプログラムで動作させたい場合以下の手順で設定します。

1. **タスク特性**（*Ctrl+P*）を開きます。
2. **拡張**タブを選択します。
3. **単一インスタンス**特性をチェックします。

これで、メニューからや、別のプログラムから起動された場合、このプログラムは独自のコンテキストを開きます。しかし、このプログラムがすでに実行中に、同じように起動された場合、新しいコンテキストを作成せず、既存のコンテキストが切り替わります。

単一インスタンスプログラムの再起動を識別するには

インスタンスプログラムが最初に起動されると、新しいコンテキストを作成します。その際、タスク前が実行されます。

単一インスタンスプログラムが２回目に起動されると、既存のインスタンスが使用されます。この場合、フォーカスは移動しますが、タスク前は実行されません。

従って、このプログラムが再度呼び出されたことを認識させるための処理が必要になります。

このような場合、プログラム再呼出という内部イベントが発生します。このイベントは、単一インスタンスプログラムが、再起動された場合のみ実行されます。最初にプログラムが起動された場合は、タスク前を利用します。

［このページは意図的に空白にしています。］

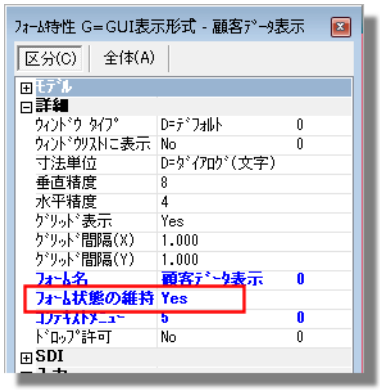
第 25 章 : ウィンドウインタフェース

エンドユーザがカスタマイズしたフォーム状態を保持するには

Windows 上で実行する Magic xpa は、ウィンドウ表示の内容を様々にカスタマイズすることができます。例えば、テーブル上のカラムのサイズを変更したり、カラムの配置を変更したりすることができます。ウィンドウの位置やサイズの変更も可能です。

ユーザは、セッション間に保存されたこれらのカスタマイズ情報を保持することを考えると思います。しかし、これを手動で実現するには、それはたくさんのプログラミング作業が必要になります。Magic xpa には、このような情報を保持するための機能が備わっています。この設定方法について説明します。

フォーム状態の維持特性を使用する

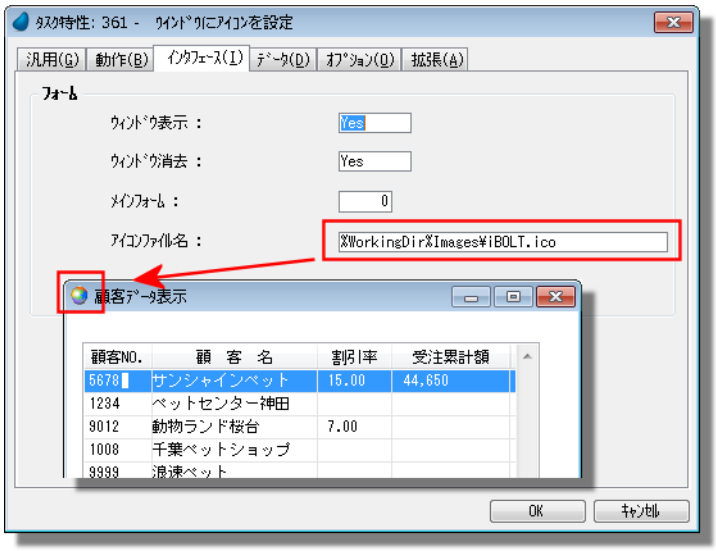


1. タスクの**フォームエディタ**を開きます。
2. ***Alt+Enter*** を押下して**フォーム特性**を開きます。
3. **フォーム状態の維持**特性に移動します。
4. ***Yes***（デフォルト値）を設定します。

これで、ユーザがウィンドウに入り直す際に、ユーザによってカスタマイズされたウィンドウ情報が記憶されます。

注 : この機能は、開発エンジンで実行している場合は無効です。つまり、**フォーム状態の維持**特性を設定し、***F7*** によってプログラムを実行するしても、フォームのカスタマイズ情報は保持されません。

ウィンドウにアイコンを設定するには



アプリケーション特性の**アイコンファイル名**特性にて、アプリケーション全体に反映されるデフォルトアイコンファイルを指定することができます。しかし、ウィンドウ毎に個別のアイコンを設定することもできます。

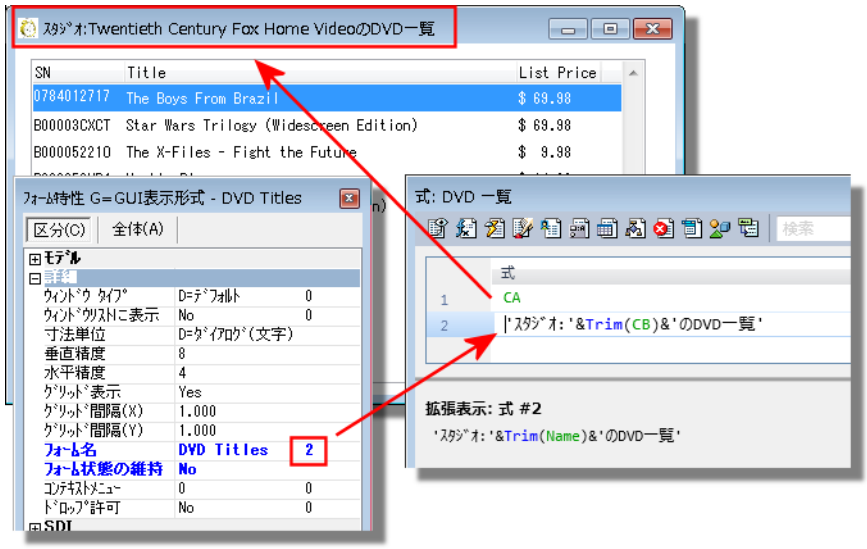
1. アイコンを設定したいタスクを開きます。
2. ***Ctrl+P*** を押下して**タスク特性**ダイアログを開きます。
3. **インタフェース**タブを**クリック**します。
4. **アイコンファイル名**特性で**ズーム**してアイコンファイルを選択するか、相対パスを入力するか、論理名を使用してアイコンファイル名を指定します。

アイコンは実行時にウィンドウの左上に表示されます。

ウィンドウタイトルを動的に設定するには

デフォルトでは、**フォーム名**特性はタスクの名前から引き継がれます。**フォーム特性**の**フォーム名**特性を変更することで簡単にこれを変更することができます。しかし、実行時に動的に名前を設定することもできます。

フォーム名特性を式で設定する



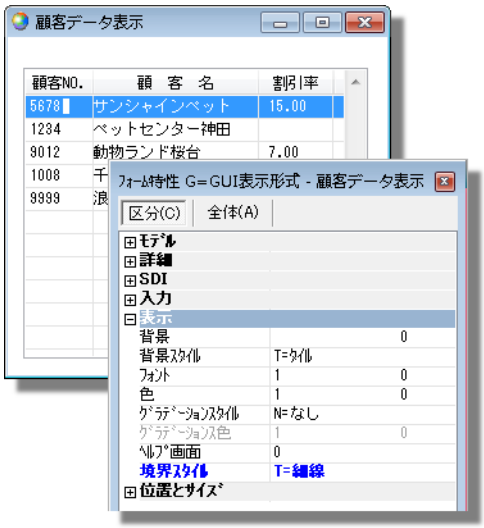
1. タスクの**フォームエディタ**を開きます。
2. *Alt+Enter* を押下して**フォーム特性**を開きます。
3. **フォーム名**特性に移動します。
4. **式**欄からズームしてフォームタイトルとして評価される文字型データを定義します。

開発時は、**フォーム名**特性のテキスト指定がタイトルとして表示されます。しかし、実行時は、式が評価されタイトルバーに表示されます。

ユーザがウィンドウのサイズ変更をできないようにするには

デフォルトでは、ユーザがウィンドウの端をドラッグすることでウィンドウのサイズを変更するこができます。しかし、必要であれば、**境界のスタイル**特性を ***T= 細線***または ***N= なし***に設定することで、この操作ができないようにすることが可能です。

サイズ変更を防止する



1. タスクの**フォームエディタ**を開きます。
2. ***Alt+Enter*** を押下して**フォーム特性**を開きます。
3. **境界のスタイル**特性に移動します。
4. **式**欄から**ズーム**してフォームタイトルとして評価される文字型データを定義します。
5. ***T= 細線***または ***N= なし***を選択します。

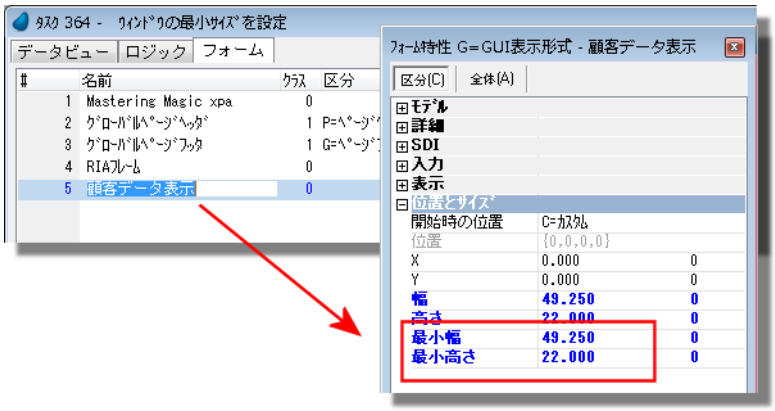
これで、カーソルの形状はウィンドウの端に移動しても変更されず、ウィンドウのサイズを変更することができなくなります。

ウィンドウの最小サイズを設定するには

Magic xpa で作成されるウィンドウは、デフォルトではユーザによってサイズが変更可能になっています。**位置**特性の指定にもとづいて、ウィンドウのサイズが変更されると、テーブルなどのコントロールもサイズが変更されます。しかし、フォームのサイズを小さくし過ぎると非常に見づらい表示になる場合があります。

これを防止するには、フォームの最小サイズを設定することです。フォームは、開発時に設定されたサイズ以上に大きくすることはできますが、指定されたサイズより小さくすることはできなくなります。

最小サイズを設定する

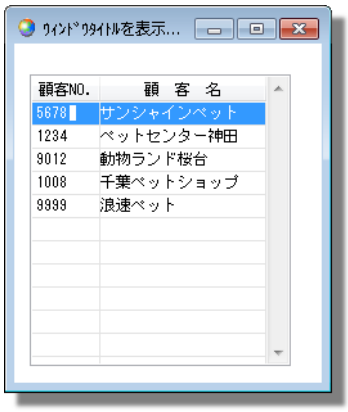
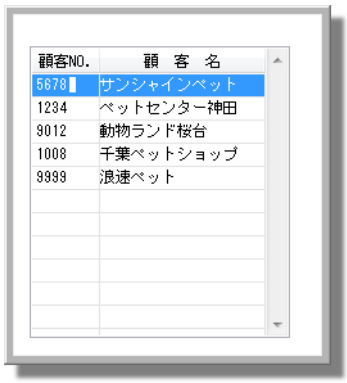


1. タスクの**フォームエディタ**を開きます。
2. *Alt+Enter* を押下して**フォーム特性**を開きます。
3. **最小幅**特性と**最小高さ**特性の値を設定します。
4. これらの特性値を式で指定することもできます。

これで、この例のように、フォームは幅 49.25（ダイアログ）、高さ 22（ダイアログ）以下には小さくすることはできません。

ヒント : *最小サイズを設定する場合の適切な値を求める方法として、手動でウィンドウのサイズを変更し、その時の幅と高さの特性値をもとに設定します。*

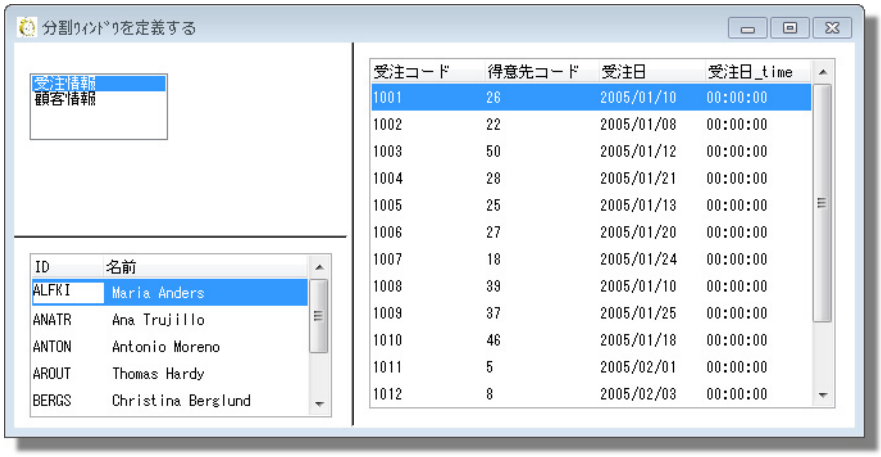
ウィンドウタイトルを表示させないようにするには

タイトルを表示	タイトルを表示しない
	
フォーム特性→タイトルバー→ Yes	フォーム特性→タイトルバー→ No

デフォルトでは、すべてのウィンドウは標準の Windows タイトルバーを持っています。**フォーム特性**の**タイトルバー**特性を ***No*** に設定することで、タイトルを表示させないようにすることができます。

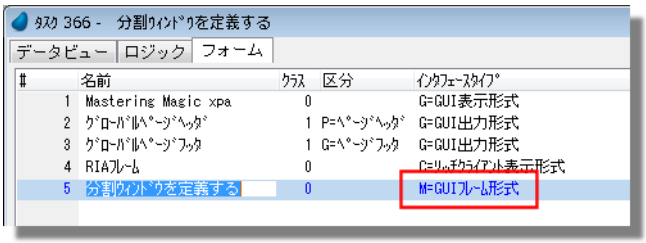
タイトルバーは表示させ、タイトルのみ表示させないようにするには、**フォーム名**特性を空白にします。

タスクに分割ウィンドウを定義するには

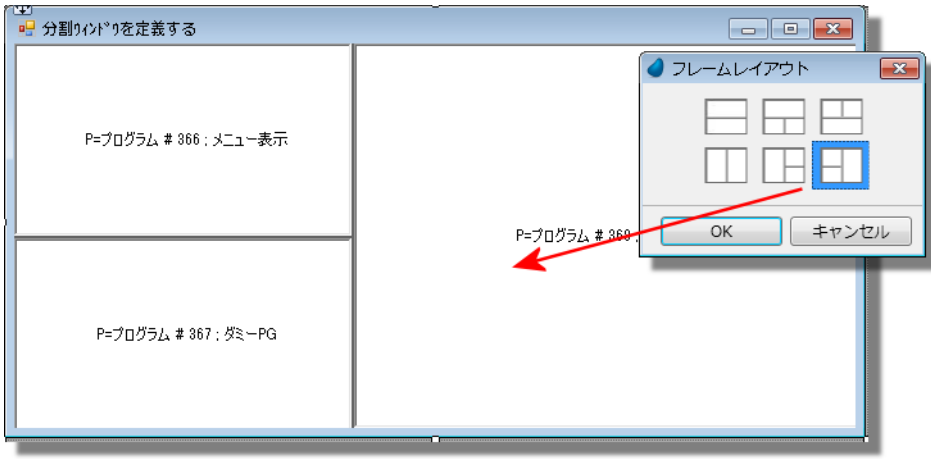


注： **GUI 表示フォームの分割特性は、Ver2 よりサポートされなくなりました。**

分割フォームを定義するには、フレームを使用する必要があります。異なるプログラム（またはサブタスク）が各フレーム上で動作します。そして、これらは実行時に異なるプログラム（またはタスク）に切り替えることができます。この例では、ユーザの選択内容にもとづいて、注文書またはユーザ一覧が左下のフレームに表示されます。



1. まず、フォームの［インターフェースタイプ］で［GUI フレーム形式］または［リッチクライアントフレーム形式］を定義します。次に、ズームしてフォームを開きます。



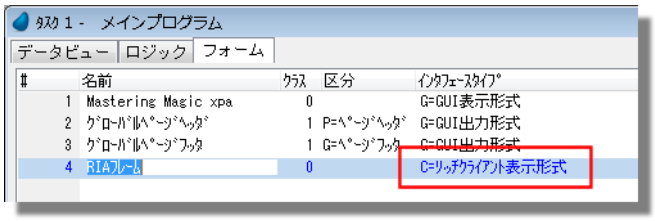
2. フォームを開き、パレットから設定したフレームパターンを選んでください。

3. 現在のタスクからプログラムやサブタスク、またはフォームを呼び出すために、各フレームの特性を設定します。すぐにプログラム（またはタスク）を開くようにしない場合は、ダミーのプログラム（何もしない空のプログラム）を使用してください。
4. リッチクライアントタスクでは、ロジックユニット内でプログラムを呼び出す際に［出力先］特性にフレーム名を指定することで表示させるプログラムを変更させることができます。（「サブフォームの内容を動的に切り替えるには」を参照）

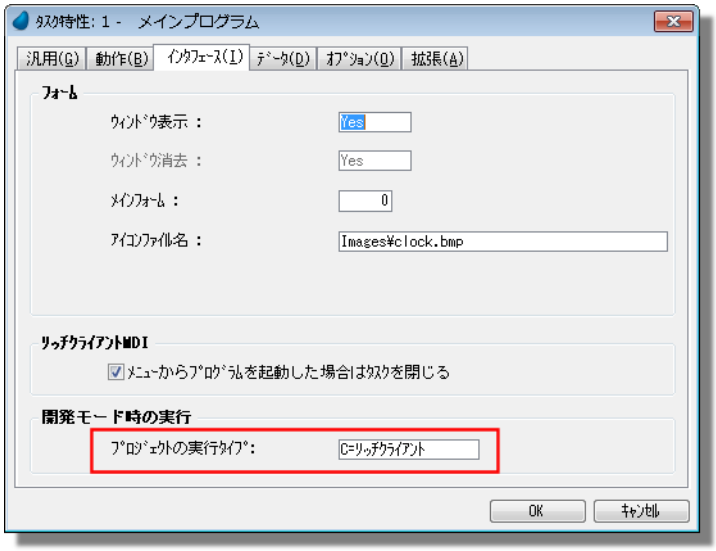
これで、プログラムを実行すると、タスクは定義された領域で動作します。

MDI コンテナを定義するには

Windows 環境で動作している大部分のプログラムは、MDI インタフェースを使用しています。この場合、上辺にプルダウンメニューが表示され、底辺にステータスバーが表示される 1 つの領域内でプログラムが実行されます。Ver2.x では、メインプログラムに表示フォームを定義することで、意図的に MDI 環境を使用するようになりました。ここでは、この方法について説明します。



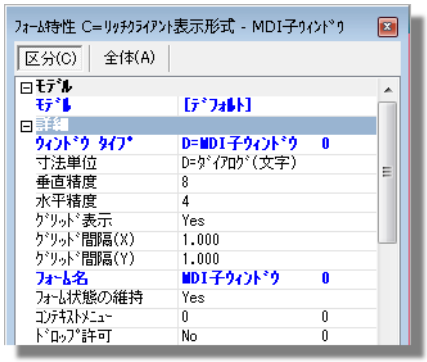
1. 最初に、**メインプログラム**でフォームを追加します。**インターフェースタイプ**として、***リッチクライアント表示形式***（または、***GUI 表示形式***）を選択します。選択された名前は、MDI フレームの最上位に表示されます。
2. **フォーム特性**で、リッチクライアントアプリケーション（または、オンラインアプリケーション）で使用したいプルダウンメニューを定義します。メニューから起動されるプログラムは、すべて同じ実行タイプのプログラムでなければなりません。MDI の背景色や、イメージまたはグラデーションスタイルを定義することもできます。



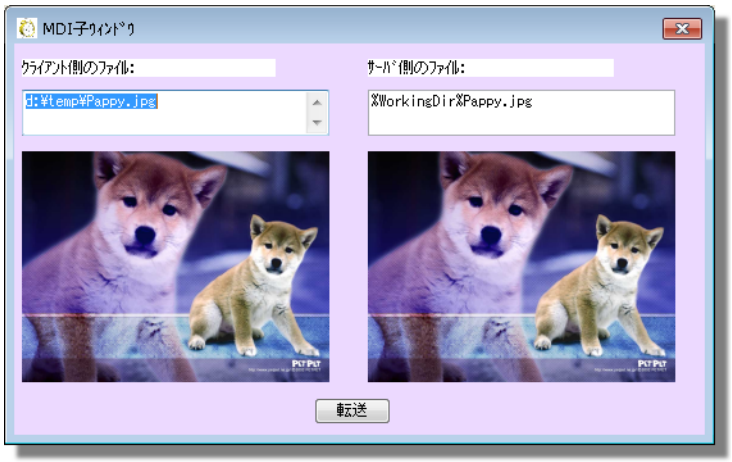
3. Magic xpa Studio で動作を確認する場合は、**メインプログラム**から**タスク特性**の**インタフェース**タブを開き、**プロジェクトの実行タイプ**特性を***リッチクライアント***（または、***オンライン***）に設定します。実行エンジンで動作させる場合は、必要ありません。

注： この設定によって、Magic Studio 環境での開く MDI フォームが決まります。オンラインタスクとリッチクライアントタスクを一つのプロジェクト内に混在させている場合は注意してください。

4. タスクを実行する場合の動作を定義します。Windows の MDI プログラムは、ネイティブな SDI プログラムと異なり、通常一度に 1 つのタスクのみ実行します。**メニューからプログラムを起動した場合はタスクを閉じる**をチェックすると、これと同じように動作します。



5. プログラムを MDI の境界内で動作するように設定したい場合は、**ウィンドウタイプ**特性を ***MDI 子ウィンドウ***または ***MDI 調整***に設定してください。



これで、MDI 環境で動作します。*F7* を押下してプログラムをテストしたり、*Ctrl+F7* を押下してアプリケーションをテストする場合、Magic xpa Studio 上でも同じように動作します。

注： メインプログラムで MDI 環境を有効にしている場合、リッチクライアントアプリケーションを公開する際、開始プログラム名を指定する必要はありません。

フォームをスケーリングするには

Magic xpa は、実行中のタスクのフォームをスケーリングするために 3 つの方法を提供しています。

- ［スケーリング許可］プロパティの使用。実行時にフォームをスケーリングさせたい場合は、「True」に設定します。
- SpecialFormScalingByDefault という特殊フラグの使用。MAGIC.INI でこのフラグを Y に設定します。これにより、プロジェクト内のすべての GUI 表示フォームの［スケーリング許可］プロパティの値がデフォルトで True として設定されます。
- ステータスバーのコンボボックスを使用。コンボボックスは、[スケーリングの許可] プロパティの値に応じて有効または無効になります。100%、110%、120%、130%、140%、150% または 200%のサイズでフォームを拡大 / 縮小することができます。

注： SDI および MDI 子フォームのみがスケーラブルです。

注： 実行時にフォームをスケーリングできます。

注： ［外部テストツール使用］が「Yes」の場合、スケーリングはサポートされません。

注： 位置プロパティは、［スケーリングを許可］が「True」の場合、［テーブル］コントロールの子コントロールでのみサポートされます。

注： RIA およびフレームのスケーリングはサポートされていません。

注： スケーリングが有効になっている場合、位置はスケーリングではサポートされないため、サブフォームの［自動調整］プロパティの値が「コントロールに依存」の値が「なし」として機能します。

サポートバージョン：4.6

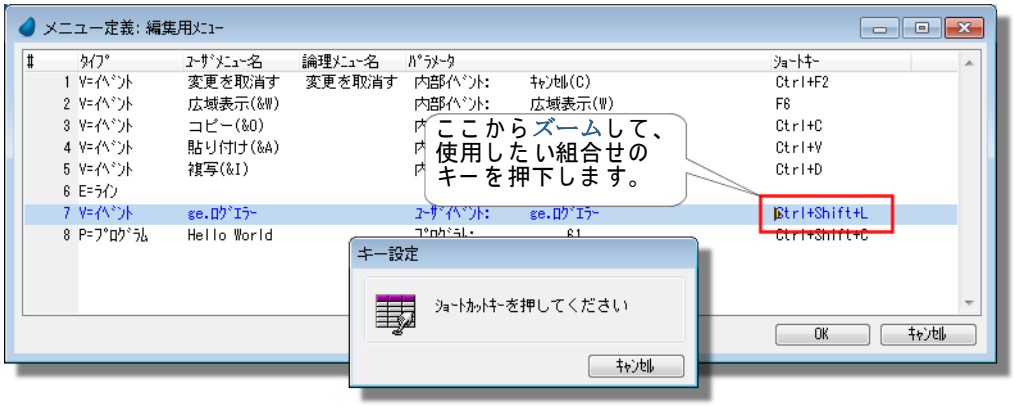
［このページは意図的に空白にしています。］

第 26 章 : メニュー

メニューにショートカットキーを定義するには

多くのユーザは、頻繁に使用する機能に対して*ショートカットキー*（または、*ホットキー*）と呼ばれる組み合わせキーを割り当てることがあります。アプリケーションを作成する際に、メニューに対して独自のショートカットキーを必要に応じて定義することができます。

内部イベント以外に対してショートカットキーを割り当てる



1. **ショートキー**カラム上にカーソルを置きます。
2. ここらか**ズーム**します（*F5*、または**ダブルクリック**）。**キー定義**ウィンドウが表示されます。
3. 割り当てたい組み合わせキーを押下します。この例では、*Ctrl+Shift+L* を押下しています。*Alt* や *Enter* を含めた、任意のキーの組み合わせを使用することができます。
4. 組み合わせキーを押下すると、**キー設定**ウィンドウがクローズされ、**ショートキー**カラム上に設定した組み合わせキーが表示されます。

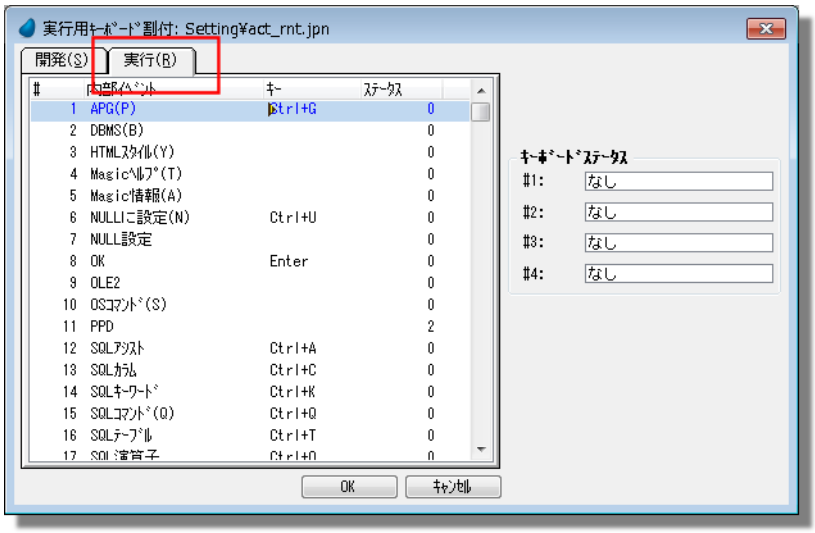
これで、アプリケーションが実行されると、定義されたショートカットキーを押下することで、メニューに定義された処理が実行されます。

内部イベントに対してショートカットキーを割り当てる

内部イベントに対して**ショートカットキー**を割り当てようとすると、以下のメッセージが表示されます。

内部イベントのショートカットキーは、キーボード割付（**オプション→設定→キーボード割付**）にもとづいて、自動的に設定されます。

キーボード割り当てを変更する場合は、以下に示されるように、実行タブ上の内容のみ変更しなければなりません。
キーボード割り当てを変更すると、変更内容はメニュー表示に自動的に反映されます。



実行中のプログラムのメニューを変更するには

メニューリポジトリを使用することで、Magic アプリケーションのメニュー構造を設計することができます。しかし、実行しているプログラムや処理内容にもとづいてメニュー内容を変更することもできます。

メニュー表示の変更

メニュー上の項目の有効 / 無効を指定することができます。以下の関数は、メニュー上の項目自体を変更することはありませんが、メニュー項目を非表示にしたり、無効にしたりすることができます。

- **MnuCheck()** ：メニュー項目にチェックマークを付けたり外したりすることができます。
- **MnuEnable()** ：メニュー項目の有効／無効を指定できます。無効にした場合、メニュー名は灰色で表示されます。
- **MnuShow()** ：メニュー項目の表示／非表示を指定できます。

これらは、メニュー項目に依存します。詳細は、「実行時にメニューの表示／非表示を行うには」（598 ページ）を参照してください。

サブメニューの追加

メニュー関数は、メニュー項目を追加したり、削除したりすることを可能にします。これらの関数は、メニュー項目の変更は行いません。メニュー項目を既存のメニュー構造に追加して、メニュー表示を動的に変更するために使用されます。この関数を使用することで、メニュー表示に柔軟性を与えることができます。

- **MnuAdd()** ：メニュー項目を追加します。
- **MnuRemove()** ：メニュー項目を削除します。
- **MnuReset()** ：メニューをデフォルト状態に戻します。

注：　セキュリティ上の理由でメニューを変更するのであれば、そのメニュー項目に対して権利設定を行うことで対応できます。ユーザが、メニュー項目を使用する権限を持っていない場合、プログラミングで操作することなく自動的に表示されなくなります。

MnuAdd() 関数を使用する

MnuAdd() 関数は、メニュー項目をメニューシステムのどの中にでも追加することができます。構文は以下の通りです。

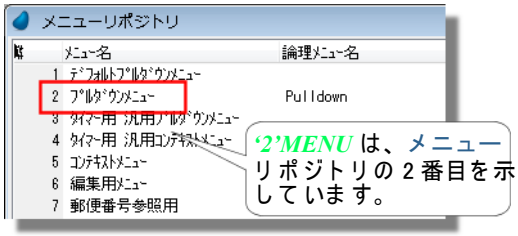
MnuAdd(*MenuNumber*, *MenuPath*)

パラメータ：

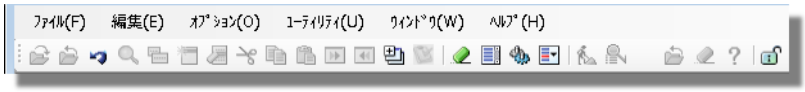
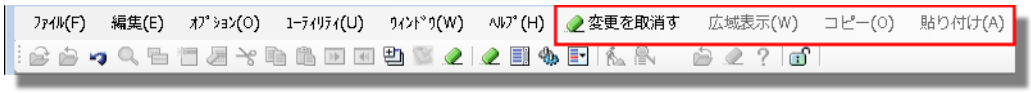
- ***MenuNumber*** ：追加対象となるメニューの番号。ここには、メニューリポジトリの行番号を指定します。MENU リテラルを指定する必要があります。この例では、***'2'MENU*** を使用しています。
- ***MenuPath*** ：この後に挿入したいメニュー項目のパス。パスの最後にバックスラッシュ（¥）が指定されている場合は、サブメニューとして追加されます。

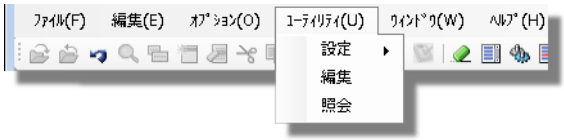
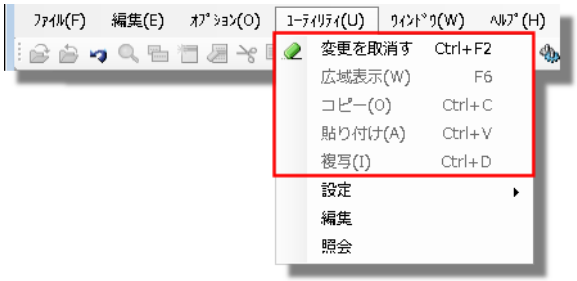
以下は設定例と、それによる表示結果を示しています。

メニューリポジトリは、以下のように定義されています。メニュー **#2** は、プルダウンメニューに追加しようとしているものです。



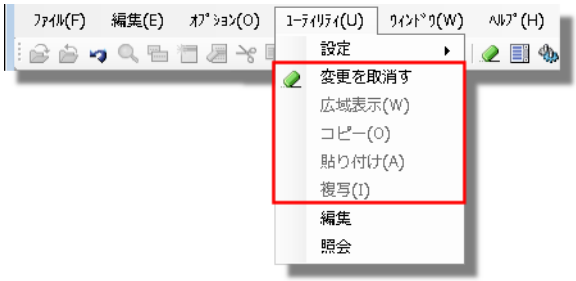
以下は、**MnuAdd()** の 3 つの異なる指定方法によってどのように表示が変わるかを示しています。

関数の指定例
変更前のメニュー

MnuAdd(*'6'MENU,''*) メニュー **#6**（編集用メニュー）をデフォルトプルダウンメニューの最後の位置に追加されます。

MnuAdd(*'6'MENU,'UtilityMenu'*) デフォルトプルダウンメニューのユーティリティメニューの後にメニュー **#6**（編集用メニュー）が追加されます。

変更前のメニュー

MnuAdd(*'6'MENU,'UtilityMenu¥'*) ユーティリティメニューのサブメニューとしてメニュー **#6**（編集用メニュー）が追加されます。

MnuAdd(*'6'MENU,'UtilityMenu¥SetupMenu'*) ユーティリティメニューのサブメニューである**設定**メニューの後にメニュー **#6**（編集用メニュー）が追加されます。

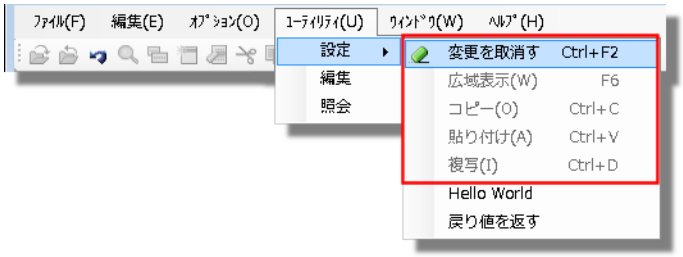
メニュー

関数の指定例



MnuAdd(*'6'MENU,'UtilityMenu¥SetupMenu¥'*)

ユーティリティメニューのサブメニューである**設定**メニューのサブメニューとしてメニュー **#6**（編集用メニュー）が追加されます。



MnuRemove() 関数を使用する

MnuRemove(*MenuNumber*, *MenuPath*)

パラメータ：

- *MenuNumber* …… 削除対象となるメニューの番号。ここには、メニューリポジトリの行番号を指定します。**MENU** リテラルを指定する必要があります。
- *MenuPath* …… 削除するメニュー項目のパス。**MenuAdd** () 関数で指定したパスと同じ値を指定する必要があります。

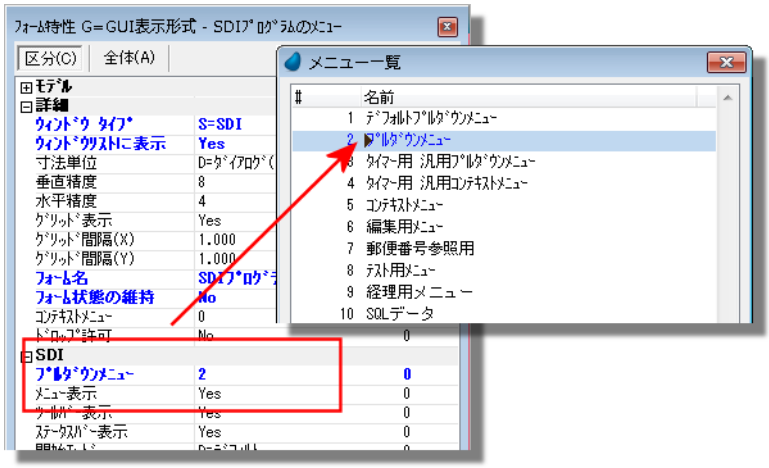
MnuRemove() 関数は、**MnuAdd()** 関数と同じようなパラメータですが、動作は逆になり、追加されたメニューを削除します。

MnuAdd(*'3'MENU,'UtilityMenu¥SetupMenu'*) や **MnuAdd(*'3'MENU*)** は、両方とも前述で追加されたメニューを削除します。

MnuReset() 関数を使用する

関数は、**MnuAdd()** 関数が実行される前のメニュー状態に戻します。**MnuRemove()** の代わりに **MnuReset()** を使用することもできますが、**MnuRemove()** が、追加された 1 部分のみを削除できるに対し、**MnuReset()** は追加されたメニューがすべて削除されます。

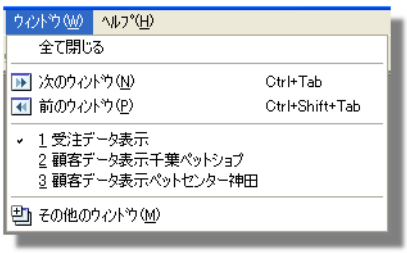
SDI プログラムのメニューを変更する



SDI プログラムは、独自のプルダウンメニューを持っています。SDI フォーム上の**プルダウンンメニュー**特性を指定することで指定することができます。式で指定することもできますが、式はプログラムが起動される前に 1 回だけ評価されるため、プログラムの実行中に式の評価結果が変更されてもメニューには影響しません。プログラムの実行中に変更したい場合は、メニュー関数を使用することで変更することができます。

キーボードでウィンドウの切り替えを有効にするには

Magic xpa には、ユーザに対して複数のウィンドウを一度にオープンしておくことを許可させるためのオプションがあります。ユーザがこれらのウィンドウ間の移動を容易にするために、ウィンドウメニュー機能を使用することができます。

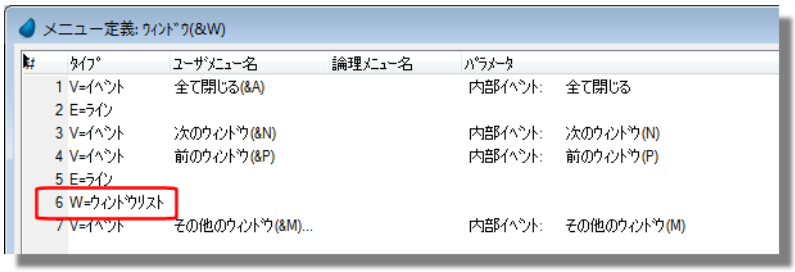


ウィンドウメニュー機能を使用して、複数のウィンドウ間を移動することができます。この例では、同時に開いている 3 つのウィンドウがあります。 1 つのウィンドウは、注文データを表示しています。他の 2 つのウィンドウは、2 つの異なる顧客用に起動された同じ顧客情報表示のプログラムです。

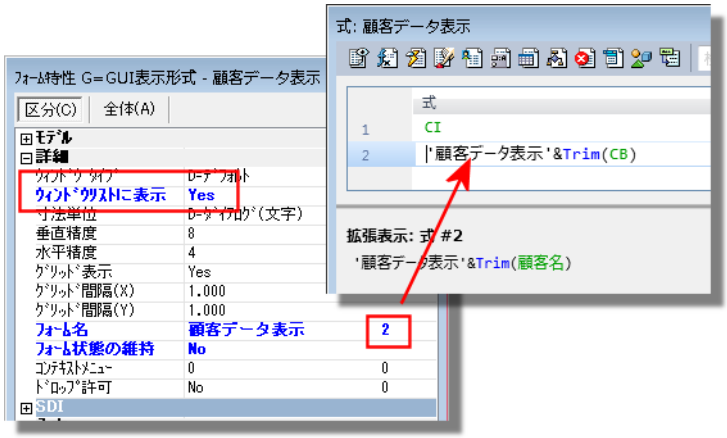
- **ウィンドウ操作**：ユーザは、オープンされたウィンドウ間で移動するための**次のウィンドウ**（*Ctrl+Tab*）または**前のウィンドウ**（*Ctrl+Shift+Tab*）を使用することができます。また、メニュー上にあるウィンドウのリストからウィンドウを選択することができます。メニュー項目の最初の文字を入力することでカーソルをその項目に移動します。
- **リストの順番**：使用された順番や作成された順番にもとづいて、最新のウィンドウのリストを表示します。表示基準は、**アプリケーション特性**の**ウィンドウリストの表示順序**特性によってアプリケーション単位で指定することができます。この設定は、メニューリストやウィンドウ間で移動する場合に使用する *Ctrl+Tab* の両方に影響します。

次に、この機能の使用方法について説明します。

ウィンドウメニューを使用する



1. 最初に、必要な場所にウィンドウメニューを定義する必要があります。デフォルトでは、上記で示されるように、**ウィンドウ（&W）**と呼ばれるプルダウンメニューオプションがあります。オープンされているウィンドウの一覧は、**ウィンドウリスト**と呼ばれる入力タイプで表示されます。必要に応じて別の場所に設定することもできます。



2. 次に、このメニューに表示させるようにプログラムを設定する必要があります。**フォーム特性**の**ウィンドウリストに表示**特性を *Yes* に設定します。**ウィンドウタイプ**特性が ***デフォルト***、***フローティング***、***ツール***、***MDI 調整***の場合に対してのみ、この設定を行うことができます。
3. フォーム名がウィンドウリスト上で表示されるため、各ウィンドウ毎にユニークなフォーム名を定義するようにしてください。この例では、各ウィンドウ毎に顧客の名前をフォーム名として表示させるように、パラメータを使用して式でフォーム名を指定しています。

これでプログラムが実行されると、**ウィンドウメニュー**に表示されます。

実行時にメニューの表示／非表示を行うには

MnuShow() 関数を使用することで、実行時にメニュー項目を表示させないようにすることができます。また、**MnuEnable()** または **MnuCheck()** を使用することで、表示されているメニューを機能を制御することができます。

注： メニュー特性の**権利**特性が設定されており、ユーザはその権利を持っていない場合、メニューは表示されません。

MnuShow() 関数を使用する

構文は以下の通りです。

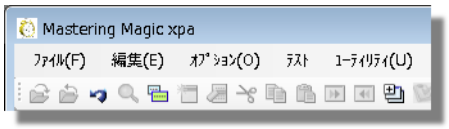
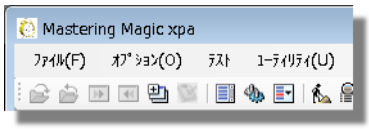
MnuShow(*MenuName*, *True/False*)

パラメータ：

- *MenuName* ：表示させたいメニュー項目の**論理メニュー名**。この例では *EditMenu* が指定されています。
- *True/False* ：指定されたメニュー項目を表示させたい場合は、*'TRUE'LOG* を指定します。表示させない場合は、*'FALSE'LOG* を指定します。この例では、２つのボタンに対応したイベントが定義されています。１つは表示用、もう１つは非表示用です。

注： 論理メニュー名は、実行時に評価されます。このため、式では固定の名前を指定する必要はありません。

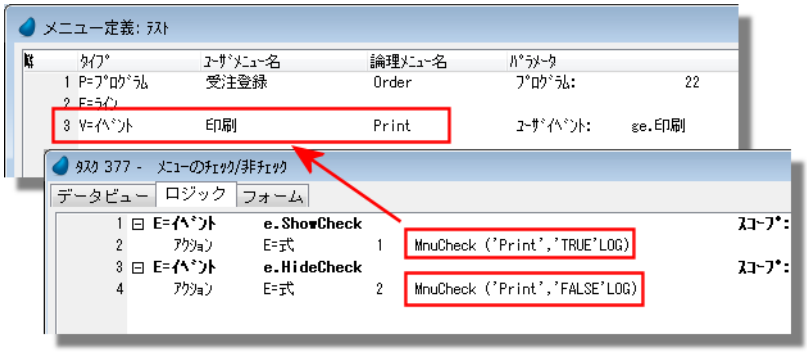
次にどのような結果になるかを紹介します。

	
MnuShow(*'EditMenu','TRUE'LOG*)	**MnuShow(*'EditMenu','FALSE'LOG*)**

実行時にメニューのチェックの表示／非表示を行うには

MnuCheck() 関数を使用することで、実行時にメニュー項目のチェックの表示／非表示を切り替えることができます。

MnuCheck() 関数を使用する



構文は以下の通りです。

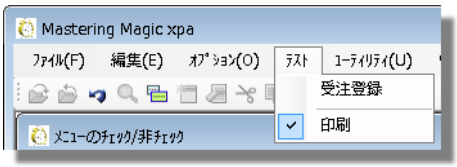
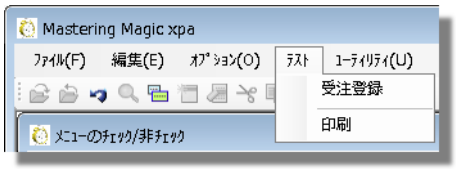
MnuCheck(*MenuName*, *True/False*)

パラメータ：

- *MenuName* ：チェック表示させたいメニュー項目の**論理メニュー名**。この例では *Print* が指定されています。
- *True/False* ：指定されたメニュー項目にチェックを表示させたい場合は、*'TRUE'LOG* を指定します。表示させない場合は、*'FALSE'LOG* を指定します。この例では、２つのボタンに対応したイベントが定義されています。１つは表示用、もう１つは非表示用です。

注： **論理メニュー名は、実行時に評価されます。このため、式では固定の名前を指定する必要はありません。**

次にどのような結果になるかを紹介します。

	
MnuShow(*'Print','TRUE'LOG*)	**MnuShow(*'Print','FALSE'LOG*)**

実行時にメニューの有効／無効を行うには

MnuEnable() 関数を使用することで、実行時にメニュー項目の有効／無効を切り替えることができます。メニューを無効にした場合、グレーで表示され、メニューをクリックしても動作しないようになります。

MnuEnable() 関数を使用する

構文は以下の通りです。

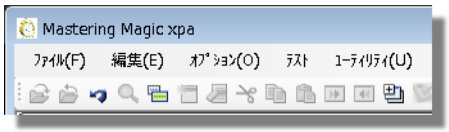
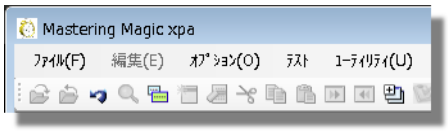
MnuEnable(*MenuName*, *True/False*)

パラメータ：

- *MenuName* ：有効にしたいメニュー項目の**論理メニュー名**。この例では *EditMenu* が指定されています。
- *True/False* ：指定されたメニュー項目を有効にさせたい場合は、*'TRUE'LOG* を指定します。無効にさせたい場合は、*'FALSE'LOG* を指定します。この例では、２つのボタンに対応したイベントが定義されています。１つは有効用、もう１つは無効用です。

注： 論理メニュー名は、実行時に評価されます。このため、式では固定の名前を指定する必要はありません。

次にどのような結果になるかを紹介します。

	
MnuEnable(*'EditMenu','TRUE'LOG*)	**MnuEnable(*'EditMenu','FALSE'LOG*)**

メニューバーを削除するには

MDI または SDI アプリケーションからメニューバーを削除することができます。以下にその方法を示します。

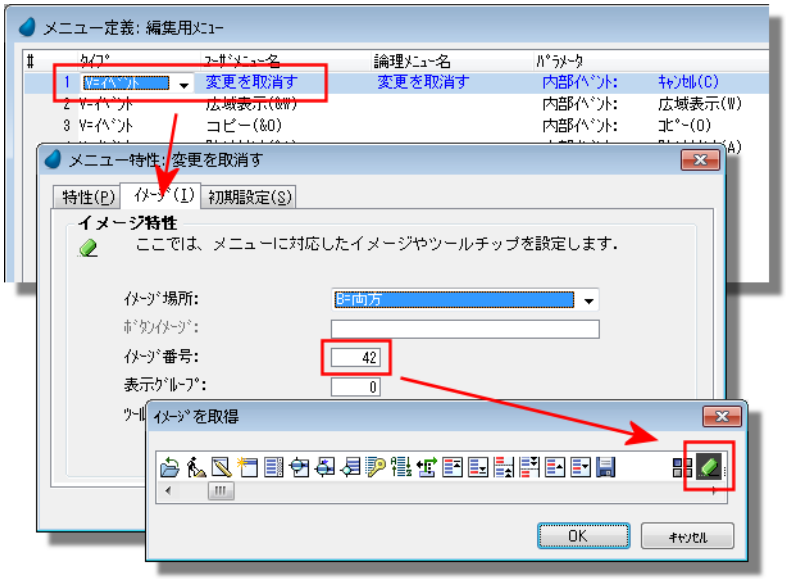
メニュー	MDI	SDI
プルダウンメニュー	メインプログラムの**フォーム特性**→**SDI** →**プルダウンメニュー**特性を *0* に設定します。	**フォーム特性**の**プルダウンメニュー**特性を *0* に設定します。 または、 **フォーム特性**の**メニュー表示**特性を *No* に設定します。
ツールバー	メインプログラムの**フォーム特性**→**SDI** →**ツールバー表示**特性を *No* に設定します。	**フォーム特性**の**ツールバーの表示**特性を *No* に設定します。
ステータスバー	メインプログラムの**フォーム特性**→**SDI** →**ステータスバー表示**特性を *No* に設定します。	**フォーム特性**の**ステータスバーの表示**特性を *No* に設定します。
タイトルバー	メインプログラムの**フォーム特性**→**入力**→**タイトルバー表示**特性を *No* に設定します。	**フォーム特性**の**タイトルバーの表示**特性を *No* に設定します。

メニューにアイコンを追加するには

プルダウンメニューまたはコンテキストメニューにアイコンを追加することができます。同じアイコンをツールバーに表示させることもできます（「ツールバーにアイコンを追加するには」（604 ページ）を参照）。

アイコンは、Magic xpa が提供するリソースファイルから選択したり、ビットマップファイルを指定することができます。

メニューアイコンを指定する



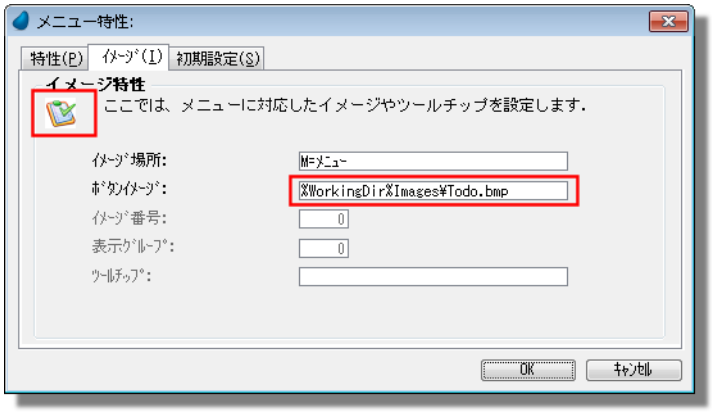
メニュー特性で指定することによりメニューにアイコンを追加することができます。

1. メニューリポジトリ（*Shift+F6*、またはプロジェクト→メニュー）に移動します。
2. アイコンが必要なメニュー項目に移動します。*Alt+Enter* を押下して、メニュー特性を開きます。
3. イメージタブをクリックします。
4. **イメージ場所**特性を***メニュー***（メニューにのみアイコンを表示させたい場合）または ***B= 両方***（ツールバーにも表示させたい場合）のどちらかを設定します。
5. Magic xpa のリソースファイルからアイコンを選択する場合は、**イメージ番号**特性からズームします。表示させたいアイコンを選択し、OK をクリックします。アイコン番号が**イメージ番号**特性に設定され、選択されたアイコンがツールタブの上部に表示されます。

これで、実行時にアイコンがメニューに表示されます。

アイコンとしてビットマップファイルを選択した場合は、以下のようになります。

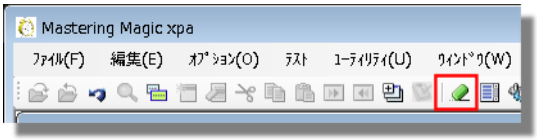
独自のアイコンを指定する



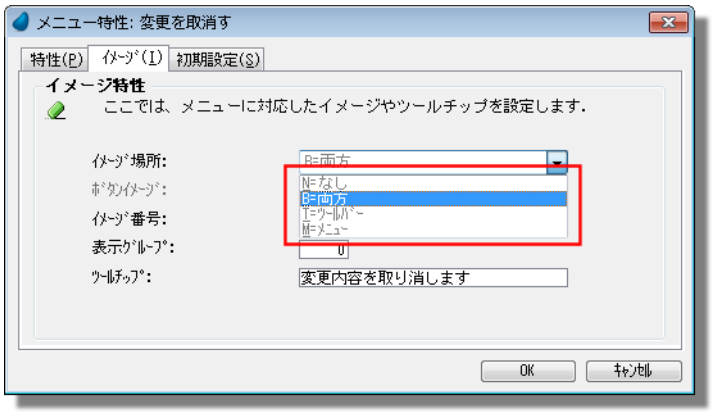
独自のアイコンを指定することもできます。アイコン用のイメージファイルは、16 × 16 ～ 24 × 24 ピクセルの bmp ファイルにしてください。

1. 前述のように、**メニュー特性**を開きます。ここでは、**ボタンイメージ**特性を指定します。ここから**ズーム**してイメージファイルを選択することもできますが、上の図のように論理名を使用した方が汎用性が高くなります。
2. アイコンが、**イメージ**タブの上部に表示されます。

ツールバーにアイコンを追加するには



メニューに対応したツールバーアイコンを追加することができます。



ツールバーにアイコンを追加するには、アイコンをメニューに設定する手順に従って操作します（「メニューにアイコンを追加するには」（602 ページ）を参照）。ただし、**イメージ場所**特性は、***B= 両方***、または ***T= ツールバー***を選択します。

第 27 章 : Unicode

多言語データをサポートできるようにするには

Magic xpa は、多言語データをサポートしています。Magic xpa は、内部で Unicode 形式のデータ書式をサポートしており、また Unicode と他のデータ形式間で変換するためのツールを持っています。

しかし、PC 上の Unicode で動作するため、使用する PC が Unicode を正しくサポートしているかどうかを確認する必要があります。

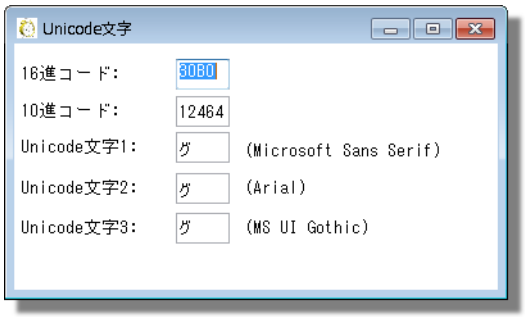
また、Unicode データは、Unicode フォントが設定された Unicode 型の項目を使用して表示させる必要があります。ここでは、これらの内容について説明します。

Unicode フォント

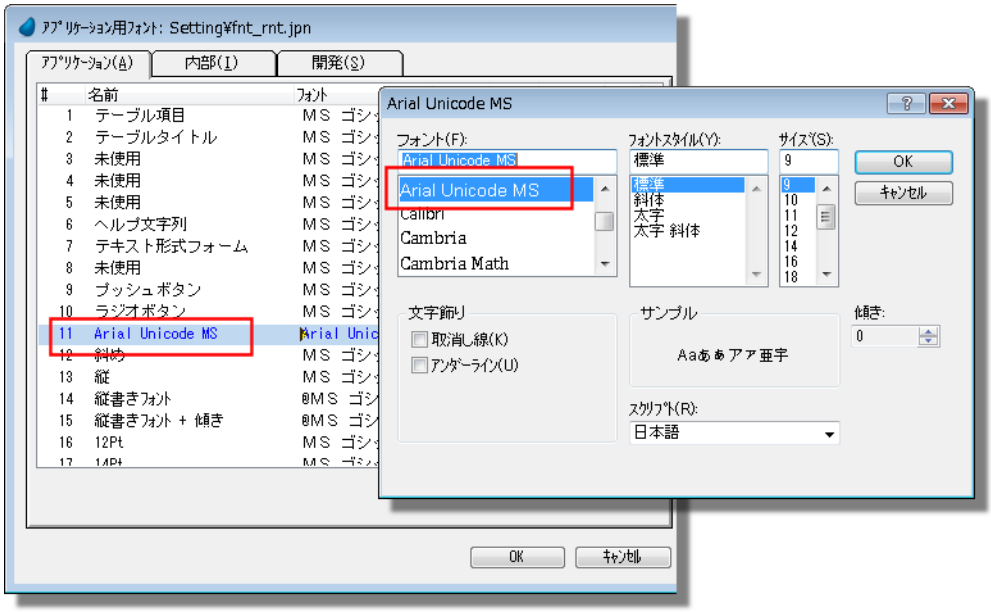
Ascii（JIS）コードを使用していた当初は、文字を 1 バイトで納めることができました。これは、一般的にプログラム言語は、英語を基準にして作成されていたためで、特殊文字を除き、アルファベットの 26 個の文字で十分な状態でした。しかし、" 特殊文字 " が、用法に応じた異なる利用方法が出てきたため、Ascii コード 210 は、使用される ***コードページ***にもとづいて、変形した E や O、またはグラフィックライン文字を表すために使用されるようになりました。

この問題を回避するために、Unicode が開発されました。Unicode は、基本的には、世界中のあらゆる文字を 1 つのコードページに統合したものです。www.unicode.org のサイトには、Unicode と文字の割り当てに関する情報が記載されています。ここには、オリジナルの Ascii フォントが 0 から 128 までに割り当てられていますが、更に様々な国の文字が追加されています。例えば、X'30B0' は日本語の「グ」が割り当てられています。歴史的な文字や架空の文字（クリンゴン文字も存在します）も割り当てられています。

これは考え方としては問題ないのですが、フォントデータが非常に大きなものになってしまいます。このため、Windows XP 以前の OS では、すべての Unicode 文字を参照できるようにフォントデータがインストールされていません。他言語用の文字を表示させたい場合は、必要に応じてフォントをインストールする必要があります。Windows Vista 以降は、Unicode 対応のため欧文フォントでも日本語が表示されます。

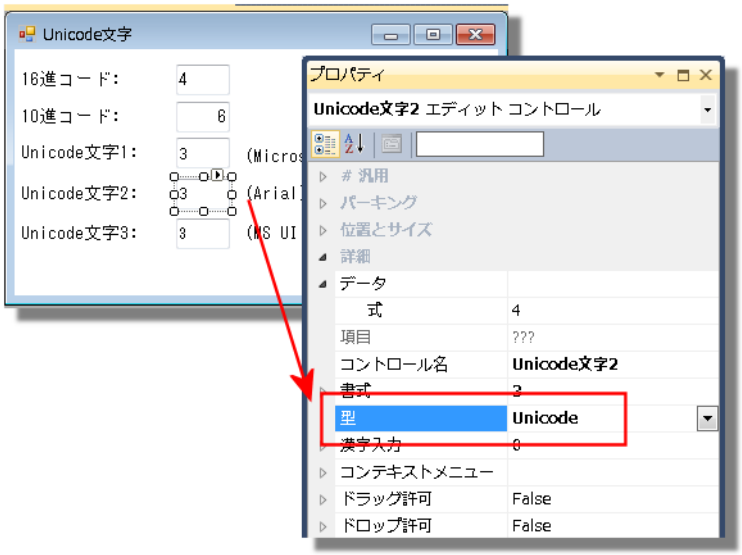


特定の言語で文字を表示している場合は、Windows XP ではその言語用の Unicode フォントが必要になります。上記の例では、日本語の Unicode 文字 H'30B0' で表示されていますが、図に表示されているほど綺麗には表示されません。日本語の専用フォントを使用した方が綺麗に表示されます。



使用したい Unicode フォントを PC にインストールしたら、Magic xpa でそのフォントを使用できるように設定する必要があります。フォント定義テーブル（オプション→設定→フォント）でそのフォントを選択し、Unicode フォントが必要な場合は、これを使用するようにします。この例では、フォント **#11** は Unicode になっています。

Unicode 型

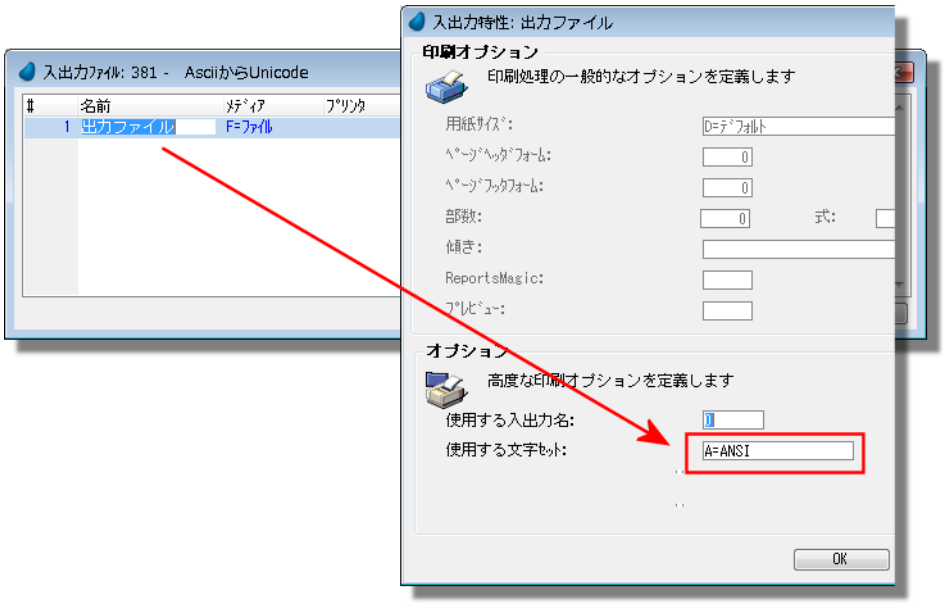


また、Unicode 文字が Unicode 型の項目で処理されていることを確認する必要があります。Unicode データを文字型や BLOB 型（RTF）に変更する場合、正しく表示されない場合があります。

Unicode のテキストファイルの作成 / 読み込みを行うには

Magic xpa では、Unicode 形式のテキストファイルを他の形式と同じように読み書きすることができます。この場合、以下の２つの処理が必要です。

- Unicode データを Unicode 型の項目に格納します。ANSI（JIS）コードから変換する必要がある場合、**UnicodeFromANSI()** 関数を、ANSI（JIS）コードに変換する場合は、**UnicodeFromANSI()** 関数を使用します。
- 以下に示すように、入出力特性の**使用する文字セット**特性で ***Unicode*** を選択します。

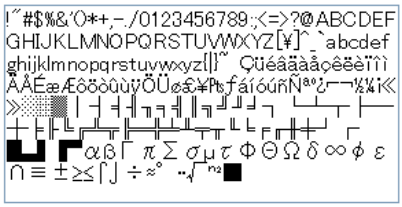


自動的に Unicode でフォーマットされます。

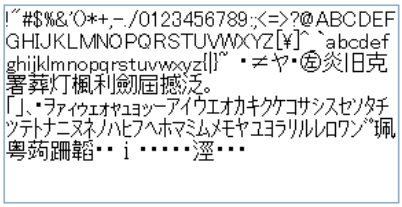
ANSI と Unicode 間でデータを変換するには

Ansi と Unicode 間で変換する場合、異なる ANSI のコードページが存在することを認識しておく必要があります。0 から 255（x'00' から x'FF'）の間の数字コードのみが使用されている場合、それを解釈するために使用されたコードページにもとづいて、異なる文字が表示されます。異なるコードページは、現在サポートされている言語にもとづいて使用されます。

例えば、#00 から #255 までの文字コードをコードページ 437（IBM PCM 基本）と 932（Shift-JIS）で表示させた場合、以下のようになります。



コードページ 437



コードページ 932

Unicode は、これらのすべての文字に対して文字変換が可能ですが、変換をする際に、コードページによって変換される文字が異なることを知っておく必要があります。

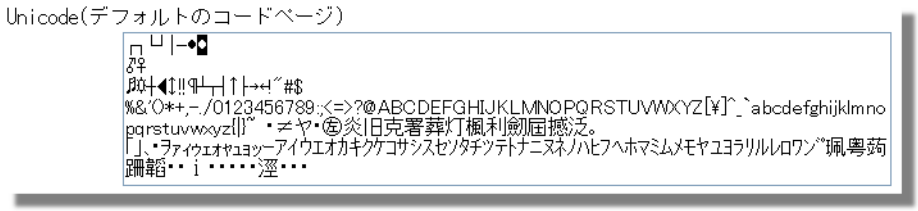
文字を ANSI から Unicode に変換する関数は、以下の通りです。

UnicodeFromANSI (*String*,*CodePage*)

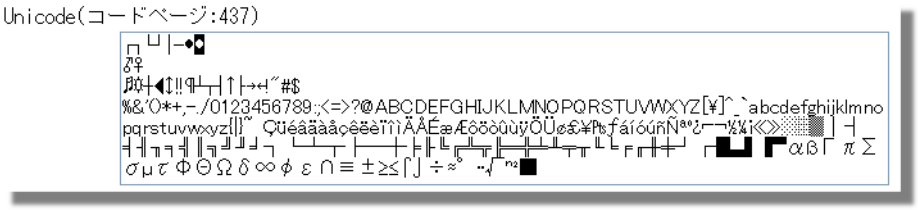
パラメータ：

- *String* ：変換元の ANSI（JIS）文字列
- *CodePage* ：変換の際に使用するコードページ

従って、M がすべての Ascii コードで共通な文字列の場合、**UnicodeFromANSI (*M*, *0*)** と指定できます。



UnicodeFromANSI (*M*, *437*) を実行した場合、以下のような結果になります。



Unicode コードを ANSI に変換する

以下の関数を使用して Unicode を ANSI 変換することができます。

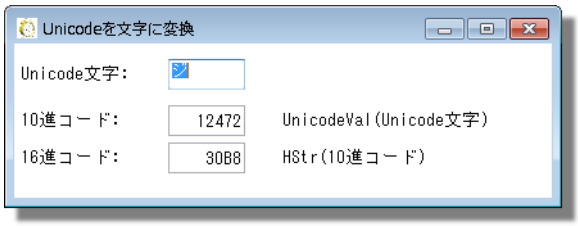
UnicodeToANSI (*String*,*CodePage*)

パラメータ：

- *String* ：変換元の Unicode 文字列
- *CodePage* s：変換の際に使用するコードページ

Unicode 文字の文字コードを取得するには

UnicodeVal() 関数を使用して Unicode 文字を 10 進数の値に変換することができます。この例では、カタカナ文字を 10 進値 12,472 に変換しています。



UnicodeVal(*UnicodeCharacter*)

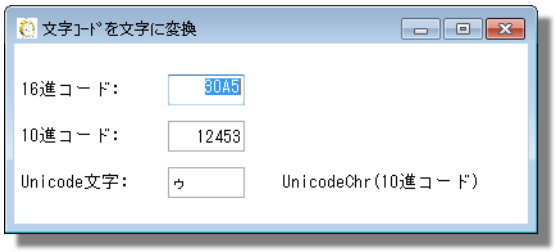
パラメータ：

- *UnicodeCharacter* ：Unicode 型の文字

指定された文字に対応した 10 進数の値が返ります。

Unicode の文字コードを文字に変換するには

UnicodeChr() 関数を使用して、10 進数の数値を Unicode 文字に変換することができます。



UnicodeChr(*Number*)

パラメータ：

- ***Number*** ：Unicode 値を表す 10 進数の数値

指定された数値に対応した Unicode 文字が返ります。

注： 指定する数値は 10 進数にする必要がありますが、Unicode のコードページは一般的に 16 進で指定します。このため、**HVal()** 関数を使用することで、10 進数に変換するすることができます。例えば、**UnicodeChr(HVal(*30A5*))** は、上記の例と同じ結果 **UnicodeChr(*12453*)** になります。

画面に UTF8 の文字列を表示するには

外部ソース（例えば、Web サービスやテキストファイル）から、UTF8 の文字列を取得する場合があります。UTF8 の文字列を参照するためには、それを ANSI に変更する必要があり、次に ANSI 値を Unicode に変更する必要があります。

これを行うには、UTF8ToAnsi() 関数を UnicodeFromANSI() 関数に埋め込む必要があります。

構文は、例えば以下のようになります。

UnicodeFromANSI(UTF8ToAnsi(*A*),*932*)

パラメータ：

- *A* …… UTF8 の文字列を含む BLOB 項目
- *932* …… 文字列のコードページ（日本語 :Shift-JIS）

注： データに 1 つの言語が含まれている場合にのみ動作します。

［このページは意図的に空白にしています。］

第 28 章 : アプリケーションのデバッグ

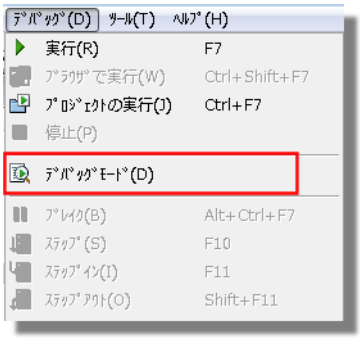
デバッガを使用してデバッグするには

Magic xpa には、デバッグツールが組み込まれています。このデバッガを使用すると以下のことが可能になります。

- エンジンの動作内容の確認
- コールスタックやデータベースアクセスの表示
- プログラム実行時に処理される内容の表示
- すべてのコンテキストの表示
- メモリ上のデータ項目の内容を表示／更新

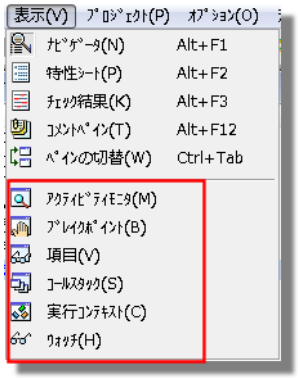
ここでは、デバッガの使用方法について説明します。

1. デバッグモードを有効にする



最初にデバッグモードを有効にする必要があります。デバッグ→デバッグモードを選択するか、ツールバー上のアイコンをクリックすることで有効になります。デバッグモードが選択されると、アイコンが押下された状態で表示されます。

2. モニタ用ウィンドウを開く



デバッガが実行されると、対応するウィンドウ内にデータが表示されます。これらのウィンドウは、1 つに統合して表示させたり、個別に表示させたりすることができます。これらウィンドウのうちどれを表示させ、どれを表示させないかを指定することができます。

プルダウンメニューの表示メニューから表示させたいウィンドウをクリックすることで、該当するモニタ用ウィンドウが開きます。またデバッグ処理中にウィンドウを開くこともできます。

3. ブレイクポイントを設定する

次に、何処で停止させるかを指定する必要があります。これには 2 つの方法があります。

- ブレイクポイントを設定します。処理コマンドや項目の値が更新される場所に設定できます。
- ブレイクを使用します。

これらの使用方法については、「アプリケーションのブレイクポイントを設定するには」（617 ページ）を参照してください。

項目やコールスタック、およびコンテキストはブレイク中でのみ参照できるため、ブレイクポイントの設定やブレイクの使用は重要になります。しかし、プログラムの実行中にこれらを設定することも可能です。

4. プログラムやプロジェクトを実行する

次に、プログラムまたはプロジェクトを実行します。1 つのプログラムをデバッグしているだけなら、そのプログラム上にカーソルを置き *F7* を押下します（デバッグ→実行）。開発中のタスクを閉じる必要はありません、Magic xpa はプログラムを実行する前に変更内容を自動的に保存します。

プロジェクト全体をテストしたい場合、*Ctrl+F7*（デバッグ→プロジェクトの実行）を押下してください。

5. プロジェクトをステップ実行する

ブレイクポイントが設定されたら、実行ウィンドウ内で実行内容を参照しながら、処理コマンド毎にプログラムの処理ステップを進めることができます。以下の表に示すように、いくつかのステップオプションがあります。

コマンド	ショートカットキー	処理内容
実行	*F7*	次のブレイクポイントに到達するまで、プログラムを実行させます。
ステップ	*F10*	現在のプログラムの次の処理コマンドに進み、停止します。しかし、処理コマンドが別プログラムを呼び出す場合、コール処理コマンドで停止しません。
ステップイン	*F11*	ステップコマンドと同じように動作しますが、コール処理コマンドやイベント実行処理コマンドの場合、呼び出されたタスク／プログラム内に入った状態で処理が停止します。
ステップアウト	*Shift+F11*	現在のハンドラ内で処理が順番に停止しますが、ハンドラの起動内で処理を再開します。
ブレイクポイント	*F9*	ブレイクポイントやステップコマンドによって処理が停止している間に、既存のブレイクポイントを解除したり設定を追加したりすることができます。

デバッグ処理を終了する場合は、デバッグ→停止を選択するか、ツールバーの  をクリックします。プログラムに不具合がある場合、*Ctrl+Shift+F9*（デバッグ→実行エンジンのリセット）を押下することで実行エンジンを再起動させることができます。

リッチクライアントのデバッグを行うには

リッチクライアントタスクは、サーバ側のアクティビティモニタを使用してデバッグすることができます。次のイベントが処理されると、クライアント側のエントリがアクティビティモニタに追加されます。

クライアント側をデバッグする：

1. サーバ側の**ロギングの設定**（**オプション→設定→ロギング**）で**クライアントの動作**を *Yes* に設定します。この設定を行わないと、サーバ側の動作のみしか表示されません。
2. **Magic.ini** ファイルの［MAGIC_RIA］セクションで以下のように設定します。
   - InternalLogFile …… クライアント側の実行ログを格納するファイル名を設定します。
   - InternalLogSync …… ログファイルのオープン／クローズのタイミングをコントロールすることができます。
   - InternalLogLevel …… ここに有効な値（SERVER、SUPPORT、GUI、DEV）を設定することによって、取得するログの内容を決めることができます。

クライアント側の処理を実行している間にリッチクライアントタスクが異常終了したり、ハングアップした場合、ログの内容はアクティビティモニタに表示されません。したがって、Magic xpa はクライアント側の動作手順をデバッグさせるためのソリューションを提供します。

クライアント側の障害をデバッグする：

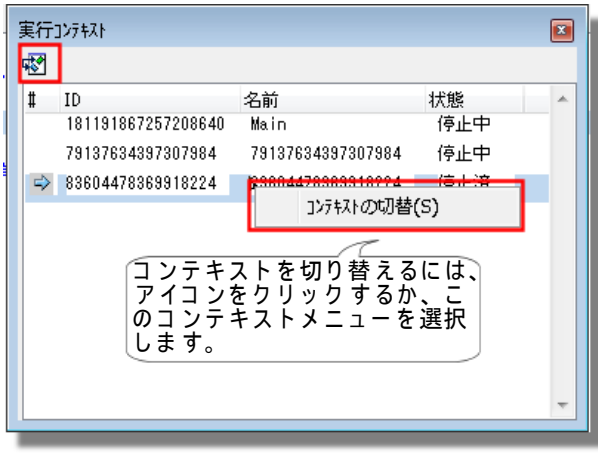
1. アプリケーション用の **Published.html** の **LogClientSequenceForActivityMonitor** を *Yes* に設定します。このファイルは、リッチクライアントインタフェースビルダを使用するとことで作成されます。*Yes* に設定すると、クライアント動作内容がクライアント側のローカルログファイルに書き込まれます。このログファイルは、サーバにまだ送信されていないすべての情報が含まれています。ログファイルは、各イベントが実行されたりアプリケーションが終了される時点で削除されます。
2. アプリケーションを起動します。Magic xpa は、既存のログファイルを調べます。最後に異常終了してからの記録が見つかった場合、それらをサーバ荷送り、アクティビティモニタに表示されます。
3. 異常終了が発生した場合、他の全ての並行処理が終了します（これによって、通常は他の処理のログは削除されます）。
4. アプリケーションを再開します。これによってクライアント側の最後の動作手順によるログファイルがサーバに送られ、アクティビティモニタに表示されます。アクティビティモニターは、クライアント側のログで更新されます。そして、次に問題の場所を特定することができます。

注： アクティビティモニタの表示行は、サーバ側はピンク、クライアント側は白で表示されます。

サーバ側の処理（またはサーバ側で実行されるニュートラルな処理）に対するブレークポイントでは、Magic エンジンが停止し、開発者のアクションを待つ状態になります。クライアント側の処理（またはクライアント側で実行されるニュートラル側な処理）に対するブレークポイントは、無視されます。ブレイクは、次のサーバ処理を停止させます。

クライアントの処理をステップ実行すると、デバッガはサーバに戻った後で停止するため、複数のクライアント処理がある場合、デバッガから干渉されずに実行されます。

並行プログラムの実行中にデバッガを使用するには



複数のコンテキストを同時に実行させている場合、デバッガは便利です。コンテキストビューは、既存のすべてのコンテキストのリストを表示します。

実行中のコンテキストリストから、コンテキストメニューの**コンテキストの切替**を選択するか、  アイコンを使用することでコンテキスト間の切り換えを行うことができます。ステータスが停止済のコンテキストのみこの機能は有効です。このため、コンテキストを切り替えたい場所にブレイクポイントを追加する必要があります。

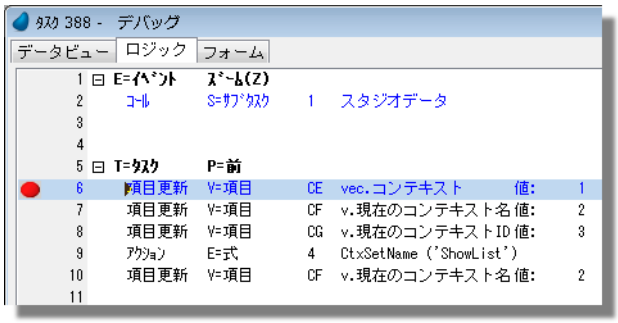
アプリケーションのブレイクポイントを設定するには

プログラムの実行中に処理内容を参照するためにブレイクポイントを設定します。ブレイクポイントには、以下の３つの異なる設定方法があります。

- 処理コマンドに設定する
- データ項目に設定する
- ブレイクを使用する

ここでは、ブレイクポイントの設定方法について説明します。

処理コマンドでの設定



ブレイクポイントは、開発エンジンで設定します。デバッガがブレイクポイントに到達した場合、処理が停止します。デバッガによって処理が停止すると、項目やコールスタック、およびコンテキストの内容を参照することができます。

ブレイクポイントを設定する

ブレイクポイントを設定するには以下のようにします。

1. ロジックエディタを開き、停止させたい処理コマンドの設定行に移動します。
2. *F9*（デバッグ→ブレイクポイント）を押下します。

同じ場所で再度 *F9* を押下するとブレイクポイントは解除されます。しかし、ブレイクポイントペインから削除することでも解除できます。この方法は、デバッグ終了後にすべてのブレイクポイントを解除する場合に便利です。

プログラムの実行中に、ブレイクポイントを設定することもできます。プログラムは読み込み専用モードでオープンされるためプログラム自体の変更はできませんが、ブレイクポイントの設定／解除は可能です。

さらに、以下に示すように、ブレイクポイントペインでブレイクポイントの処理を修正することができます。

ブレイクポイントを切り替える

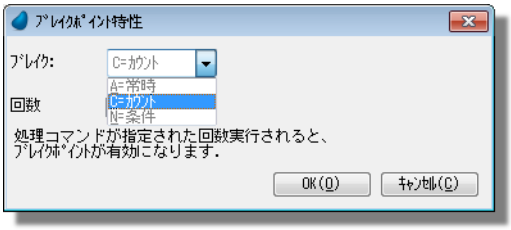
設定されたブレイクポイントは、ブレイクポイントペインに表示されます。ここでは、色々な方法で設定内容を修正することができます。

- 有効のチェックボックスをチェック／アンチェックすることで、ブレイクポイントの有効／無効が切り替わります。

   ですべて有効になり、  ですべて無効になります。

- コンテキストメニュー（または *F3* の押下、または  アイコンのクリック）から削除を選択することでブレイクポイントを削除できます。  をクリックするとすべて削除されます。

ブレイクポイント特性を設定する



さらに、ブレイクポイント特性を使用して、ブレイク処理を微調整することができます。

1. 修正したいブレイクポイント上にカーソルを置きます。
2. コンテキストメニューから**特性**を選択します（または  のクリックか *Alt+Enter* の押下）。ブレイクポイント特性が表示されます。
3. **ブレイク**特性で、*A= 常時*、*C= カウント*、または *N= 条件*を選択します。
   - *A= 常時*を選択した場合、その他は何もする必要はありません。このポイントを通過した場合は、必ず停止します。
   - *C= カウント*を選択した場合、停止させたい繰り返し数を入力してください。例えば、1,000 回ループする処理がある場合、カウントを 300 に設定することで、最初の 300 回の繰り返し後に処理が停止します。
   - *N= 条件*を選択した場合、**条件**特性でズームしてブレイクを有効にさせる条件を設定することができます。条件が True として評価されると、ブレイクポイントで処理が停止します。

ソースに移動する

ブレイクポイントペインでコンテキストメニューから**ソースに移動**を選択すると、ブレイクしている場所に位置付けることができます。

データビューエディタでの設定

データビューエディタ上に定義された項目にブレイクポイントを設定した場合、その項目の値が変更されると、デバッガは処理を停止させます。

ブレイクポイントを設定する

ブレイクポイントを設定するには、以下のようにします。

1. データビューエディタを開き、停止させたいデータ項目の定義行に移動します。
2. *F9*（デバッグ→ブレイクポイント）を押下します。

データ項目の更新によるブレイクポイントは、処理コマンドの場合と同じようにブレイクポイントリストに表示されます。

項目にウォッチに設定する

これは、データ項目の内容を監視するための簡易リストです。**ウォッチリスト**には項目を追加することができます。データ項目が更新されブレイクが発生すると、項目の内容を参照することができます。**ウォッチリスト**に項目を追加するには以下のようにします。

1. **データビューエディタ**を開き、追加したい項目（カラムか変数／パラメータ）にカーソルを置きます。
2. **デバッグ→ウォッチに追加**（または、コンテキストメニューから**ウォッチに追加**）を選択します。

処理コマンドに定義した時のように**ブレイクポイント**を設定しても、項目の隣に赤いドットは表示されませんが、**ウォッチ**ペインに項目リストが表示されます。**ウォッチ**ペインに表示される項目は、**項目**ペインにも表示されますが、指定された項目だけが表示されるので**項目**ペインより便利な場合があります。

ウォッチペインでは以下の処理が可能です。

- 項目名を削除することでウォッチを解除できます（または  を**クリック**するか、*F3* を**クリック**するか、コンテキストメニューから**行削除**を選択します）。
- 項目の値を別の値に変更できます（コンテキストメニューから**データの設定**を選択するか、  を**クリック**します）。
- 設定可能であれば、NULL 値を設定できます（コンテキストメニューから**データに Null を設定**を選択するか、  を**クリック**します）。
- ソースコードのデータ定義行に移動できますコンテキストメニューから**ソースに移動**を選択するか、  を**クリック**します）。

ブレイクの使用

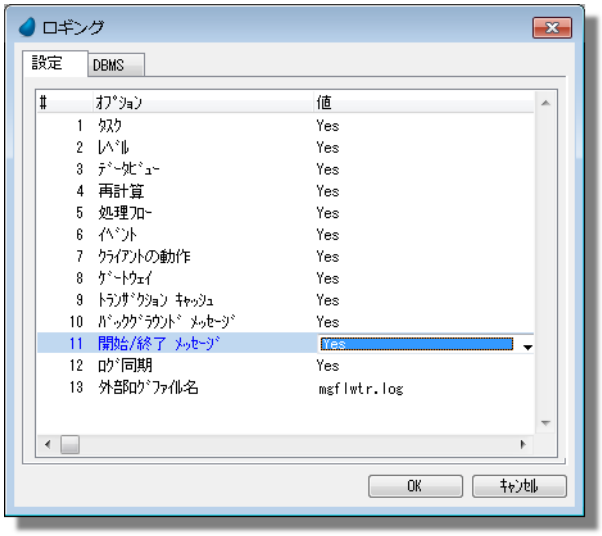
ブレイクポイントやウォッチが設定されていない場合でも、項目の内容やコールスタックを参照するために**ブレイク**を使用することができます。（特にオンラインの）プログラムの開発時、それをテストしていて予期しない結果が返るような場合、この機能は便利です。項目やコンテキスト、およびコールスタックはブレイクしている場合のみ参照できるため、この機能を利用することで内容を確認することができます。

ブレイクを使用するには以下のようにします。

1. プログラムの実行中に開発ウィンドウを表示します。
2. *Alt+Ctrl+F7* を押下します（**デバッグ→ブレイク**を選択するか、ツールバーの  を**クリック**します）。

デバッガは、ブレイクポイントやウォッチが設定されている場合と同じように停止します。

デバッガによって記録される情報を制御するには



ロギングテーブル（オプション→設定→ロギング）の設定を変更することで、デバッガによって記録される情報を制御することができます。

これらの設定は、デバッガだけでなく出力されるログファイルの内容にも影響します。

Logging() 関数を使用する

Logging() 関数を使用してログ出力の有効／無効を切り替えることもできます。この関数は、Magic.ini のロギングの設定を変更しません。関数の構文は以下の通りです。

Logging(*start/stop*, *Filter*)

パラメータ：

- *start/stop* …… *'TRUE'LOG* を指定するとログの出力が開始されます。*'FALSE'LOG* を指定すると停止します。
- *Filter* …… 対象となるログ内容を指定するキーワード。

指定例は以下の通りです。

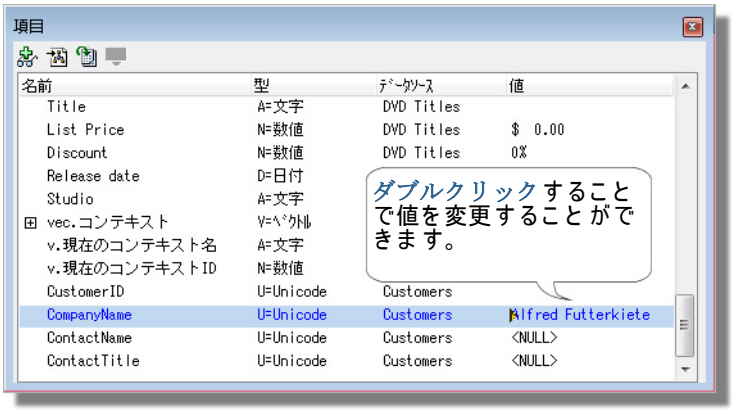
式の例	結果
Logging(*'TRUE'LOG, 'Levels'*)	処理レベルに対するログ出力が開始されます。
Logging(*'FALSE'LOG, 'Recompute'*)	再計算に対するログ出力が停止します。
Logging(*'FALSE'LOG, 'ALL'*)	すべてのログが停止します。
Logging(*'TRUE'LOG, 'RESET'*)	Magic.ini に格納された値に戻します。
Logging(*'TRUE'LOG, 'Oracle=D'*)	Oracle のログを開発者レベルに設定します。

構文についての詳細情報は、『リファレンスヘルプ』を参照してください。

パフォーマンスの問題

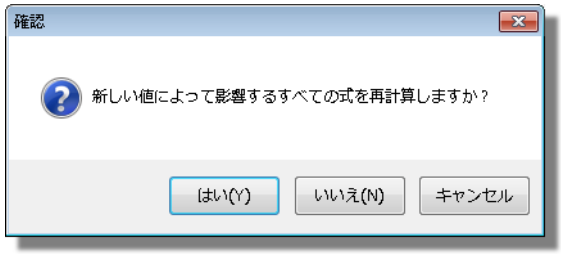
Logging() 関数を使用した場合、キャビネットファイルの作成前に無効になるように設定されているかどうかを確認してください。ログ出力機能が有効な場合、プログラム実行のパフォーマンスが低下します。

デバッグ中にデータを操作するには



デバッガが動作している間は、データ項目の内容を変更することができます。**項目**または**ウォッチ**ペインのどちらかで可能です。

1. 変更したい項目が参照できる処理で停止できるように**ブレイクポイント**を設定します。
2. 変更したい項目を見つけるために、**項目**ペインか**ウォッチ**ペインに移動します。
3. **項目**カラム上でダブル**クリック**するか、コンテキストメニューの**データを設定**を選択するか、  を**クリック**します。
4. 変更された項目が再計算対象の場合、「新しい値によって影響するすべての式を再計算しますか？」という確認メッセージが表示されます。再計算を行う場合は、はいを**クリック**します。



コンポーネントをデバッグするには

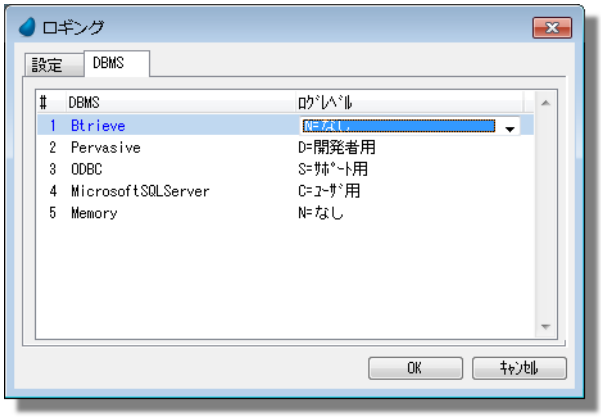
コンポーネントが現在のプログラムのモジュールとして定義されている場合、現在のプロジェクトと同じようにコンポーネントプログラムをデバッグすることができます。

モジュールを現在のプロジェクトに追加するには以下のようにします。

1. **プロジェクト→モジュール追加**を選択します。**プロジェクトを選択**ダイアログが表示されます。
2. 追加したいプロジェクトファイル（.edp ファイル）を選択します。

ナビゲータペインのモジュールツリーにコンポーネントが追加されたら、モジュール名を**クリック**してプロジェクトをオープンします。処理を停止させたい場所に移動してブレイクポイントを設定します。ホストプロジェクトと同じように**ブレイクポイントリスト**に**ブレイクポイント**が表示されます。デバッグ中にコンポーネントに定義された**ブレイクポイント**に達すると、現在のタスクが閉じコンポーネントが開きます。そしてコンポーネント内での実行内容が確認できます。

データベースの動作を記録するには



使用する DBMS へのアクセスを記録するログのオプションもあります。このログを有効にするには以下のようにします。

1. **ロギング**テーブル（**オプション→設定→ロギング**）を開きます。
2. **DBMS** タブを**クリック**します。
3. 記録したい **DBMS** にカーソルを置きます。
4. **ログレベル**を ***N= なし***以外に選択します。ログのレベルは ***C= ユーザ用***、***S= サポート用***、***D= 開発者用***の 3 種類あります。***C= ユーザ用***は最も短く最も簡単なログレベルで、***D= 開発者用***は最も詳細なログレベルです。

ログレベル

ログレベル	動作内容
N= なし	ログは出力されません。
C= ユーザ用	最も簡単な内容のログが出力されます。SQL コマンドのみが出力されます。
S= サポート用	中程度の内容のログが出力されます。
D= 開発者用	詳細な内容のログが出力されます。

DBMS のロギング

Magic xpa のロギング機能に加えて、使用している DBMS に組み込まれているロギング機能を利用することも有効です。機能の内容は DBMS によって変わりますが、何らかのロギングツールが用意されています。

パフォーマンスの問題

ロギング機能を使用している場合、キャビネットファイルの作成前に無効になるように設定されているかどうかを確認してください。ログ出力機能が有効な場合、プログラム実行のパフォーマンスが低下します。

リモートアプリケーションをデバッグするには

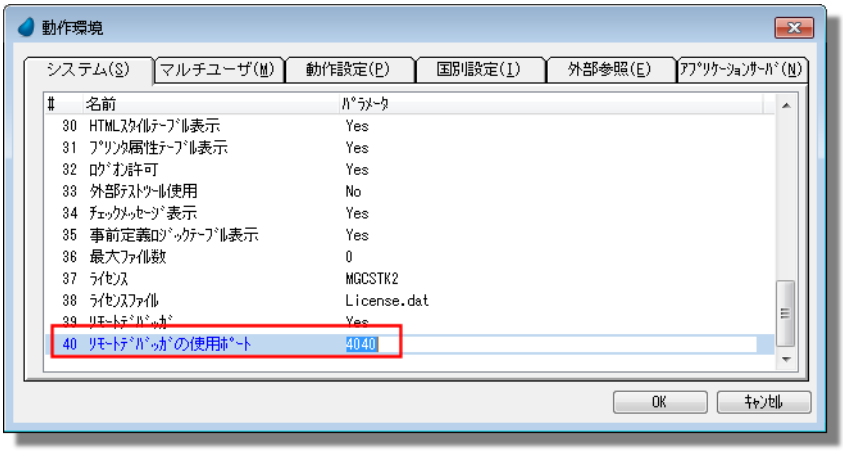
リモートデバッガを利用することで、リモート PC 上で実行しているアプリケーションのデバッグを行うことができます。この場合、リモート側とローカル側は以下のような構成にする必要があります。

- リモート PC : **Magic xpa Enterprise Client** 、または **Magic xpa Enterprise Server** や **Magic xpa RIA Server** でアプリケーションを実行します。
- クライアント PC : **Magic xpa Enterprise Studio** でリモート側で実行している同じアプリケーションのプロジェクトファイルをオープンします。

リモートデバッガを有効にする

最初にリモート PC 側でリモートデバッガを有効にする必要があります。

1. サーバ PC 側の Magic xpa で、**動作環境**ダイアログ（**オプション→設定→動作環境**）を開き、**システム**タブの**リモートデバッガ**を ***Yes*** に設定し、**リモートデバッガの使用ポート**に TCP のポート番号を指定します。ポート番号は、他に使用されていない番号を指定してください（Windows の netstat コマンドなどで確認できます）。

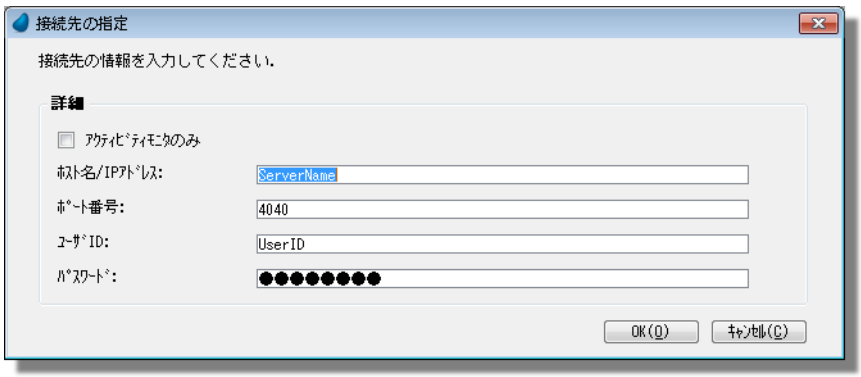


2. Magic エンジンを再起動し、アプリケーション（ecf）をオープンします。

リモートアプリケーションに接続する

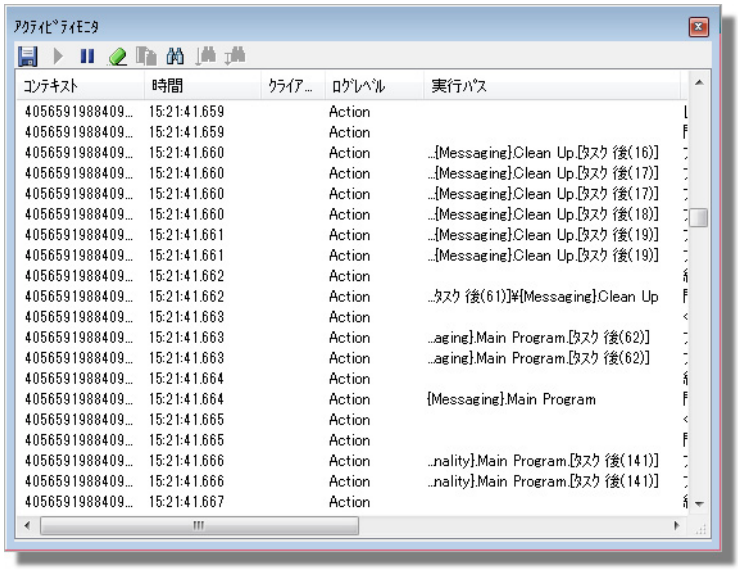
次に、クライアント PC 側でリモートアプリケーションに接続します。

1. クライアント PC 側で **Magic xpa Studio** を起動し、リモート側で実行しているアプリケーションのもとになるプロジェクトをオープンします。
2. **デバッグ→リモートエンジン**に接続を選択し、**接続先の指定**ダイアログを開きます。



  - ホスト名 /IP アドレス …… リモート PC のホスト名か IP アドレスを指定します。
  - ポート番号 …… リモート PC 側の動作環境ダイアログで設定したポート番号
  - ユーザ ID …… リモート PC 側の Magic xpa 環境で登録されているユーザ ID を指定します。この ID は、アプリケーションをアクセスする権利やリモートデバッガにアクセスする権利も持っている必要があります。
  - パスワード …… ユーザ ID に対応したパスワードを指定します。
- アクティビティモニタのみ …… 通常は、リモート側のアプリケーションファイルとローカル側のプロジェクトファイルは同期がとれている必要がありますが、ここをチェックすることでアクティビティモニタ表示のみサポートすることでプロジェクトが異なる状態でも実行できるようになります。

全てのパラメータを設置し、OK をクリックすると通常のデバッガと同じようにモニタウィンドウにデバッグ情報が表示されるようになります。



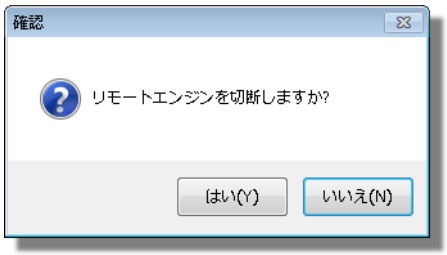
リモートアプリケーションを切断する

リモートアプリケーションのモニタを終了させる方法は、２つあります。

- リモート側でアプリケーションを終了させる。
- クライアント側で切断処理を行う。

クライアント側で切断処理を行う

1. クライアント PC 側でデバッグ→リモートエンジンを切断を選択します。確認ダイアログを開きます。はいをクリックするとリモート側との接続が終了します。



［このページは意図的に空白にしています。］

第 29 章 : イベントとハンドラ

注 : V9Plus 以前における「イベントハンドラ」は、Magic V10 以降「ロジックユニット」という名称になりました。ただし、ロジックユニットを実行する上での内部機構をこのドキュメントではイベントハンドラ（またはハンドラ）という表記で説明しています。また、処理レベル（レコード前／後、タスク前／後、等）と区別するためにも、この章ではイベントハンドラという表記を使用しています。

イベントによって起動された選択プログラムの戻り値でデータ項目を再表示するには

ハンドラ（ロジックユニット）で処理される場合、ハンドラを抜けるまで項目の値は更新されません。これは、フォームに配置されているデータ項目がハンドラ内で更新された場合、ユーザが表示項目を抜けるまで更新された値が画面上に表示されないことを意味しています。

これは、選択リストを使用する場合に特に問題になります。一般的に、選択リストは表示項目からズームすることでアクセスされます。ここでは、このようなズーム処理を実現する場合の定義方法について説明します。

問題点 : 項目が再表示されない

最初に、正しく動作しないズーム処理の例を示します。ユーザがスタジオコードコントロールでパークしている時にズームすると、スタジオの選択プログラムが起動され、スタジオコードを選択できるようになっています。しかし、カーソルが次のコントロールに移動するまで項目は空白の状態で表示されるはずです。

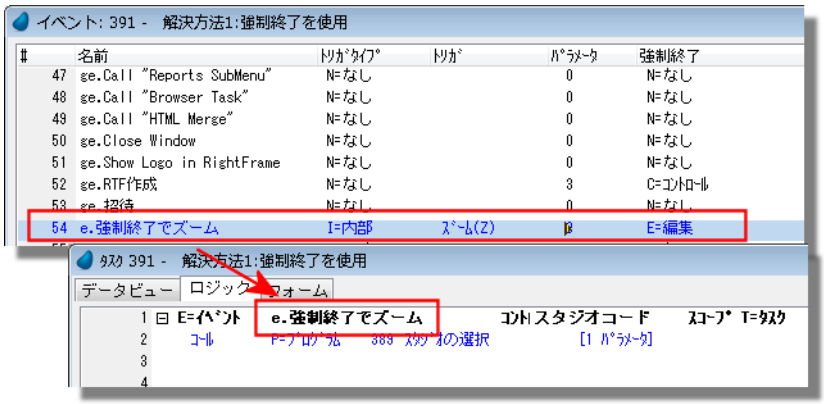
解決方法 :

更新された項目が再表示されない問題点の解決方法には、以下の 3 つの方法が考えられます。

- イベントの**強制終了**特性を使用する
- **コントロール特性**の**選択プログラム**特性を使用する
- **コンボボックス**を使用する

以下、これらの詳細について説明します。

解決方法 1: イベントの強制終了特性を使用する



項目を更新する場合に発生する問題点の解決方法は、次の通りです。

1. 新しいユーザイベントを作成します（この例では、e. 強制終了でズームという名前で**強制終了**特性を設定しています）。このイベントは、ズームの内部イベントが割り当てられているため、ズームと同じよう動作します。**強制終了**

特性には、図で示されているように、*E= 編集*に設定されています。これは、表示項目を再表示させるように定義するものです。

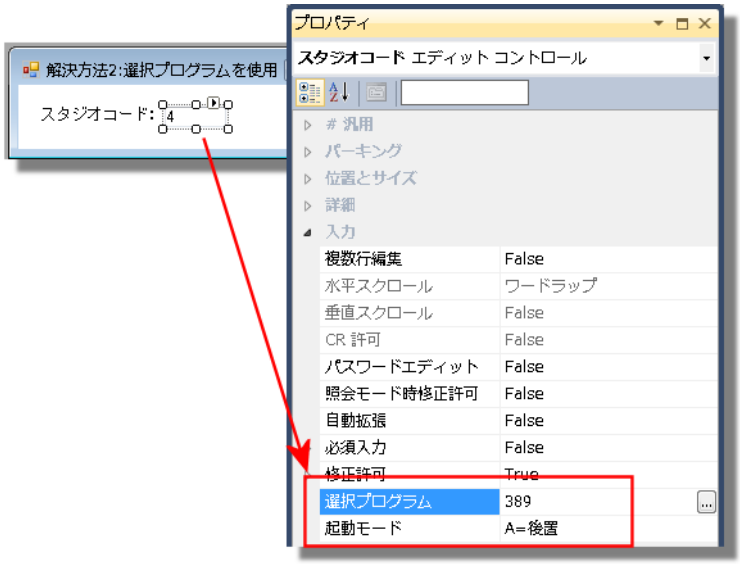
2. 新しいユーザイベントを内部イベントのズームの代わりに使用します。

メインプログラム内でこの新しいイベントを作成した場合、アプリケーションのグローバルイベントとして使用できます。

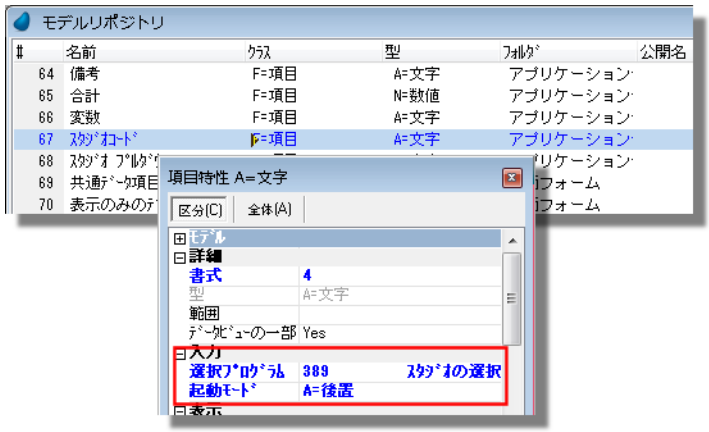
注： 通常、ズーム項目にパークしている場合、ステータスバーに「ズーム」という文字が表示されます。上記の方法でロジックを定義した場合、このイベントはズームをトリガとして定義されていため、同じように「ズーム」が表示されます。

ヒント：後置の ズーム処理を定義したい場合、次項目の内部イベントを発行するイベント実行処理コマンドを追加してください。

解決方法 2: 選択プログラム特性を使用する



別の方法として、イベントを使用しないで選択リストを起動させことができます。この場合、表示項目に**選択プログラム**特性を定義するだけで対応できます。

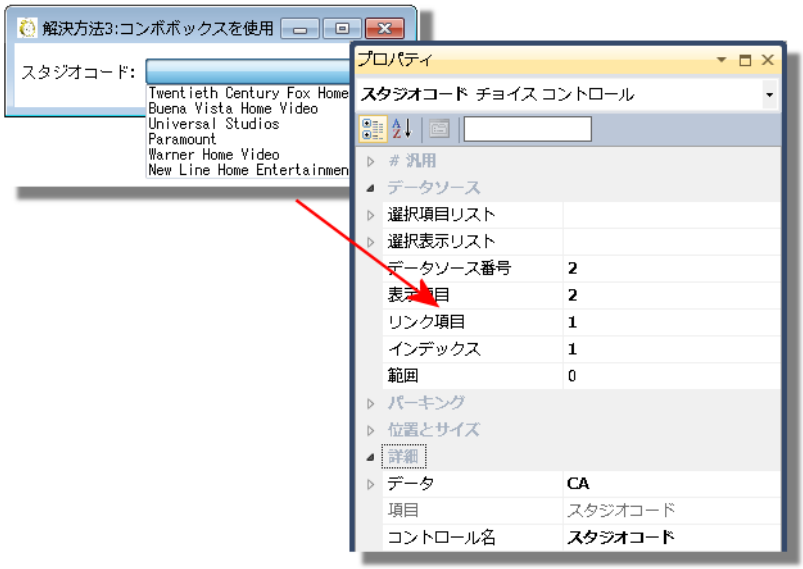


選択プログラム特性は、モデルやフォーム上のコントロールに対して定義できます。この特性が定義されるとズームが有効になり、指定された選択プログラムを起動することができるようになります。

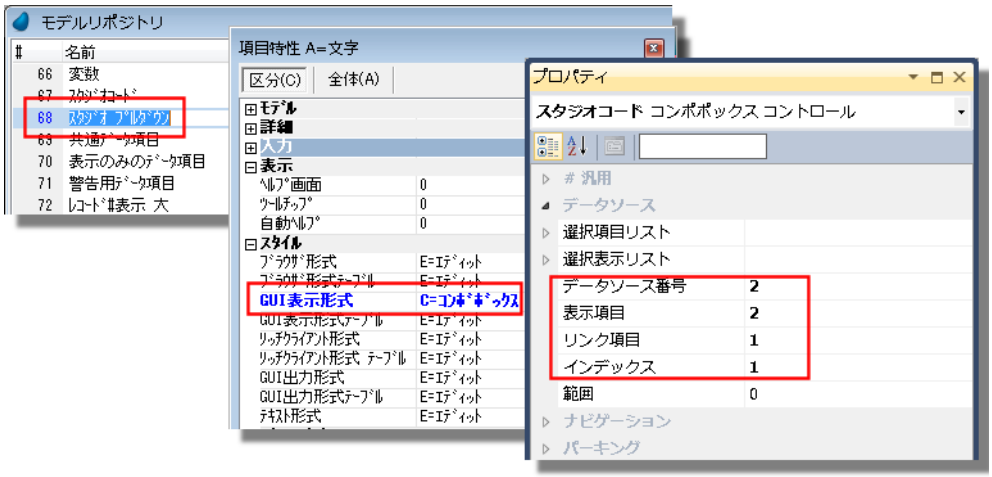
この方法は、タスク毎にロジックを定義する必要がなくなる点で便利です。この例では、項目モデルに対して**選択プログラム**特性で**スタジオの選択**プログラムが定義されています。プログラム側でこのモデルとの継承関係を解除しない限り、常にこのプログラムが起動されるようになります。

この方法の不便な点は、1 つの値しか返らないということです。この例では、スタジオコードのみ選択していますが、スタジオコードとスタジオタイプの両方を取得する必要がある場合、イベントを使用することになります。

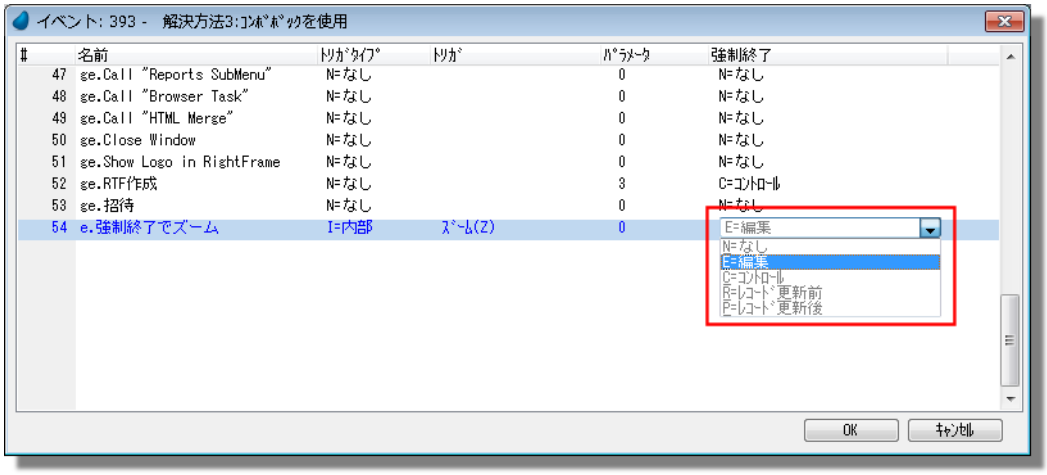
解決方法 3: コンボボックスを使用する



Magic xpa では、複数の項目から選択するための方法として**データコントロール**（コンボボックス、リストボックス、タブ）を使用することもできます。この場合も、モデルで定義することができるので、プログラム毎に定義する必要はありません。以下の例では、**スタジオプルダウン**という名前のモデルを作成しています（これは、フォームにはコンボボックスとして配置され、スタジオテーブルと自動的に連動するようになっています）。



確定されていない更新値で強制的に更新するには



ハンドラで処理される場合、ハンドラを抜けるまで項目の値は更新されません。これは、フォームに配置されているデータ項目がハンドラ内で更新された場合、ユーザが表示項目を抜けるまで更新された値が画面上に表示されないことを意味しています。

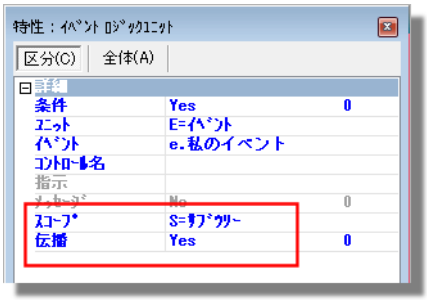
別のケースとして、ハンドラ内の処理が完了する前にレコードを更新したい場合もあります。例えば、まだ編集中の受注データを印刷するためにプログラムを起動する例を考えてみます。この場合、プログラム起動時にレコードへの書き込みが行われなければなりません。

このような処理は、イベントテーブルの**強制終了**カラムを使用することで対応できます。このパラメータの詳細は、『リファレンスヘルプ』を参照してください。また動作の違いを理解するために、パラメータの設定を変えて、デバッガを使用して確認してみてください。**強制終了**を ***E= 編集***に設定することで、項目の更新処理の問題の解決に役立ちます。

同一イベントで複数のハンドラを実行させるには

タスク内で発行されるイベントは、上位に伝播させることができます。イベントが子タスクで発行された場合、**メインプログラム**を含めたすべての上位タスクで処理させることができます。

イベントの伝播を許可する



イベントツリー内のすべての**ロジックユニット特性**を設定することで、発行されたイベントの処理を伝播させるかどうかを制御できます。

1. **伝播**特性を ***Yes*** に設定します（'TRUE'LOG と評価される論理式でも指定できます）。
2. 最下位レベルのタスクでなければ、**スコープ**特性を ***S= サブツリー*** に設定します。

これでイベントは、1 つのタスクレベルで処理された後、次のレベル（上位タスク）に伝播されます。

これはすべてのユーザイベントで有効です。しかし、内部イベントの場合はイベントを再度発行する必要があるかもしれません。例えば、**終了**イベントが孫タスクで実行された場合、孫タスクが終了した時点で完了することになります。従って、次のタスク（子タスク）に対しても**終了**イベントを処理させたい場合、孫タスク内で別の**終了**イベントを発行させる必要があります。

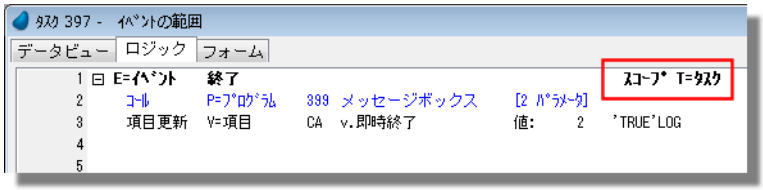
ユーザハンドラによって処理された内部イベントを Magic 機能でも有効にさせるには

内部イベントによって実行されるロジックユニットを作成する場合、そのイベントを伝播させるかどうかを指定することができます。

この例では、*Esc* キーを押下するか、  をクリックしてタスクを終了させた場合、終了イベントが実行されます。

- 伝播特性が *Yes* に設定されている場合、警告メッセージが表示され、終了イベントは Magic xpa に渡されタスクが終了します。
- 伝播特性が *No* に設定された場合、メッセージは表示されますが、イベントは Magic xpa に渡されず、タスクはオープンされたままの状態になります。

イベントが定義されたタスクでのみ有効にするには

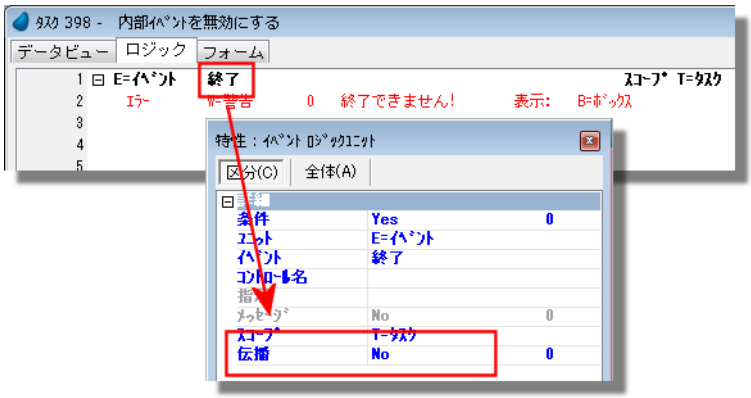


イベントを特定のタスク内でのみ有効にしたい場合があります。例えば、ユーザがある重要なタスクを終了する際に、警告メッセージを表示させる処理を考えてみます。しかし、そのタスクにはそれ程重要でない子タスクが定義されているかもしれません。このような子タスクに対しては、警告メッセージを表示させないようにします。

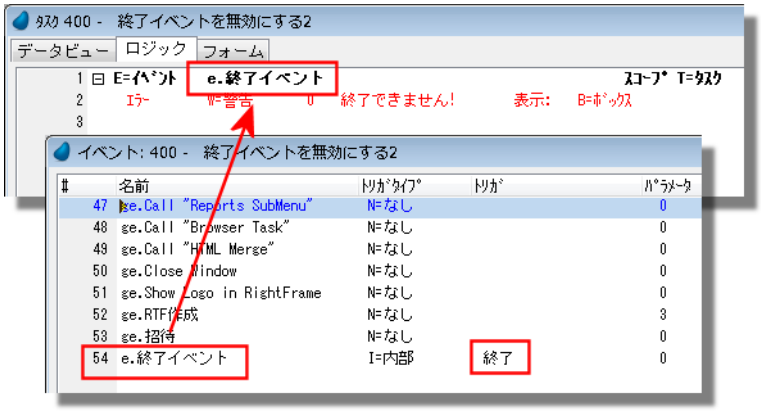
この例において、ユーザが ***Esc*** を押下した場合、「確認メッセージ」が表示されます。ユーザがはいを**クリック**した場合、項目 v. 即時有効がタスクを終了させるために更新されます。オリジナルの**終了**イベントは伝播されないため、タスクは終了しません。

イベントがこのタスク内でのみ処理されるようにするため、ハンドラの**スコープ**特性を ***T= タスク***に設定します。

内部イベントを Magic の標準機能で処理しないようにするには



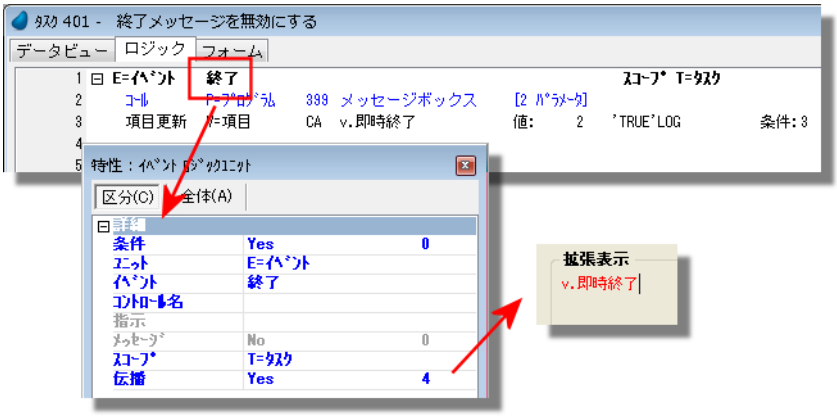
伝播特性を ***No*** に設定することによって内部イベントを処理しないようにすることができます。この例では、**終了**イベントがブロックされているため、ユーザが ***Esc*** を押下したり  を**クリック**してもタスクは終了しません。(開発環境でテストする場合は、  を**クリック**するか**デバッグ→停止**を選択することで終了させることができます)。



内部イベントがユーザイベントのトリガとして使用されている場合もブロックされる場合があります。これによって全体的に終了処理がブロックされ、別のタスクのように（何もなかったように）動作します。

イベントの処理を特定の条件でのみ機能するようにしたい場合は、**伝播**特性を式で指定します。詳細は次の例で明します。

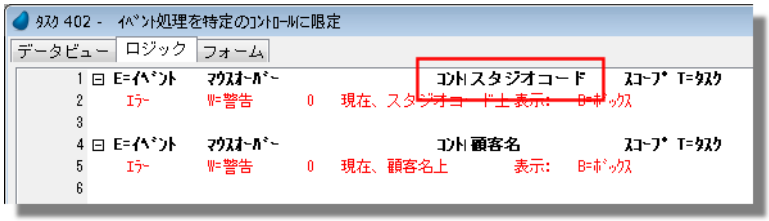
条件付きの伝播特性を使用する



伝播特性を式で指定することもできます。

この例では、**終了**イベントが発生したら「確認メッセージ」を表示させるようにしています。ユーザが**はい**を**クリック**したら、変数項目 v.即時終了を 'TRUE'LOG に更新します。この変数は、**伝播**特性に使用されています。このため、**はい**が**クリック**されると、**終了**イベントが Magic xpa に渡されタスクが終了するようになります。

イベント処理を特定のコントロールに限定するには



特定のコントロールでイベントが発生した場合にのみ処理したい場合があります。このような場合、コントロールを選択するには以下のようにします。

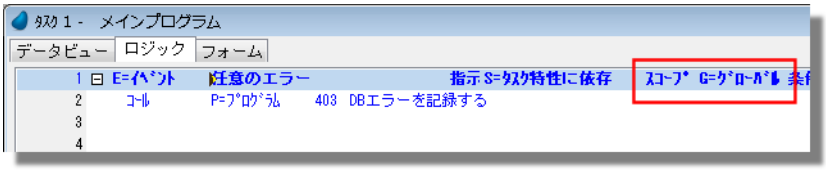
1. **イベント**ロジックユニットの**コントロール**カラムに移動します。
2. ここから**ズーム**して**コントロール一覧**を開きます。
3. 使用したいコントロール名を選択します。

コントロール名を直接入力して指定することもできます。イベントハンドラが親タスクか**メインプログラム**内に定義されている場合は、一覧から選択できません。このような場合は直接入力します。またこのようにすることで、コントロールがまだ定義されていなくても指定できます。

ヒント : *複数のコントロールに対して動作する汎用的なイベントを作成したい場合、**ロジックユニット**に**コントロール名**を割り当てることで対応できます。*

*例えば、「注文」（通常は注文番号が格納されています）と名付けられたコントロールに対する複数のイベントロジックユニットを定義することができます。注文コントロールで **F1** を押下すると入力された注文情報に対するヘルプ画面を表示させ、**F2** を押下するとその注文の現在の状態を表示させ、**Ctrl+P** を押下すると注文内容が印刷させるようにできます。しかし、コントロール名として「顧客」と定義されている別のコントロールに対しては、同じショートカットキーで異なるプログラムを起動させることができます。*

アプリケーション全体で利用できるハンドラを定義するには



アプリケーション全体で有効になるハンドラを作成したい場合、以下のようにします。

1. メインプログラムでイベントロジックユニットを作成します。
2. スコープを *G= グローバル*に設定します。

この例では、任意のエラーイベントを受け取ってデータベースエラーを記録するプログラムを呼び出しています。

スコープが *G= グローバル*に設定されている場合、このアプリケーションをコンポーネントとして定義しているホストアプリケーションによって発生した DB エラーも処理することができます。*S= サブツリー*に設定された場合、実行中のアプリケーション内で発生した DB エラーのみ処理されます。

コンポーネント内に定義されたハンドラを使用して、ホストアプリケーションでイベントを処理するには

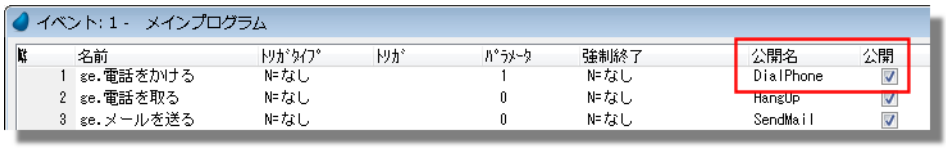
コンポーネント内で定義されたイベントを、ホストアプリケーションで実行させることができます。例えば、指定された電話番号で電話するイベントをコンポーネントで定義し、ユーザが電話番号フィールドでズームした場合、ホストアプリケーションでそのイベントを実行させるように定義できます。

このような処理を実現するには、以下の３つの手順が必要です。

1. コンポーネント内でイベントを定義します。
2. コンポーネント内でこのイベントに対するロジックユニットを定義します。
3. ホストプロジェクトでこのイベントを実行するように定義します。

これらの 3 つの手順の詳細を以下で説明します。

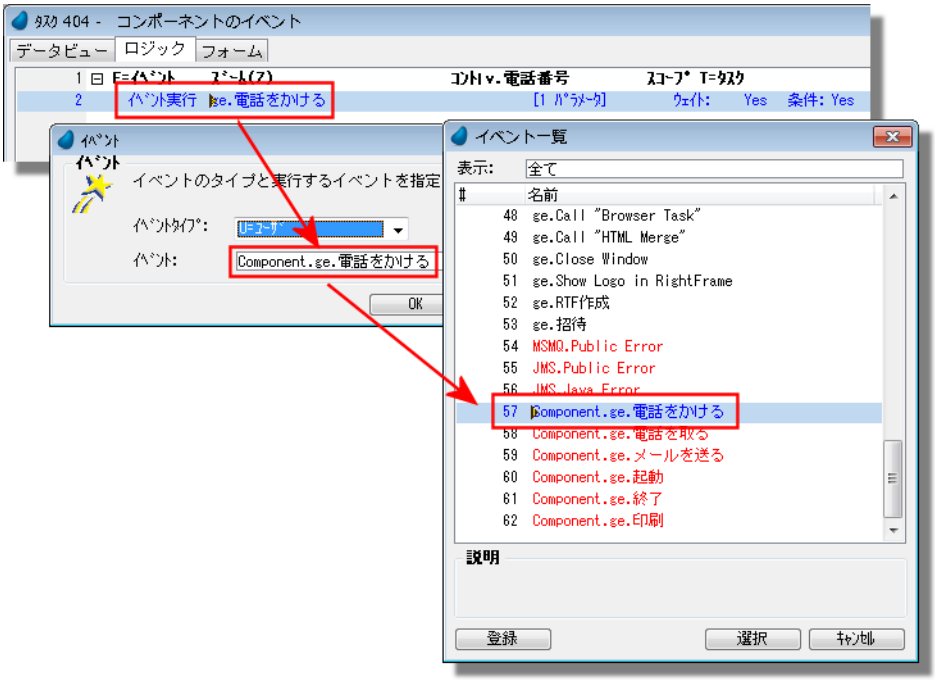
コンポーネント内でイベントを定義する



1. コンポーネントのメインプログラムでイベントを定義します。**公開**カラムがチェックされて、**公開名**カラムに名前が定義されていることを確認します。
2. インタフェースファイル（*.eci*）を作成する際に、ここで定義されたイベントを選択します。

これでコンポーネントを使用した場合、このイベントを選択することができます。

ホストアプリケーションでイベントを発行する



1. ホストアプリケーションのプログラムを開き、イベントを発行させたい場所に移動します。
2. 発行させるユーザイベントを選択します。コンポーネントイベントは、ホストアプリケーションで定義されたユーザイベントの下に赤色で表示されます。

コンポーネント内でイベントを処理する

1. コンポーネントのプロジェクトを開きます。
2. メインプログラムに移動します。
3. 定義されたイベントに対するイベントロジックユニットを作成します。**スコープ**を *G= グローバル*に設定します。

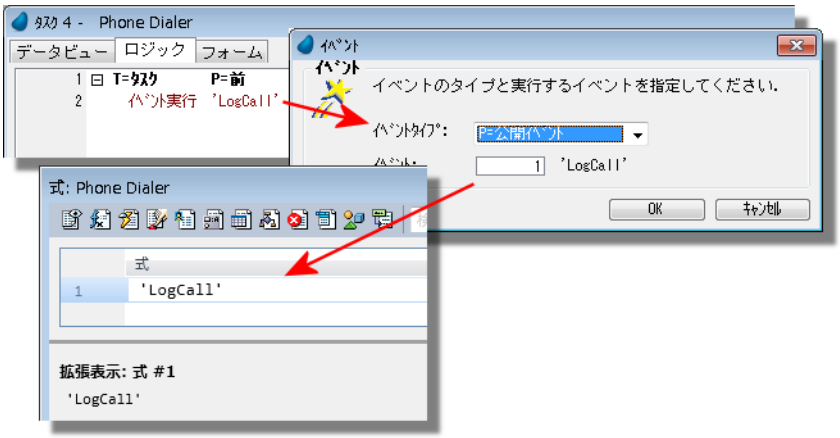
これで、ホストプログラムがイベントを発行すると、このロジックユニットが実行されます。

コンポーネントからホストアプリケーションで定義されたイベント実行するには

ホストアプリケーションで処理が可能なコンポーネント内でイベントを発行することができれば便利な場合があります。例えば、エラーが発生したら、パッケージ化されたモジュールによってイベントが発行されるようにすることが一般的に考えられます。

この例では、LogCall という名前のイベントをコンポーネント内で発行しています。このコンポーネントは、呼び出す際にパラメータを渡し、必要であればホストアプリケーションの情報を格納できるようになっています。

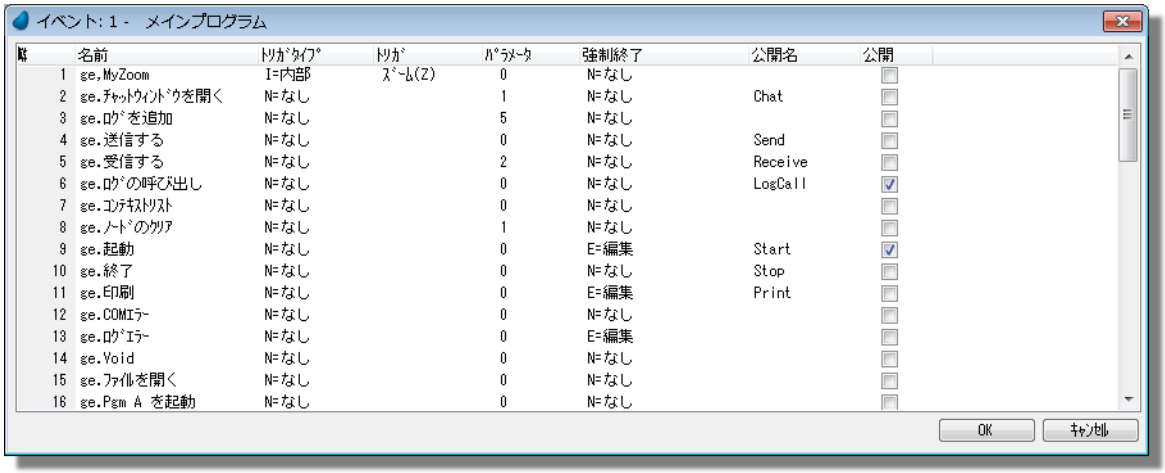
ホストイベントを発行する



1. 必要な場所にイベント実行処理コマンドを定義します。イベントダイアログが表示されます。
2. イベントタイプで *P= 公開イベント*を選択します。
3. イベントに移動します。ここからズームして式エディタを開きます。発行するイベント名を正確に指定します（イベント名は、シングルクォーテーションで囲みます）。

これにより、ホストアプリケーションによって処理できるイベントをコンポーネントが発行するようになります。次に、イベントを受け取るためのホストアプリケーションを設定する必要があります。

ホストアプリケーションでイベントを処理する



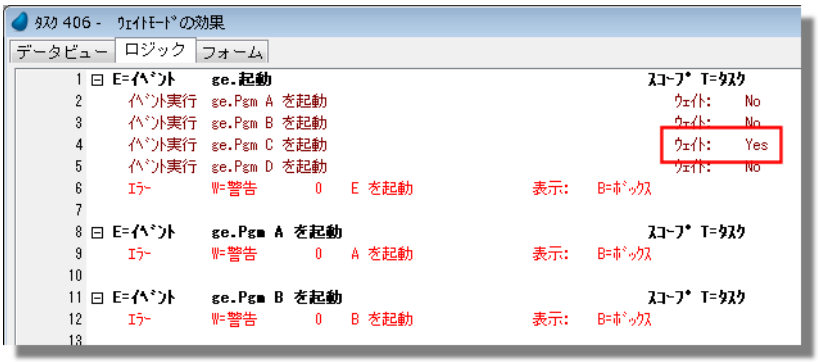
	名前	トリガータイプ	トリガー	パラメータ	強制終了	公開名	公開
1	ge.MyZoom	I=内部	ズーム(Z)	0	N=なし		
2	ge.チャットウィンドウを開く	N=なし		1	N=なし	Chat	
3	ge.ログを追加	N=なし		5	N=なし		
4	ge.送信する	N=なし		0	N=なし	Send	
5	ge.受信する	N=なし		2	N=なし	Receive	
6	ge.ログの呼び出し	N=なし		0	N=なし	LogCall	☑
7	ge.コンテキストリスト	N=なし		0	N=なし		
8	ge.ノードのクリア	N=なし		1	N=なし		
9	ge.起動	N=なし		0	E=編集	Start	☑
10	ge.終了	N=なし		0	N=なし	Stop	
11	ge.印刷	N=なし		0	E=編集	Print	
12	ge.COMエラー	N=なし		0	N=なし		
13	ge.ログエラー	N=なし		0	E=編集		
14	ge.Void	N=なし		0	N=なし		
15	ge.ファイルを開く	N=なし		0	N=なし		
16	ge.Pgm A を起動	N=なし		0	N=なし		

1. メインプログラム内でイベントを作成します。**公開名**カラムには、コンポーネントで定義されたイベント名と同じ名前を指定します。
2. **公開**カラムのチェックボックスをチェックします。

これで、必要に応じてイベントを処理するロジックユニットを作成することができます。

イベントを即時に処理させるには

イベントを即時に処理するようにしたい場合、**ウェイト**特性を ***Yes*** に設定します。この場合、他のコマンドは処理されず直ちにイベントが処理されます。これは処理コマンドの定義順に処理されるため、同期処理と呼ばれます。



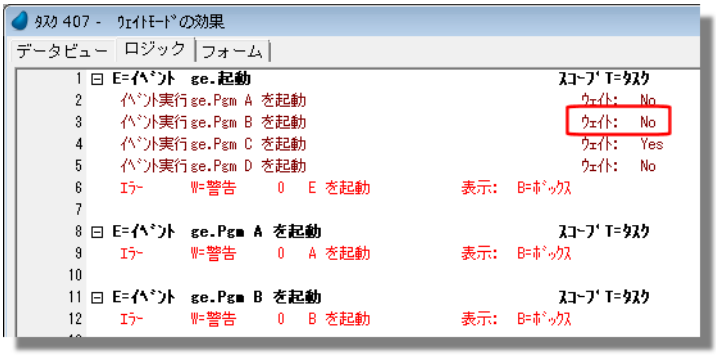
例えば、上記のように 4 つの**イベント実行**処理コマンドが定義されている場合、以下のように動作します。

1. イベント A は、**ウェイト** =***No*** で発行されます。この場合、イベント A はキューに入りますが、即時には実行されません。次の処理コマンドが処理されます。
2. イベント B は、**ウェイト** =***No*** で発行されます。この場合、イベント B はキューに入りますが、即時には実行されません。次の処理コマンドが処理されます。
3. イベント C は、**ウェイト** =***Yes*** で発行されます。この場合、次のコマンドが処理されず、イベント C が直ちに実行されます。
4. イベント D は、**ウェイト** =***No*** で発行されます。この場合、イベント D はキューに入りますが、即時には実行されません。次の処理コマンドが処理されます。
5. 次のコマンドは、E を表示する**エラー**処理コマンドです。これは即実行されます。
6. キューに入ったイベント A、B および D が実行されます。

これにより、メッセージボックスが以下の順番で表示されます。C、E、A、B、D

イベントの処理を延期させるには

イベントを後で処理させたい場合、**ウェイト**特性を *No* に設定します。この場合、他のコマンドが処理された後でイベントが実行されます。イベントは割り込みを許すため、処理コマンドの定義順に実行されない場合があります。これを非同期処理と呼びます。



例えば、上記のように 4 つの［イベント実行］処理コマンドが定義されている場合、以下のように動作します。

1. イベント A は、**ウェイト** =*No* で発行されます。この場合、イベント A はキューに入りますが、即時には実行されません。次の処理コマンドが処理されます。
2. イベント B は、**ウェイト** =*No* で発行されます。この場合、イベント B はキューに入りますが、即時には実行されません。次の処理コマンドが処理されます。
3. イベント C は、**ウェイト** =*Yes* で発行されます。この場合、次のコマンドが処理されず、イベント C が直ちに実行されます。
4. イベント D は、**ウェイト** =*No* で発行されます。この場合、イベント D はキューに入りますが、即時には実行されません。次の処理コマンドが処理されます。
5. 次のコマンドは、E を表示するエラー処理コマンドです。これは即実行されます。
6. キューに入ったイベント A、B および D が実行されます。

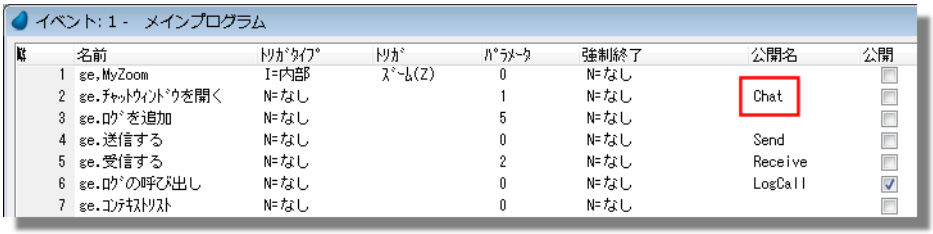
これにより、メッセージボックスが以下の順番で表示されます。C、E、A、B、D

イベント名を動的に指定してイベントを発行するには

イベント名を式で動的に指定することもできます。例えば、コンポーネントからホストアプリケーションに定義されているイベントを発行したい場合、イベント名が直接参照できないため選択することができません。また、ユーザが入力したり、テーブルに格納されたイベント名を選択することでそのイベント名に応じて発行するイベントを切り替えたりする方法も必要です。

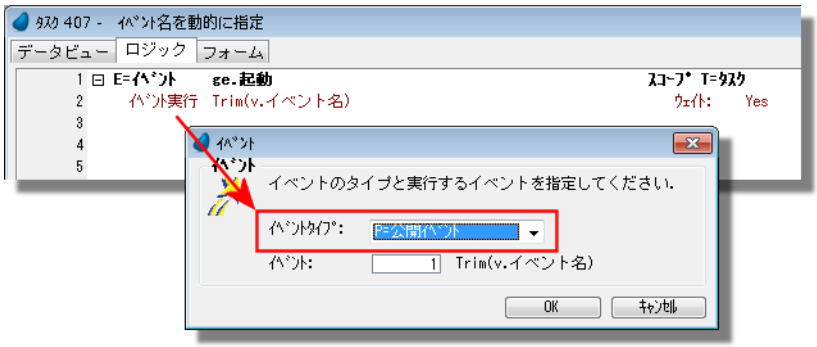
この例では、コンボボックスからイベントを選択することで、選択されたイベントが発行されるようにしています。

メインプログラム内でイベントを設定する



最初にメインプログラム内でイベントを設定し、それに公開名を定義する必要があります。この例では、以下のイベントが公開定義されています。Chat, Send, Receive, LogCall

イベントを公開名で発行する



次に、イベントを発行する処理を定義します。

1. イベントを発行させたい場所に移動します。
2. *R* を入力し、イベント実行処理コマンドを作成します。イベントダイアログが表示されます。
3. イベントタイプで *P= 公開イベント* を選択します。
4. イベントに移動します。ここからズームして式エディタを開きます。発行するイベント名が返る式を定義します。（イベント名は、大文字小文字を区別しません。また空白は入れないでください。）

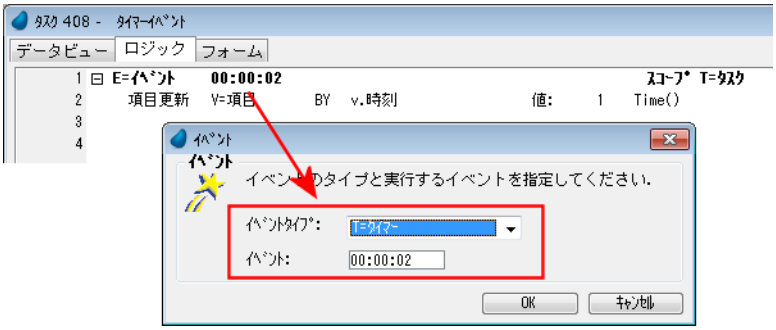
この例では、コンボボックスからイベント名を選択しています。イベント名は空白が削除されて発行されます。

一定時間の経過後に処理を実行させるは

一定の時間が経過後に処理が実行されるようにすることができます。例えば、数分ごとにメールのキューを確認するような処理などが考えられます。

Magic xpa では、このような場合タイマーイベントを使用して実現できます。この例では、２秒ごとに画面上の時刻表示を更新するようにしています。

タイマーイベントを使用する



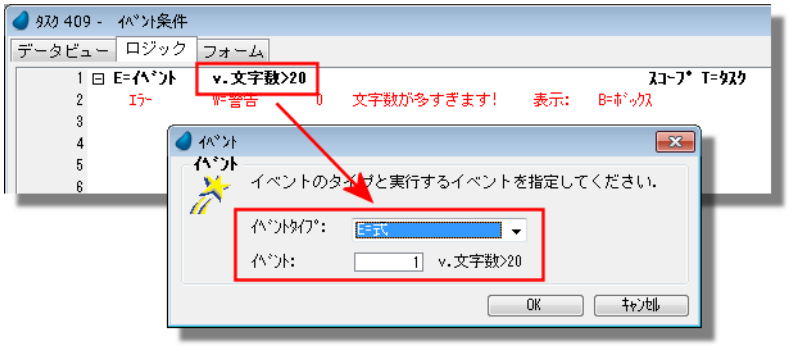
1. イベントを発行させたい場所に移動します。
2. ***R*** を入力し、イベント実行処理コマンドを作成します。イベントダイアログが表示されます。
3. **イベントタイプ**で ***T= タイマー***を選択します。
4. **イベント**に移動します。イベントを発行する時間間隔を入力します。書式は、HH：MM：SS です。この例では、00:00:02 と入力されているため、イベントは 2 秒ごとに発行されます。

これで、指定された時間間隔に従ってイベントが発行されます。

条件が満たされた場合に処理が実行されるようにするには

特定の条件にもとづいて処理が実行されるようにイベントを発行することができます。例えば、注文合計が大き過ぎたり電子メールが到着した場合にメッセージを表示させる処理が考えられます。これらのイベントは、式イベントとして定義できます。

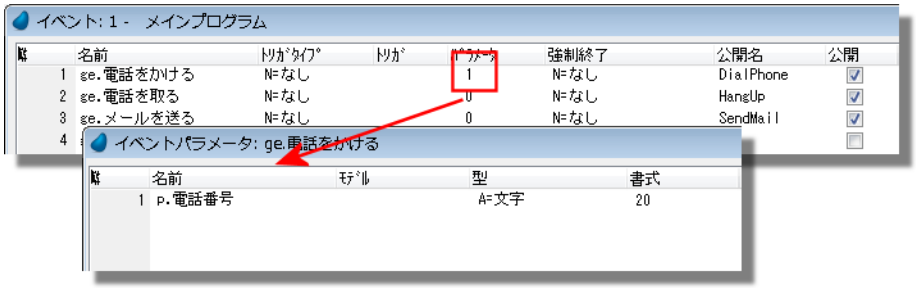
式イベントを作成する



1. イベントを発行させたい場所に移動します。
2. *R* を入力し、イベント実行処理コマンドを作成します。イベントダイアログが表示されます。
3. イベントタイプ欄で式を選択します。
4. イベント欄に移動します。ここからズームして式エディタを開きます。イベントを発行する条件を定義します。式が'TRUE'LOG と評価された場合にイベントは発行されます。この例では、ユーザが 20 桁を越える文字が入力された場合にイベントが発行されます。

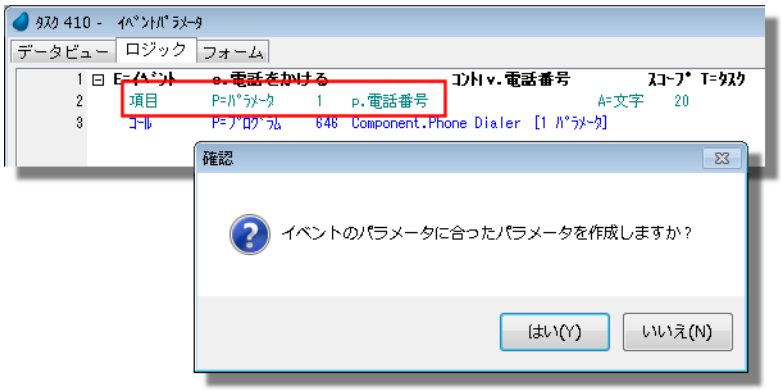
イベント発行時に情報を渡すには

パラメータ付きのイベントを作成する



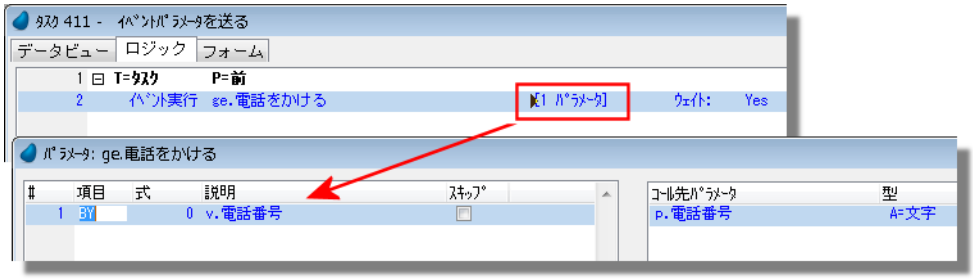
1. イベントテーブルでイベントを作成します。
2. パラメータカラムに移動します。ここからズームしてイベントパラメータテーブルを開きます。
3. イベントハンドラに渡したいパラメータを定義します。

イベントハンドラでパラメータを受け取るようにする



1. イベントを発行させたい場所に移動します。
2. *R* を入力し、イベント実行処理コマンドを作成します。イベントダイアログが表示されます。
3. イベントタイプで *U= ユーザ* を選択します。
4. イベントに移動します。ここからズームしてイベントを選択します。選択されたイベントにパラメータが定義されている場合、「イベントのパラメータに合ったパラメータを作成しますか？」というメッセージが表示されます。はいをクリックするとパラメータが自動的に追加されます。

イベントを発行する

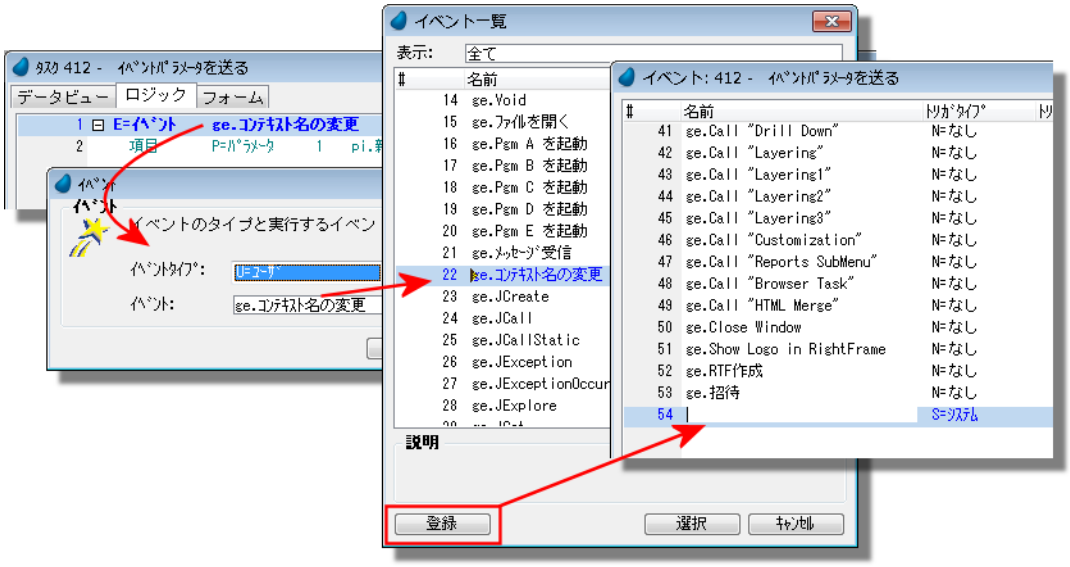


これで、イベント発行すると、コール処理コマンドと同じようにパラメータを渡すことができます。

ユーザイベント選択時にイベントを登録するには

ユーザイベントの選択時に該当するイベントが見つからず、登録する必要な場合があります。このような場合、イベント一覧からイベントテーブルを開くことで、イベントを追加することができます。

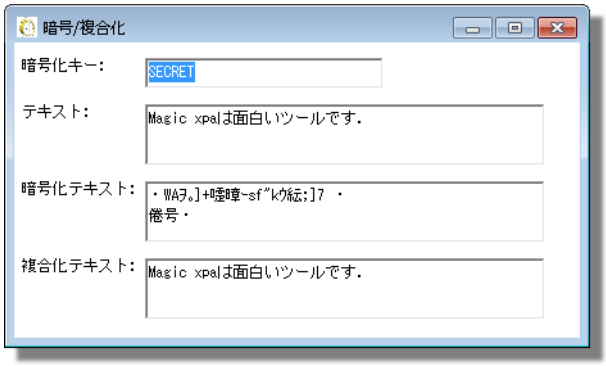
ユーザイベントを追加する



1. イベントロジックユニットのイベント選択カラムに移動します。
2. ズームしてイベントダイアログを開きます。
3. イベントタイプで *U= ユーザ*を選択します。
4. イベントからズームして、イベント一覧を開きます。
5. 登録ボタンをクリックするとイベントテーブルが表示されます。その際、1 行追加された状態で開くためここに追加したいイベントを入力します。
6. OK をクリックするとイベント一覧に追加されたイベントが表示され、位置付けされます。
7. 選択をクリックすると追加されたイベントが設定されます。

第 30 章 : セキュリティ

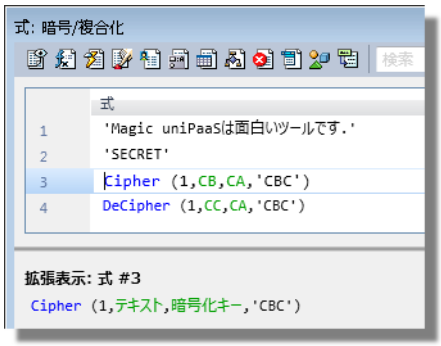
データの暗号化 / 復号化を行うには



Cipher() と **DeCipher()** 関数を使用することで、BLOB 型項目に格納されたデータの暗号化 / 復号化を行うことができます。これらの関数は、汎用的な暗号化アルゴリズムをサポートしているため、Magic xpa 以外のアプリケーションで作成されたデータを解読することもできます。サポートされるアルゴリズムには、Blowfish のような対称型のものや RSA などの非対称なものが含まれています。

データの暗号化 / 復号化を行うために、対称型のアルゴリズムは同じキーを使用します。非対称型のアルゴリズムの場合は、対のキーと暗号化 / 復号化対象のデータが必要です。

Cipher() 関数を使用する



Cipher() 関数の構文は以下の通りです。

Cipher(*Cipher ID*, *Buffer*, *Key [, Mode, IV]***)**

パラメータ :

- *Cipher ID* …… 暗号化アルゴリズム ID を表す数値。上記の例の場合、Blowfish を表す 1 が指定されています (「サポートされる暗号化モード」 (649 ページ) を参照してください)。
- *Buffer* …… 暗号化するデータを含む BLOB 型項目
- *Key* …… キーを含む BLOB 型項目。必要なキー長は、指定されたアルゴリズムに依存します。この例では、SECRET という文字列をキーとして指定しています。
- *Mode* …… どのモードを使用するかを指定するオプションパラメータです。指定可能なモードは暗号化アルゴリズムに依存します。
- *IV* …… 初期ベクトルを含む BLOB 型項目です。このパラメータはオプションです。

この関数は、暗号化されたデータを含む BLOB データを返します。

Decipher() 関数を使用する

Decipher() 関数の構文は、**Cipher()** と同じです。

Decipher(*Cipher ID*, *Buffer*, *Key [, Mode, IV]***)**

パラメータ：

- *Cipher ID* …… 暗号化アルゴリズム ID を表す数値。上記の例の場合、Blowfish を表す 1 が指定されています（「サポートされる暗号化モード」（649 ページ）を参照してください）。
- *Buffer* …… 復号化するデータを含む BLOB 型項目
- *Key* …… キーを含む BLOB 型項目。必要なキー長は、指定されたアルゴリズムに依存します。この例では、SECRET という文字列をキーとして指定しています。
- *Mode* …… どのモードを使用するかを指定するオプションパラメータです。指定可能なモードは暗号化アルゴリズムに依存します。
- *IV* …… 初期ベクトルを含む BLOB 型項目です。このパラメータはオプションです。

この関数は、復号化されたデータを含む BLOB データを返します。

サポートされる暗号化モード

アルゴリズム名	暗号化コード	サポートされるモードと IV の長さ	キー数（バイト）	対称／非対称
BLOWFISH	1	ECB - NA CBC - 8 CFB - 8 OFB - 8	サポート：1 ～ 56 推奨値：16	対称
CAST	2	ECB - NA CBC - 8 CFB - 8 OFB - 8	サポート：5 ～ 16 推奨値：8	対称
DES	3	ECB - NA CBC - 8 CFB - 8 OFB - 8	サポート：8 推奨値：8	対称
RC2	5	ECB - NA CBC - 8 CFB - 8 OFB - 8	サポート：5 ～ 16 推奨値：8	対称
RC4	6	未対応	サポート：1 ～ NR 推奨値：16	対称
RC5	7	ECB - NA CBC - 8 CFB - 8 OFB - 8	サポート：1 ～ 255 推奨値：16	対称
DES3	8	ECB3 - NA CBC3 - 8	サポート：16、24 推奨値：24	非対称
RSA	9	未対応	サポート：48 ～ 2048 推奨値：128	非対称
AES	10	ECB - NA CBC - 8 CFB - 8 OFB - 8	サポート：16、24、32 推奨値：16	対称

テーブルのデータを暗号化するには

パスワードを使用してアプリケーションへのアクセスを制限した場合でも、低レベルのツールを使用して直接ディスク内のデータを参照することが可能です。このような理由により、非常に重要な情報を含んだテーブルを暗号化する必要性があります。

Magic xpa では、このようなことを容易に実現することができます。以下の手順で行います。

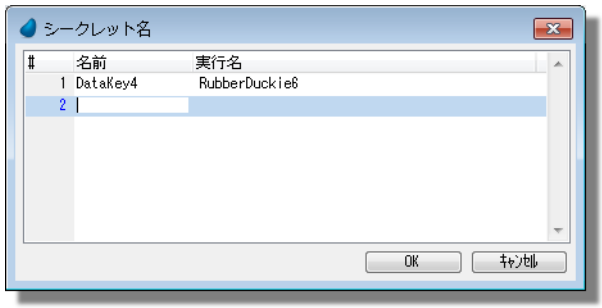
データベーステーブルを暗号化する

1. データリポジトリを開き、暗号化したいデータソースに移動します。
2. **データソース特性**（*Alt+Enter*、または**編集→特性**）を開きます。
3. **アクセスキー**特性に任意の文字列を入力します。**シークレット名**を使用することを推奨します（「データベースのログイン情報を非表示にするには」（651 ページ）を参照してください）。
4. **テーブルの暗号化**特性を *Yes* に設定します。

これで、テーブル内にデータは暗号化されます。同じキーを持っているユーザのみがこのデータを参照することができます。

注： すべての DBMS が暗号化をサポートしているわけではありません。サポートされていない場合、**アクセスキー**特性や**テーブルの暗号化**特性が無効表示になります。

データベースのログイン情報を非表示にするには

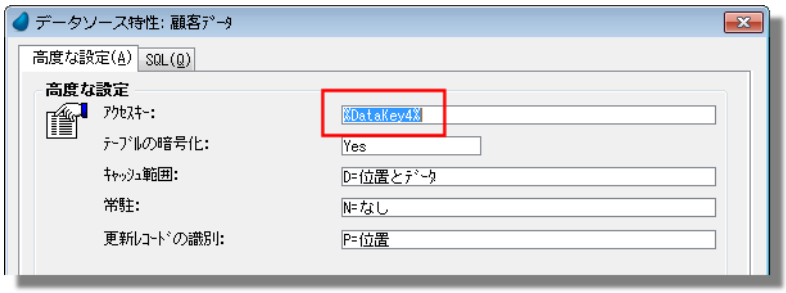


シークレット名を設定する

1. SUPERVISOR で Magic xpa にログオンします。
2. シークレット名テーブル（オプション→設定→シークレット名）を開きます。
3. シークレット名を入力します。名前カラムの内容は、SUPERVISOR でしか参照できない点を除いて、論理名と同じように使用できます。

シークレット名は、セキュリティファイル内で暗号化された状態でユーザ ID とパスワード一緒に格納されます。

シークレット名を使用する



アクセスキーにシークレット名を使用する場合は、論理名と同じ方法で指定します。この名前は、実行時に実際の名前に変換されます。シークレット名は、プロジェクト特性のアプリケーションアクセスキー特性やサーバ／データベース特性のパスワード特性、およびデータソース特性のアクセスキー特性などの特定の特性値でしか使用できません。

シークレット名を使用することで、アプリケーション毎に独自のセキュリティ情報が定義できることになります。開発者は、論理名を使用して値をコード化できますが、実行時にどのような値に変換されるかは分かりません。同じように、例えば、運用時のデータベースのパスワードを開発者が知る必要がなくなります。

アプリケーションの管理者権利を定義するには

Magic アプリケーションを開発する場合、SUPERVISOR と呼ばれるユーザは特別な権利を持っています。SUPERVISOR としてログインすると、他のユーザ用にアカウントを作成したり、修正することができます。

最初に Magic アプリケーションを作成する場合、SUPERVISOR はすでにユーザリスト内に定義された状態になっています。SUPERVISOR としてログインするには、以下の操作が必要です。

1. プロジェクトがオープンされている場合、**ファイル→プロジェクトを閉じる**を選択します。
2. **オプション→ログオン**を選択します。
3. **ログオン**ダイアログが表示されます。
4. **ユーザ ID** に、***SUPERVISOR*** を入力します。
5. **パスワード**を空白のままにします。
6. **OK** を押下します。

これで、SUPERVISOR としてログインされ、ユーザ ID の保守が可能になります。SUPERVISOR ユーザにパスワードをすることを推奨します。

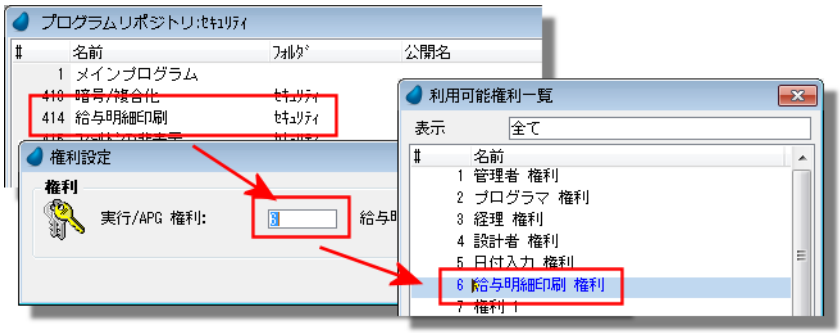
セキュリティファイルを削除する

セキュリティファイルが削除されたり、指定された場所に存在しない場合は、パスワードを空白にした状態で SUPERVISOR としてログインすることができます。SUPERVISOR としてログインすることでアプリケーションに対するユーザ ID やグループや権利などへのアクセスが可能になります。

このようにデフォルトの SUPERVISOR でアクセスできないようにするには、以下のオプションを使用してアプリケーション毎のセキュリティ設定を行う必要があります。

- 権利の非公開
- **アプリケーション特性**の**スーパー権利キー**
- **シークレット名**

特定プログラムの実行を制限するには



権利のないユーザによって実行されないようにプログラムに権利を設定することができます。

例えば、給与明細を印刷するプログラムを考えてみます。ほとんどのユーザはこのプログラムを実行することはできません。メニューに権利を設定することで表示を制限することができますが、開発者レベルでの実行を制限できるようにしたい場合があります。このような場合は、以下のようにプログラムに実行権を設定することで対応できます。

1. 実行を制限させたいプログラムに移動します。
2. **オプション→権利**を選択します。**権利設定**ダイアログが表示されます。
3. **実行／APG 権利**で**ズーム**して、設定したい権利名を選択します。

これで、この権利を持っていないユーザはプログラムを実行することができなくなります。

この方法でプログラムの実行を制御できますが、メニューに対しても同じように権利を設定しておくことで、より確実に制限することができます。メニューに対する権利設定については、「ログオンユーザにもとづいてメニューをカスタマイズするには」（654 ページ）を参照してください。

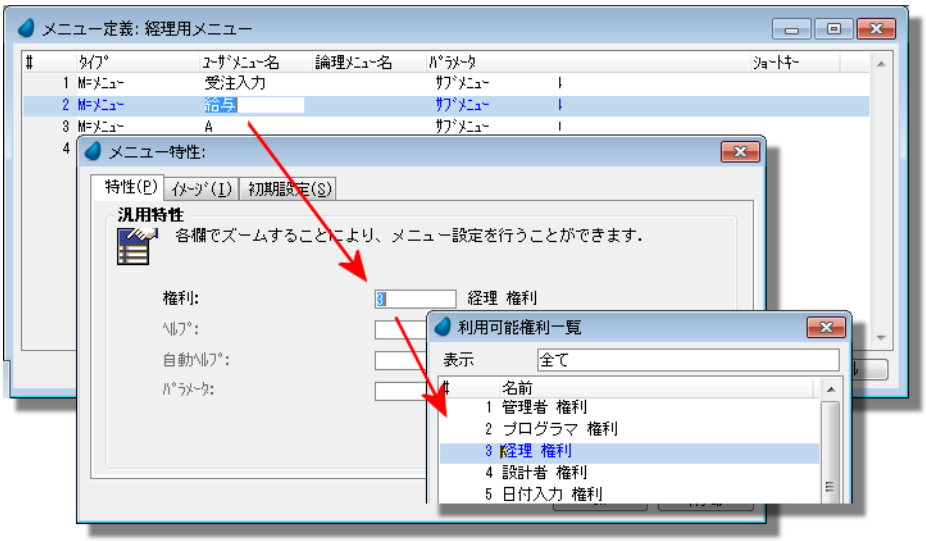
ヒント : *開発者が、このプログラムをテストするために実行する必要がある場合は、評価用と開発用の異なる２つのセキュリティファイルを用意してください。その際開発者は、評価用のデータで実行するようにしてください。*

ログオンユーザにもとづいてメニューをカスタマイズするには

ユーザが使用するメニューのみを表示するようにした方が、使いやすく見た目もすっきりしたものにります。また、セキュリティの面から考えて、すべての機能を表示させないようにした方が安全です。

Magic xpa では、メニュー表示を任意にカスタマイズすることができます。詳細は、第 26 章：「実行時にメニューの表示／非表示を行うには」（598 ページ）を参照してください。しかし、プログラムでメニュー表示を制御する方法より、セキュリティ機能を利用した方が簡単な場合があります。

メニュー項目に権利を設定する



1. メニューリポジトリ（*Shift+F6*）を開きます。
2. 権利を設定したいメニュー項目に移動します。一般的に、職務権限にもとづいて利用可能な最も高い権利レベルをメニューに割り当てるようにしますが、どのメニュー項目に対しても権利を割り当てることができます。
3. *Alt+Enter* を押下して、メニュー特性を開きます。
4. カーソルを特性タブの、**権利**特性に置きます。ここからズームして権利を選択します。

これで、権利を持たないユーザがアプリケーションを実行した場合、メニューが表示されなくなります。この例では、経理の権限を持たないユーザは給与支払台帳のメニューが表示されなくなります。

注： 上記の操作を行うには、予め開発者が権利にアクセスできるようにしておく必要があります。

ログオンユーザにもとづいて機能を制限するには

Rights() 関数を使用することで、ユーザの権利レベルに応じて実行できる機能をプログラム内で変更することができます。ここでは、関数の定義方法や使用例を説明します。

Rights() 関数を使用する

Rights() 関数の構文は以下の通りです。

Rights(*right*)

パラメータ：

- *right* ……権利リポジトリに定義されている権利名

プログラムを実行しているユーザが指定された権利を持っている場合、'TRUE'LOG が返ります。例えば、

Rights(*'Accounting Right'RIGHT*)

は、'Accounting Right' という権利を持っているユーザが実行すると 'TRUE'LOG が返ります。

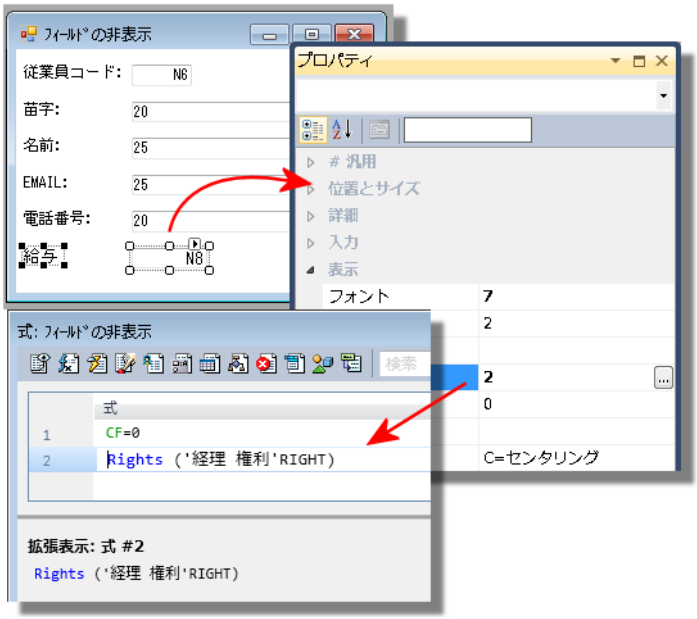
以下のような操作を行うことで簡単に入力することができます。

1. 式テーブル（*Ctrl+E*）を開きます。
2. *F4* を押下して 1 行追加します。
3. ri を入力します。オートコンプリート機能が働いて関数リストが表示されます。ここから **Rights** を選択します。
4. これにより式には以下のように入力されます。
   Rights(
5. 右上のコンボボックスから権利を選択します。権利一覧が表示され、アクセス可能な権利のリストが表示されます。設定したい権利（この例では、経理 権利）を選択します。これによって以下のように権利名が入力されます。
   Rights (*' 経理 権利 'RIGHT*
6. 次に、') ' を入力して括弧を閉じるだけで式の入力が完了します。

複数の権利を指定したり、他の論理型の条件式に追加することもできます。

次に指定された式によって、どのように動作するかを紹介します。

権利を持たないユーザに対してコントロールを非表示にする



セキュリティレベルにもとづいて実現させたい内容の 1 つとして、画面にデータを表示させないようにすることです。この例では、給与の表示を制御しようとしています。この方法は、プッシュボタン等を含めたすべてのコントロールで有効です。

1. 制御したいコントロールを選択します。複数のコントロールに対して同じように制御させたい場合は、*Ctrl* キーを押下しながら対称となるコントロールを 1 つずつクリックします。この例では給与に関するフィールド（ラベルとエディット）を対象にしています。
2. *Alt+Enter* を押下してコントロール特性を開きます。
3. **可視**特性に移動し、右側の**式**特性に移動します。

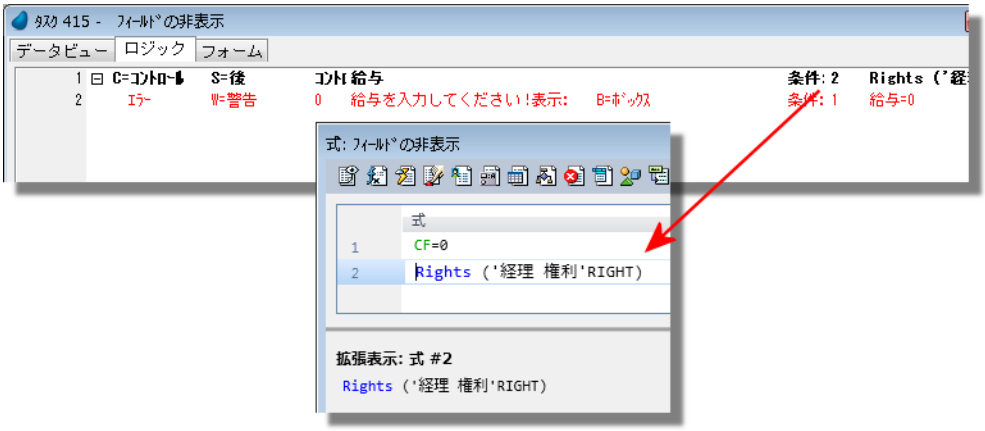
4. ここからズームして式エディタで **Rights()** 関数を使用した式を定義します。

これで、権利を持っていないユーザがこのプログラムを実行すると設定されたフィールドは表示されなくなります。

表示はさせるけれど、アクセスできないようにしたい場合は、**有効**特性に式を定義します。

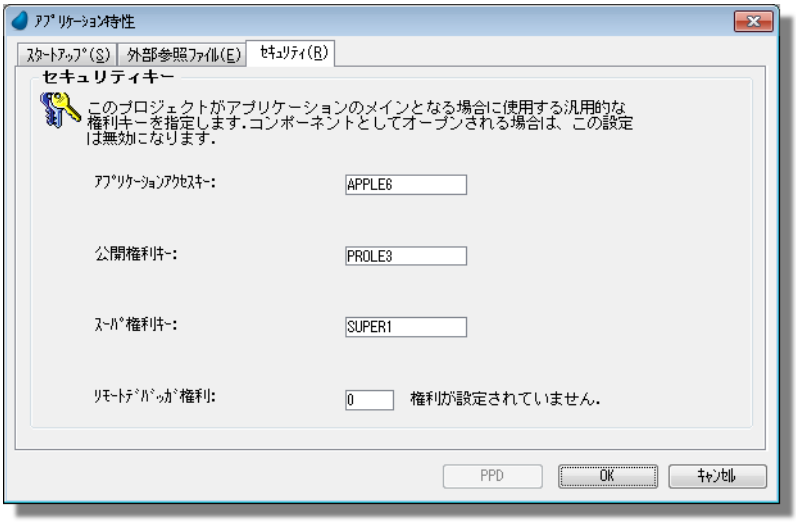
注： フィールドが非表示になった場合、そのフィールドに対する検証ロジックが実行されないことを確認する必要があります。例えば、ほとんどのユーザに対して表示されない給与フィールドがフォームに定義されているものとします。また、「給与を入力してください！」と言う確認を表示するエラー処理コマンドが定義されています。このような場合、ユーザがフィールドにデータを入力できない場合、必ずエラーメッセージが表示されることになります。従って、エラーを表示するロジックに、表示を制御させるために定義された条件式を同じように設定する必要があります。しかし、コントロールが無効の場合、このコントロールに対するコントロール検証ロジックユニットも無効になるため、問題にはなりません。

権利を持たないユーザに対してロジックの実行を防止する



Rights() 関数を使用してロジックの実行を制御するには、**条件**特性に式を設定します。この例では、**コントロール後**に条件を設定しています。このように、**処理コマンド**や**ロジックユニット**に条件式を設定することで無効にすることができます。

アプリケーション全体のアクセスを制限するには



アプリケーション特性（*Ctrl+Shift+P*）の**セキュリティ**タブにある特性値を使用することでアプリケーション全体に対する実行時のアクセス制限を行うことができます。ここには、セキュリティに関係する４つの特性があります。各特性の詳細は Magic xpa のリファレンスを参照してください。

特性	機能	効果
アプリケーションアクセスキー	アプリケーションへのアクセスを許可する	ユーザがこのキーを持っていない場合、アプリケーションのオープン時に以下のメッセージが表示されます。 「アクセスが拒否されました：実行エンジンは、アプリケーションのオープンに失敗しました」
公開権利キー	権利キーを持つユーザに対して管理者権利を許可する	このアクセスキーがない場合、実行環境ではアプリケーションにどういったキーが割当てられているか知ることはできません。つまり、［ユーザ ID］テーブルの［権利］カラムをズームしても、［権利名］カラムの内容を見ることができません。また、［キー］カラムで**ズーム**して［公開権利一覧］を表示させることもできません。
スーパー権利キー	アプリケーションの実行に対するすべての権利を許可します	これはすべての権利より優先されます。この権利を持つことですべてのプログラムを実行することができるので、テストを行う場合は便利です。
リモートデバッガ権利	リモートデバッガの使用を許可します。	リモートデバッガ起動時に指定されたユーザに、ここで指定されたアクセスキーがない場合は、接続できません。

リモートデバッガ権利は、**権利**リポジトリに**ズーム**して設定します。他のキーは権利キーを直接入力します。権利キーが入力されると、そのキーを持たないユーザからは参照することができなくなります。このため、入力されたキーは正しく管理しておいてください。

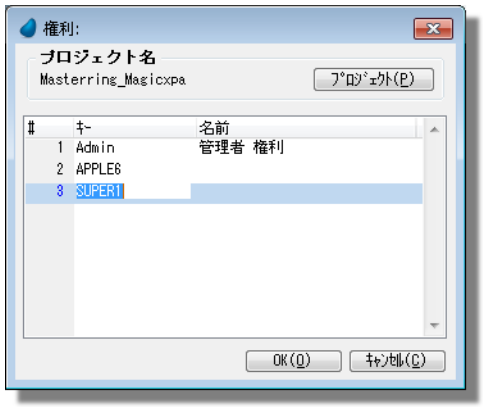
注： **開発上のアクセス制限を設定することはできません。また、このキーを持たない開発者であってもキーの内容を参照することができます。**

セキュリティキーを使用する

ユーザに対してセキュリティキーを割り当てるには、以下の手順で実行します。

1. プロジェクトが開いている場合は一旦閉じます。
2. SUPERVISOR でログオンします。
3. **ユーザ ID** テーブルを開き、権利を割り当てたいユーザにカーソルを置きます。
4. **権利**カラムから**ズーム**します。**権利一覧**が表示されます。
5. *F4* を押下して、１行追加します。
6. **アプリケーション特性**に入力されキー名と同じ名前を、**キー**カラムに入力します。

これで、ユーザはキーを持つことになります。



権利グループを定義するには

Magic xpa でセキュリティシステムを利用する場合、通常はユーザの職務権限にもとづいて権利を定義します。この職務権限を Magic xpa では**グループ**として扱っています。例えば、経理グループに所属するユーザは、開発グループのメンバーとは異なるメニューや画面を持つことになります。しかし、給料明細を印刷したり勤務表を確定するような権利は、グループではなく特定の個人に割り当てることになります。

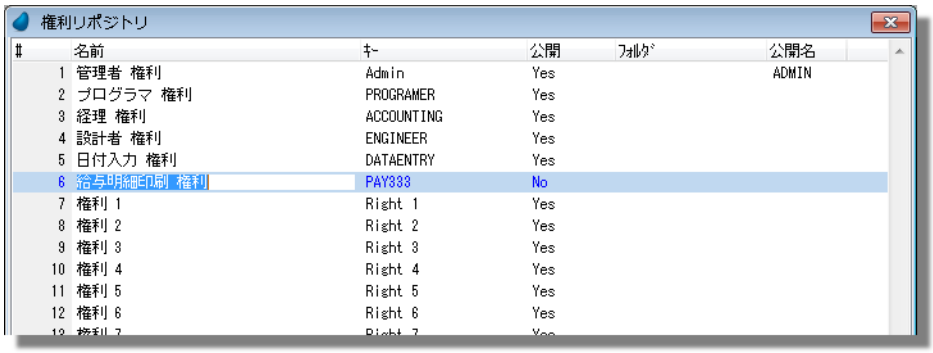
多くの権利を詳細に分割することはシステムを運用する上で扱いにくいものになりますが、抽象的すぎるのも柔軟性を失う可能性があります。システムの設計時には、このような点を考慮する必要があります。

ユーザに権利グループを設定するための手順は次の通りです。

1. **権利**を設定します。
2. **グループ**を設定します。
3. **ユーザ**を設定します。

以下、これらの手順の詳細について説明します。

1. 権利を設定する



権利は、各プロジェクトの**権利**リポジトリで設定されます。権利に**公開名**を設定することで、コンポーネントとして公開することができます。

権利を入力するには以下のようにします。

1. *F4* を押下して 1 行追加します。
2. **名前**カラムを入力します。任意の文字が入力できます。式や権利特性で権利を選択する場合に表示される**権利一覧**には、この名前が表示されます。
3. **キー**カラムを入力します。
4. 必要であれば**公開**カラムを *No* に設定します（この内容については、以下で説明します。）
5. 必要であれば、**公開名**カラムを入力します。

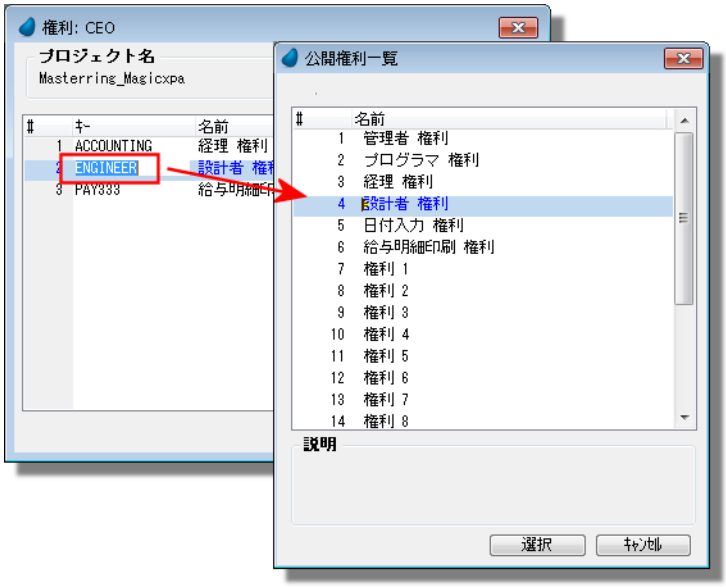
公開権利

公開カラムを *No* に設定した場合、この権利を持たないユーザがキー名を参照できなくなることになります。この例では、給料明細印刷の権利が非公開になっています。SUPERVISOR がキー名 PAY333 を知っている場合は、この権利を他のユーザに割り当てることができます。

セキュリティファイルが存在していなかったり、何も定義されていない場合、Magic xpa ではデフォルトの SUPERVISOR としてログインし、権利設定を行うことができるため、権利を公開するか否かの指定が必要になります。SUPERVISOR に対して公開したくない項目が存在する場合、公開カラムを No に設定してください。

しかし、ログイン時のユーザ ID とパスワードおよび、非公開のキー名を忘れた場合、誰もアクセスできなくなることになるので、注意してこの機能を使用してください。

2. グループを設定する



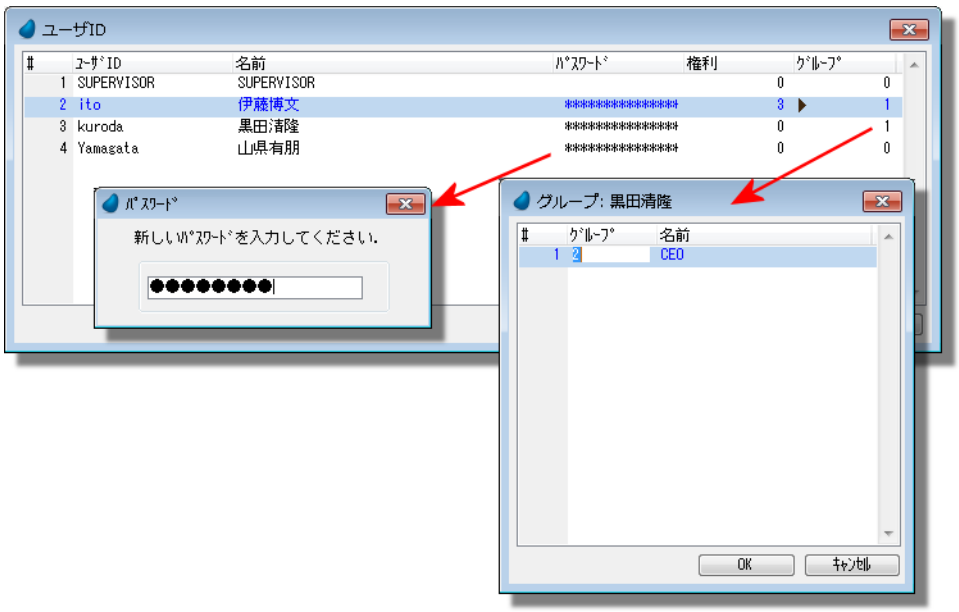
グループ情報は、プロジェクト内には定義されません。これらはセキュリティファイルに格納されます。このファイルは、複数の Magic アプリケーション間で共有することができます。**動作環境**ダイアログ（**オプション→設定→動作環境**）の**セキュリティファイル**でファイル名やパス名を確認することはできますが、暗号化されているため直接ファイルを編集することはできません。従って、グループを設定するには **Magic xpa Studio** が必要です。

必要条件： 最初に、SUPERVISOR としてログインする必要があります。詳細は、「アプリケーションの管理者権利を定義するには」（652 ページ）を参照してください。

1. プロジェクトが開いている場合は一旦閉じます。Magic xpa のスタートアップ画面が表示されます。
2. **オプション→設定→ユーザグループ**を選択します。
3. **ユーザグループ**テーブルが表示されます。デフォルト状態は、SUPERVISOR GROUP のみが表示されています。
4. ***F4***（**編集→行作成**）を押下して 1 行追加します。
5. 任意のグループ名を入力します。この例では、CEO と入力されています。
6. **権利**カラムに ***Tab*** 移動して**ズーム**します。このグループにはまだ権利が割り当てられていないため、空のリストが表示されます。ここに任意の権利を追加します。
   - ***F4***（**編集→行作成**）を押下して 1 行追加します。
   - **キー**を入力するか、**ズーム**して一覧から選択します。
   - 権利が非公開の場合、一覧からは選択できません。それを直接入力する必要があります。この例では、非公開権利として、PAY333 が入力されています。

同様に必要なグループを作成します。

3. ユーザを定義する



まだ、アプリケーションがクローズされている状態のはずです。この状態のまま、次にユーザを設定します。

1. **ユーザ ID** テーブル（**オプション→設定→ユーザ ID**）を表示します。を選択します。デフォルトでは、SUPERVISOR のみが表示されています。
2. ***F4***（**編集→行作成**）を押下して 1 行追加します。
3. **ユーザ ID** カラムで、ユーザのログイン ID を入力します。ActiveDirectory 認証や LDAP 認証を行う場合、ユーザ ID は OS から渡すことができるため、ネットワークログインを使用することができます。
4. **名前**カラムには、ユーザ名を入力します。この名前は Magic xpa では使用されませんが、ログインしたユーザの名前を表示させるために使用することができます。
5. **パスワード**カラムでズームすることでログイン時のパスワードを指定することができます。ネットワーク経由でログインする場合は、パスワードが不要になる場合があります。
6. ユーザにグループを割り当てる場合は、**グループ**カラムから**ズーム**します。
7. 各グループに対して、このユーザを追加するには以下のようにします。
   - ***F4***（**編集→行作成**）を押下して 1 行追加します。
   - ここから**ズーム**して**グループ**を選択します。

権利キーを設定したグループにユーザを割り当てることで、ユーザに権利を割り当てる必要はありません。しかし、グループを使用しない場合は、ユーザ毎に**権利**カラムから**ズーム**して権利を設定する必要があります。

ユーザ ID/ グループを動的に追加／修正／削除するには

Magic xpa には、アプリケーションの実行時にセキュリティファイルのユーザ ID を追加／修正／削除することのできる関数があります。この機能を利用することでアプリケーション内でユーザ管理を行うことができます。

この機能を利用する場合は、あらかじめ Supervisor でログオンしている必要があります。

以下の関数が利用できます。

ユーザ ID を追加する

UserAdd() 関数の構文は以下の通りです。

UserAdd(*User ID,Name,Password,Information*)

パラメータ：

- User ID …… ログイン時に使用するユーザ ID
- Name …… 追加するユーザの名前
- User ID …… ユーザのパスワード
- User ID …… ユーザの情報

ユーザをグループを割り当てる

GroupAdd() 関数の構文は以下の通りです。

GroupAdd(*User ID,UserGroup*)

パラメータ：

- User ID …… 割り当て対象のユーザ ID
- UserGroup …… 割り当てるグループの名前

ユーザに権利を割り当てる

RightAdd() 関数の構文は以下の通りです。

RightAdd(*User ID,Rights*)

パラメータ：

- User ID …… 割り当て対象のユーザ ID
- Rights …… 割り当てる権利の名前

ユーザ ID を削除する

UserDel 関数の構文は以下の通りです。

UserDel(*User ID*)

パラメータ：

- User ID …… 削除するユーザ ID

グループに権利を割り当てる

GroupRightAdd() 関数の構文は以下の通りです。

GroupRightAdd(*UserGroup,Rights*)

パラメータ：

- UserGroup …… 割り当てるグループの名前
- Rights …… 割り当てる権利の名前

グループから権利を削除する

GroupRightRemove() 関数の構文は以下の通りです。

GroupRightRemove(*UserGroup,Rights*)

パラメータ：

- UserGroup …… 削除するグループの名前
- Rights …… 削除する権利の名前

グループを削除する

GroupRemove() 関数の構文は以下の通りです。

GroupRemove(*UserGroup*)

パラメータ：

- UserGroup …… 削除するグループの名前

ユーザから権利を削除する

RightRemove() 関数の構文は以下の通りです。

RightRemove(*User ID,Rights*)

パラメータ：

- User ID …… 割り当て対象のユーザ ID
- Rights …… 割り当てる権利の名前

Azure Key Vault のシークレットにアクセスするには

Azure Key Vault は、システムにおける機密情報の保管と安全なアクセスを実現するために提供される、Microsoft Azure のセキュリティサービスです。

Azure Key Vault は、パスワードやデータベース接続文字列などの汎用シークレットのセキュリティで保護されたストレージを提供します。

Azure Key Vault のシークレットにアクセスするためには、以下の手順が必要です。

1. Magic.ini の [MAGIC_ENV] セクションに、以下の 5 つのパラメータを手動で追加します。
   - ExternalVault=A …… 'A' は Azure Key Vault であることを意味します。このエントリがない場合、実行エンジンは Magic xpa 自体が論理名を保存するための vault であると見なします。
   - ExternalVaultName=XPAQA-Vault …… Azure vault の名前を指定します。
   - ExternalVaultTenantID …… Azure サービスを使用するために Azure Active Directory インスタンスを識別するためのユニークな方法
   - ExternalVaultApplicationID …… クライアント ID.
   - ExternalVaultClientSecret …… クライアントシークレット
2. 外部の vault シークレットを暗号化するために、次のコマンドを使用します。
   ```
   MgxpaRuntime -Encrypt=<ExternalVaultClientSecret>
   ```
   Magic xpa は暗号化された クライアントのシークレットを EncryptedContent.txt ファイルに書き込みます。
3. ファイルの内容を Magic.ini の [MAGIC_ENV] セクションの ExternalVaultClientSecret に配置します。
4. 実行 エンジンを起動します。
   - vault シークレットの認証に成功した場合、Azure Key Vault からのすべてのシークレットがロードされます。
   - vault シークレットの認証に失敗したた場合、起動時に mgerror.log にエラーが記録されます。また、ExternalVaultReload() 関数が呼び出された場合、この関数は空でないエラー文字列を返します。

論理名と Azure key vault に同じエントリが存在する場合、Azure key vault のシークレット名が優先されます。

key vault のシークレット名は、論理名のように複数の場所で使用することができます。たとえば、次のような場合です。Translate()、TranslateNR()、DB アクセスなど。

論理名 'AA' を使用してデータベースに接続している場合、エンジンの実行中に 'AA' から 'BB' に変更すると、最新の key vault を再度読み込むまで DB に接続されたままになります。

注： vault のシークレット名は、magic xpa の論理名より優先されます。

サポートバージョン：4.9

ActiveDirectory サーバを使用して認証させるには

ActiveDirectory サーバに登録したユーザ情報をもとに、Magic xpa にログインすることができます。これによりユーザ管理を Windows サーバ側で一本化させることができます。ActiveDirectory サーバを使用した認証方法には以下の２通りあります。

- ActiveDirectory認証を使用してドメインにログインしているクライアントPCの情報をもとに自動的にMagic xpaをログインする。
- ActiveDirectory サーバに対し LDAP 認証でログインする。

ここでは、これらの方法について説明します。

ActiveDirectory 認証を使用する

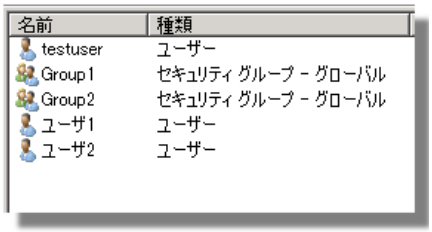
ActiveDirectory 認証を使用する場合は、クライアント PC がドメインに参加している必要があります。既に参加済みであることを前提に説明します。ドメインへの参加方法については、Windows の解説書などを参考にしてください。

環境設定を行う

1. 各クライアンの Magic xpa を SUPERVISOR でログインし、シークレットテーブル（オプション→設定→シークレット名）を開きます。
2. シークレットテーブルに以下の２行を追加します。

名前	実行名
Directory_Binding	WinNT:// ドメイン名 /
Domain_Name	ドメイン名

3. ユーザグループテーブル（オプション→設定→ユーザグループ）を開いてグループを登録します。ここには、Active Directory サーバ側でセキュリティグループとして登録されているグループ名と同じ名前を登録します。各グループには、権利キーを割り当てます。



Active Directory サーバ側

Magic xpa のグループテーブル

4. 動作環境ダイアログ（オプション→設定→動作環境）の**システムログオン**を ***D=Active Directory*** に設定します。

クライアント PC をドメインにログオンする

1. 各クライアンをドメインに対してログオンさせます。

2. この後、Magic xpa を起動すると自動的に Windows のログインユーザでログオンされた状態になります。

Windows のログオンダイアログ

Magic xpa のステータスバー

ヒント :*Magic xpa を Active Directory 認証に設定すると、ログオンダイアログを表示させることができなくなります。これにより、Windouws のログオンユーザ以外で使用することを防止させることができます。*

LDAP 認証を使用する

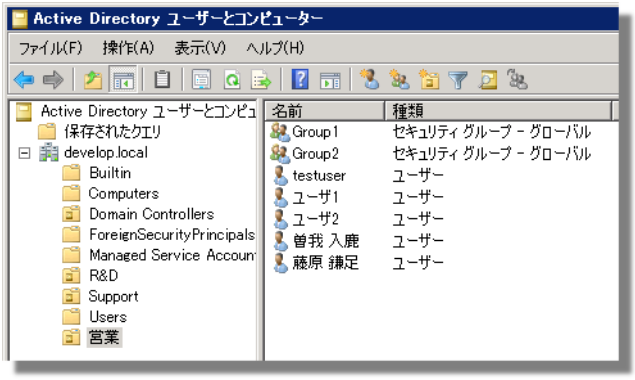
ActiveDirectory サーバに対しても LDAP 認証を使用することができます。LDAP 認証を使用する場合は、クライアント PC をドメインに参加させる必要はありません。ここでは、Active Directory サーバ上には上記と同じユーザが登録されていることを前提とします。

環境設定を行う

1. 各クライアンの Magic xpa を SUPERVISOR でログインし、Active Directory 認証の場合と同じように**ユーザグループ**テーブル（**オプション→設定→ユーザグループ**）を開いてグループを登録します。
2. **動作環境**ダイアログ（**オプション→設定→動作環境**）の**外部参照**タブを開き、LDAP サーバへの接続情報を設定します。

パラメータ	設定内容
LDAP アドレス：ポート番号	Active Directory サーバのホスト名：ポート番号（デフォルトは、389）
LDAP 接続文字	CN=$USER$,OU= ユーザが定義されている OU 名 ,DC= ドメイン名
LDAP ドメインコンテキスト	DC= ドメイン名

例えば、Active Directory サーバ上に以下のようにユーザやグループが登録されていた場合。



以下のように定義します。

- LDAP アドレス：ポート番号 …… ServerName:389
- LDAP 接続文 fujiwara 字 …… CN=$USER$,OU= 営業 ,DC=develop
- LDAP ドメインコンテキスト…… DC=develop

この設定により、OU（組織単位）が営業内に登録されているユーザに対してのみログオンできます。

3. 動作環境ダイアログの**システムログオン**を ***L=LDAP*** に設定します。

Magic xpa でログオンする

1. 各クライアンで Magic xpa を起動します。
2. ログオンダイアログ（オプション→ログオン）を開き、ActiveDirectory サーバに登録されているユーザ ID とパスワードでログオンします。
3. ログオン情報に誤りがある場合は、Invalid credentials というエラーメッセージがステータスバーに表示されます。

自動ログオンを設定する

LDAP 認証の場合も、Active Directory 認証と同じように自動ログオンを行うことができます。この場合、クライアント PC は、ドメインに参加する必要はありませんが、Active Directory サーバに登録されているユーザ ID と同じ ID でログオンされている必要があります。

1. 各クライアンの Magic xpa を SUPERVISOR でログインし、シークレットテーブル（オプション→設定→シークレット名）を開きます。
2. シークレットテーブルに以下の２行を追加します。

名前	実行名
LDAP_USER	Active Directory サーバに登録されているユーザ ID
LDAP_PASS	パスワード

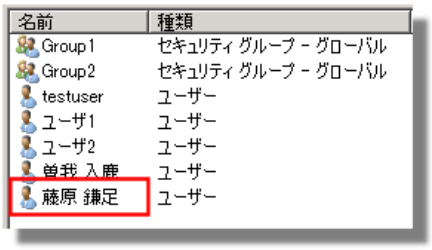
これで Magic xpa を起動すると、Windows にログオンしたユーザ ID で Magic xpa はログオンされます。Active Directory サーバに登録されていないユーザ ID で Windows にログオンした場合、Magic xpa 起動時に以下のエラーが表示されます。

The user does not exist in the LDAP server or the credentials are invalid.

このような場合も、ログオンダイアログを開いてログオンし直すことができます。

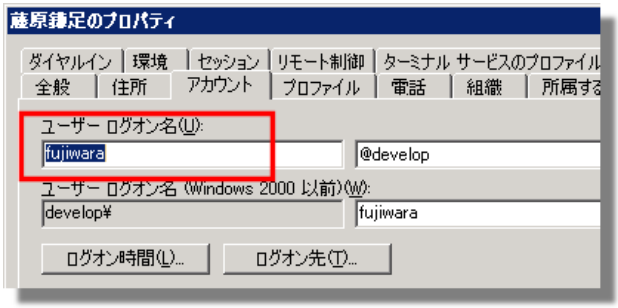
ユーザログオン名を使用する

LDAP で認証する場合、通常は DN（識別名）を使用しますが、Windows が Active Directory サーバにログインする場合は、ユーザログオン名が使用されます。Magic xpa では、このどちらを使用しても認証させることができます。



dn:CN= 藤原鎌足 ,ou= 営業 ,DC=develop

DN（識別名）



userPrincipalName:fujiwara@develop

ユーザログオン名

LDAP 接続文字列の設定

ユーザログオン名を使用して認証させる場合は、**動作環境**ダイアログ（**オプション→設定→動作環境**）で **LDAP 接続文字**を、***CN=$USER$@ ドメイン名***と設定します。

注： この設定にすると、DN でログオンできなくなります。

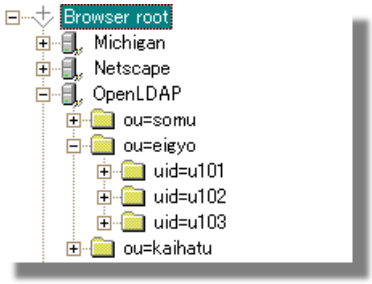
OpenLDAP サーバを使用して認証させるには

OpenLDAP は、フリーの LDAP サーバです。ここでは、OpenLDAP を使用した認証方法について説明します。

OpenLDAP 自体の操作については、関連する書籍や Web サイトなどを参照してください。

ディレクトリ構造例

ここでは、以下のようなディレクトリ構造が定義されていることを前提としています。



ldif ファイル上は以下のようになります。

```
#TOP ツリー用
dn: o=example
objectClass: organization
o: example
# 部署ツリー用
dn: ou=somu,o=example
objectClass: organizationalUnit
ou: somu
dn: ou=eigyo,o=example
objectClass: organizationalUnit
ou: eigyo
# 社員ツリー用
dn: uid=u001,ou=somu,o=example
objectClass:inetOrgPerson
cn: nobunaga
sn: oda
uid: u001
userPassword: p-oda
homePhone:123-456-7890
mail:oda@magic.co.jp
homePostalAddress:Inazawa City Aich-ken
dn: uid=u101,ou=eigyo,o=example
objectClass:inetOrgPerson
cn: ieyasu
sn: tokugawa
uid: u101
userPassword: p-tokugawa
homePhone:098-765-4321
mail:tokugawa@magic.co.jp
homePostalAddress:okazaki City Aich-ken
```

環境設定を行う

1. 各クライアンの Magic xpa を SUPERVISOR でログインし**ユーザグループ**テーブル（**オプション→設定→ユーザグループ**）を開いてグループを登録します。
2. **動作環境**ダイアログ（**オプション→設定→動作環境**）の**外部参照**タブを開き、LDAP サーバへの接続情報を設定します。

パラメータ	設定内容
LDAP アドレス：ポート番号	LDAP サーバのホスト名：ポート番号（デフォルトは、389）
LDAP 接続文字	uid=$USER$,OU= ユーザが定義されている OU 名 ,o= ツリーの組織名
LDAP ドメインコンテキスト	o= ツリーの組織名

3. 動作環境ダイアログの**システムログオン**を *L=LDAP* に設定します。

Magic xpa にログオンする

1. 各クライアンで Magic xpa を起動します。
2. ログオンダイアログ（オプション→ログオン）を開き、LDAP サーバに登録されているユーザ ID（uid）とパスワード（userPassword）でログオンします。
3. ログオン情報に誤りがある場合は、Invalid credentials というエラーメッセージがステータスバーに表示されます。

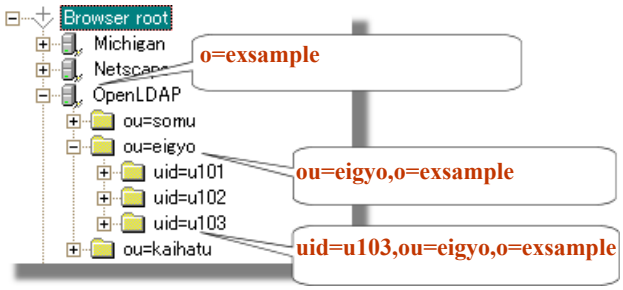
ディレクトリのデータ検索を行う

LDAP サーバを使用してログオンした場合、**LDAPGet** 関数を使用してディレクトリサーバで管理だれているデータベースを検索することができます。関数の構文は以下の通りです。

LDAPGet(*SearchBase, SearchLevel, SearchFilter, Attribute, Delimiter***)**

パラメータ：

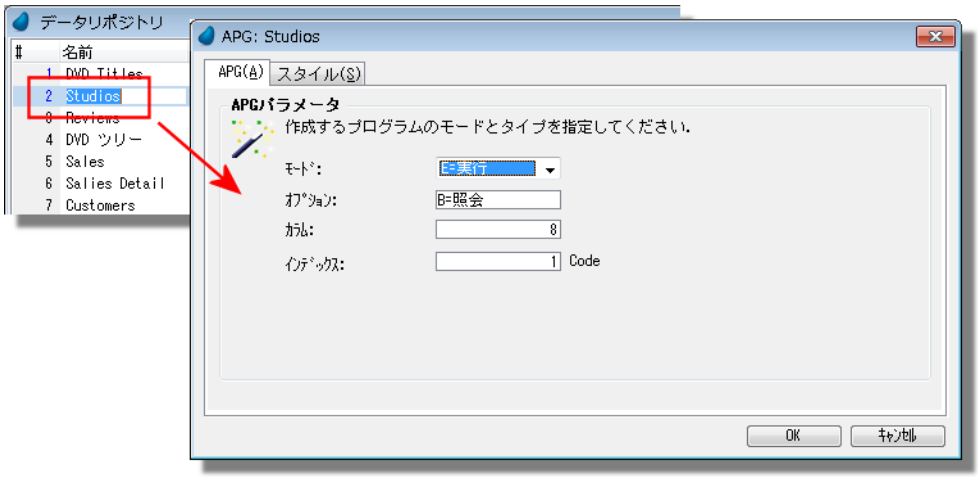
- *SearchBase* ：検索の開始点を指定した文字列。空白の場合は、ドメインコンテキストの値が使用されます。
  例えば、*SearchBase* は以下のようになります。



- *SearchLevel* ：検索レベルを指定します。
  SearchBase が *o=example* として指定された場合、検索可能な範囲は以下のようになります。
  • B…基本検索 → *SearchBase* で指定されたルートのみ。
  • T…サブツリー検索 → ツリー内の全てのノードに対して検索することができます。
  • O…１階層検索 → 各 ou レベルまで検索されます（uid 以下は検索されません）。
- *SearchFilter* ：LDAP の検索フィルタを含む文字列です。
  例えば、*homePhone=1** と指定した場合、homePhone の先頭が 1 になっているユーザ情報が対象となります。
- *Attribute* ：必要とされる情報タイプを定義する文字列です。
  例えば、*uid* と指定した場合、uid の内容が戻り値として返ります。
- *Delimiter* ：戻り値の値が複数の場合、値と値の間を区切る文字列を指定します。

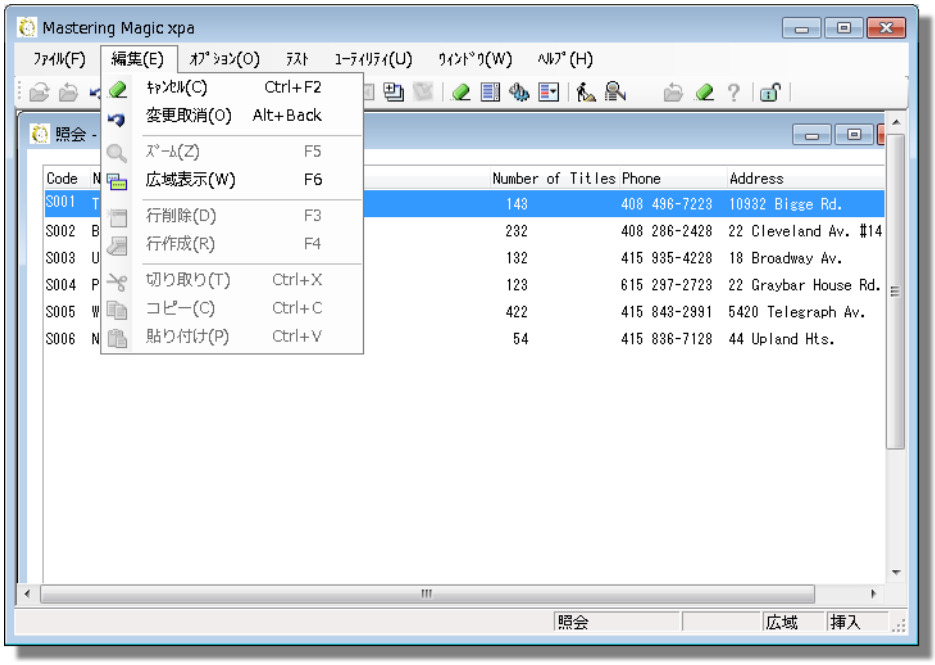
第 31 章 : ユーティリティ

データソースを参照するには



アプリケーションの開発中に、使用しているデータソースの内容を簡単に参照できれば便利です。Magic xpa では、以下のようにすることで参照することができます。

1. データソースリポジトリ（*Shift+F2*）を開きます。
2. 参照したいデータソースにカーソルを置きます。
3. *Ctrl+G*（オプション→ APG）を押下します。APG ダイアログが表示されます。
4. OK をクリックします。データソースのすべてのレコードが表示される参照プログラムが起動されます。



表示されるまでには、2 ～ 3 秒かかる場合もあります。表示結果をもとに、特定のレコードを検索するために位置付けや範囲指定の機能を使用することができます。レコードの追加／削除／修正を行ったり、プルダウンメニューに表示されるオプションを利用することができます。

APG には、表示をより便利に行うためのさまざまなオプションがあります。詳細は、「データソースを参照する簡単なプログラムを作成するには」（673 ページ）を参照してください。

*ヒント : 参照プログラムが実行される前に、メインプログラムのタスク前が実行されます。作業時間管理表などの参照プログラムを実行しようとする間、タスク前で時間のかかる処理が定義されている場合、**RunMode()** 関数を使用してそれを使用不可にすることもできます。詳細は、第 18 章 :「初期設定のプログラムを省略するには」（423 ページ）を参照してください。*

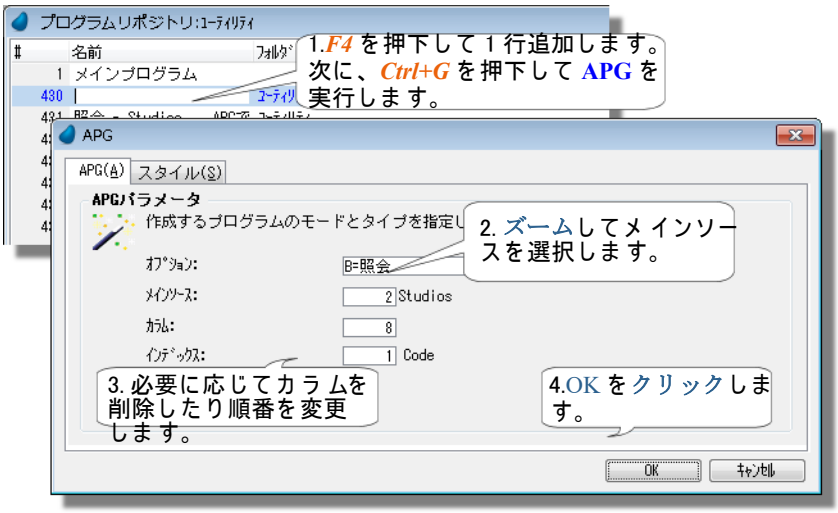
注 : データソースを参照するプログラムをプログラムリポジトリ上に作成したい場合は、似たような操作を行います。「データソースを参照する簡単なプログラムを作成するには」（673 ページ）を参照してください。

データソースを参照する簡単なプログラムを作成するには

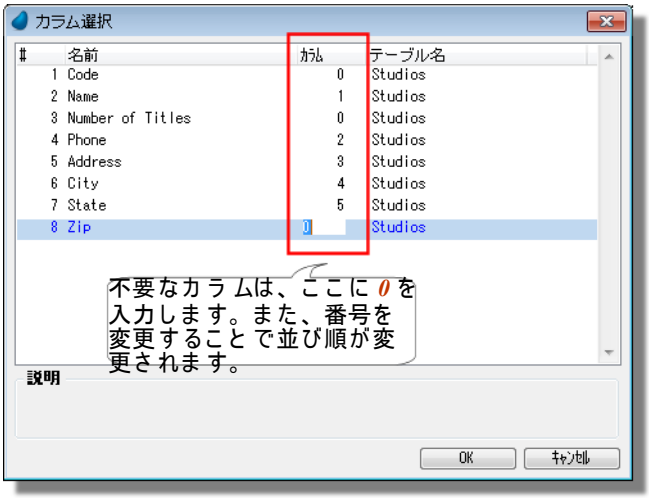
データソースの簡単な参照プログラムを容易に作成することができます。これらの簡単な参照プログラムは、デバッグ時に利用でき、レコードの追加、削除や、修正を行うことができます。Magic xpa のデータ出力ウィザードを実行したり、データを XML などの他のフォーマットで出力するための基本プログラムとして使用することもできます。また、より複雑なプログラムを作成するためのスケルトンプログラムとして使用することができます。

どのように作成するかを以下で説明します。

参照プログラムを作成する

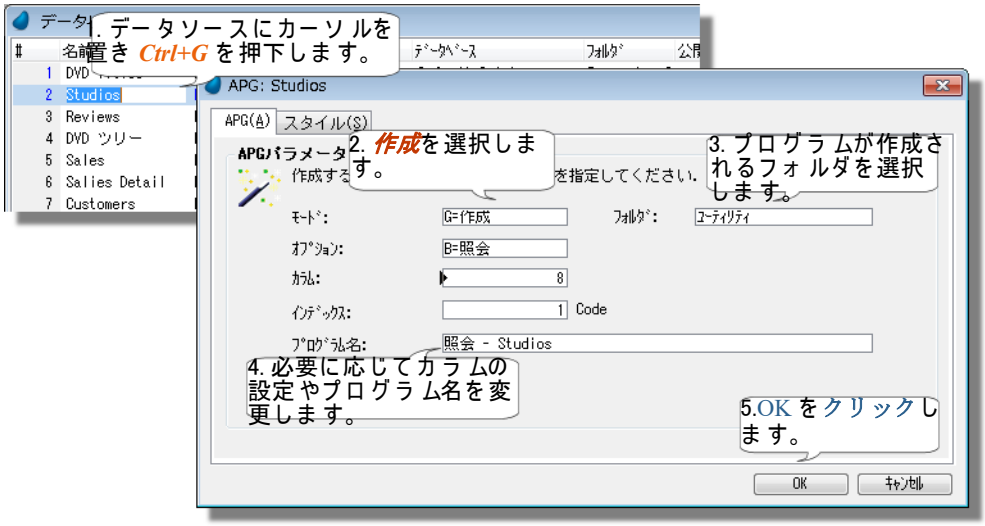


1. *F4*（編集→行作成）を押下してプログラムリポジトリに 1 行追加します。*Ctrl+G*（オプション→ APG）を押下して APG を起動します。
2. メインソースからズーム（*F5* またはダブルクリック）して一覧から使用するのデータソースを選択します。



3. カラムカラムからズームすることで不要なカラムを削除したり、カラムの順番を変更したりできます。デフォルトでは、すべてのカラムがデータソースに定義されている順番に並びます。
4. OK をクリックすると参照プログラムが作成されます。

データソースリポジトリ上でカーソルがパークしているデータを参照するプログラムを作成することもできます。APG ダイアログでデータソースを選択する必要がない以外は、上記の場合と同じような操作で作成できます。



スタイルタブでは、データをテーブル形式で表示させるか、１レコード１画面の形式で表示させるかとか、立体表示させるか平面表示させるか、ウィンドウのサイズの指定、モデルを使用するかどうかが指定できます。

OK をクリックすると、プログラムが作成されます。*F7* を押下してプログラムを実行したり、ズームして編集することができます。

プログラムの構文チェックを行うには

どのような開発言語であってもプログラムを実行する前に、構文が正しいかどうかのチェックを行う必要があります。Magic xpa では、プログラムの実行前にコンパイルを行わないため、構文チェックを行うか否かは任意ですが、プログラムにエラーがある場合は正しく動作しません。

構文チェックで Magic プログラムの確認を行うと（プログラム量によりますが）数秒かかります。構文チェックが完了すると、エラー箇所が存在すればエラーリストが表示されます。エラーをクリックすると、該当するオブジェクトに移動し、エラー箇所を修正することができます。

すべてのプログラムの構文チェックを一度に行うことができます。これは、アプリケーションを製品化する前にどこにもエラーが存在しないことを確認する上で必要な作業です。

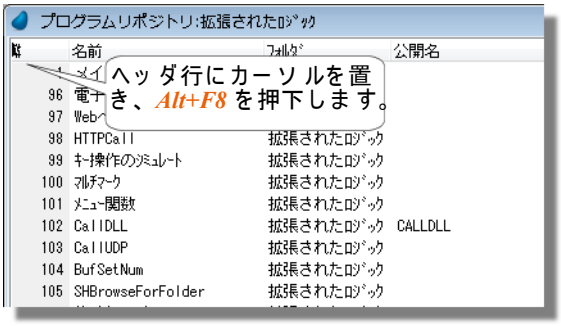
1 つのプログラムの構文をチェックする

1. チェックしたいプログラム上にカーソルを置きます。
2. *F8*（オプション→構文チェック）を押下します。プログラムのチェック中にいくつかのウィンドウが表示されます。また、以下の２つのどちらかの状態になります。
   - ステータス行に、「プログラムは正常です。」が表示されます。
   - ステータス行に「構文チェックが終了しました - チェック結果を確認してください .」が表示され、チェック結果ペインにいくつかのエラーメッセージが表示されます。チェック結果ペインが表示されない場合は、表示→チェック結果（*Alt+F3*）を選択してください。

プログラムにエラーがある場合、チェック結果ペインに表示されるエラーのリストを確認しながらプログラムを修正します。詳細は、「チェック結果を使用するには」（679 ページ）を参照してください。

どのエラーをどのように表示させるかといったカスタマイズを行うこともできます。第 16 章 :「表示されたチェックメッセージを制御するには」（389 ページ）を参照してください。

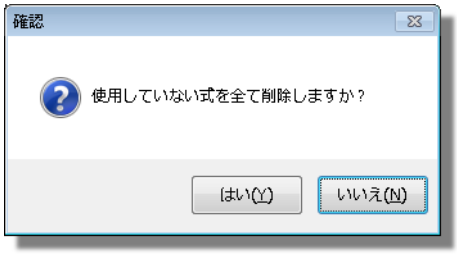
複数のプログラムを一度にチェックする



1 度に複数のプログラムをチェックすることができます。エラーメッセージは、チェック結果ペインでグループ化されて表示されます。1 つのプログラムをチェックした場合と同じように表示されるエラーリストをもとにプログラムを修正します。

1. チェックする一番先頭のプログラムか、プログラムリポジトリのヘッダ行にカーソルを置きます。特定のフォルダのみを表示させている場合は、そのフォルダ内のプログラムのみチェックされます。

2. *Alt+F8*（**オプション→カーソル以降チェック**）を押下します。

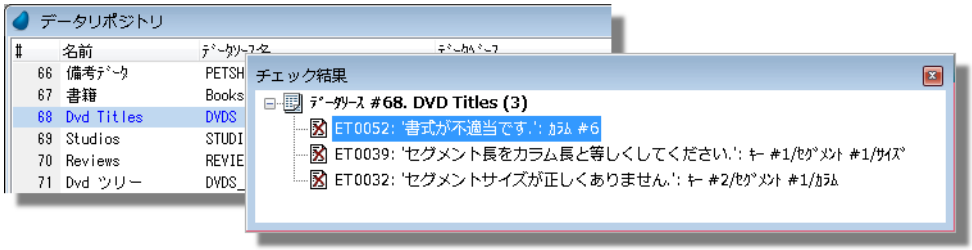


3. 「使用していない式をすべて削除しますか？」という**確認**ダイアログが表示されます。はいを**クリック**すると、どこからも参照されていない式が存在した場合、**確認**ダイアログを表示しないで自動的に削除されます。
4. 構文チェッカーは**プログラム**リポジトリ内のすべてのプログラムをチェックし、エラーが存在した場合、**チェック結果**ペインにリスト表示されます。

データソースの構造を検証するには

新しいデータソースを作成する場合、それを使用する前に構造に問題がないかどうかをチェックすることは、バグを減らす１つの手段です。この処理には、（データソースのサイズによりますが）数秒かかります。

１つのデータソースを構文チェックする



1. チェックしたいデータソース上にカーソルを置きます。
2. *F8*（オプション→構文チェック）を押下します。データソースのチェック中にいくつかのウィンドウが表示されます。また、以下の２つのどちらかの状態になります。
   - ステータス行に、「データソースは正常です .」が表示されます。
   - ステータス行に「構文チェックが終了しました - チェック結果を確認してください .」が表示され、チェック結果ペインにいくつかのエラーメッセージが表示されます。チェック結果ペインが表示されない場合は、表示→チェック結果（*Alt+F3*）を選択してください。

データソースにエラーがある場合、チェック結果ペインに表示されるエラーのリストを確認しながらデータソースを修正します。詳細は、「チェック結果を使用するには」（679 ページ）を参照してください。

どのエラーをどのように表示させるかといったカスタマイズを行うこともできます。第 16 章：「表示されたチェックメッセージを制御するには」（389 ページ）を参照してください。

複数のデータソースを一度にチェックする

1 度に複数のデータソースをチェックすることができます。エラーメッセージは、チェック結果ペインでグループ化されて表示されます。1 つのデータソースをチェックした場合と同じように表示されるエラーリストをもとに修正します。

1. チェックする一番先頭のデータソースか、データリポジトリのヘッダ行にカーソルを置きます。特定のフォルダのみを表示させている場合は、そのフォルダ内のデータソースのみチェックされます。
2. *Alt+F8*（オプション→カーソル以降チェック）を押下します。
3. 構文チェッカーはデータリポジトリ内のすべてのプログラムをチェックし、エラーが存在した場合、チェック結果ペインにリスト表示されます。

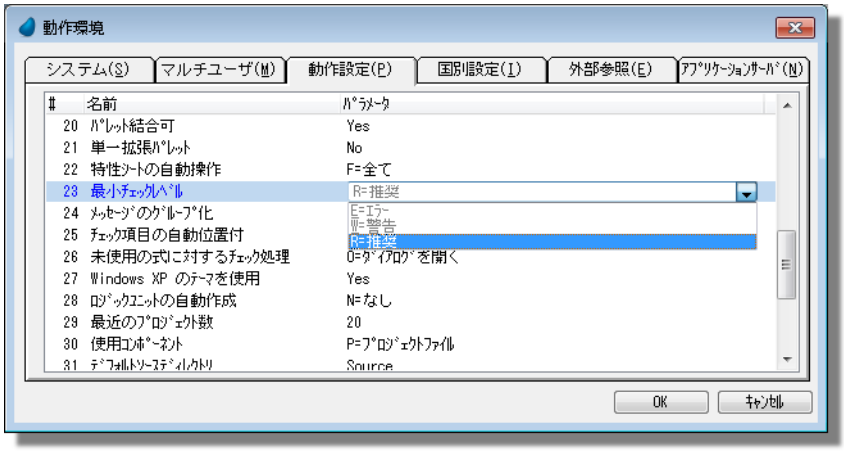
構文チェックの表示メッセージを絞るには

構文チェックユーティリティを使用する際に、どのメッセージを表示させるかを指定することができます。これは２つの方法で指定できます。

- チェック結果ペインに表示されるエラーの最小のレベルを設定する
- 各メッセージレベルを設定する

以下、これらの設定方法について説明します。

構文チェックの最小レベルを設定する

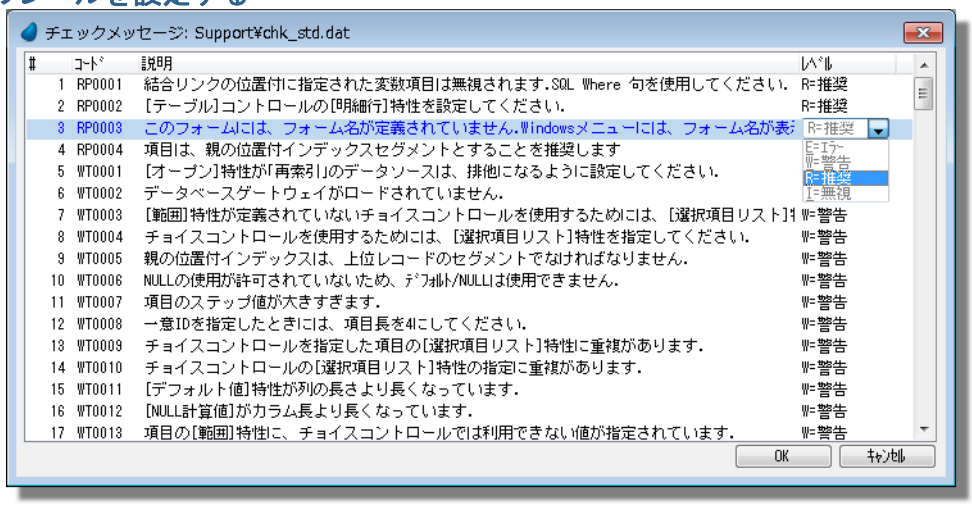


1. オプション→設定→動作環境→動作設定の**最小チェックレベル**に移動します。
2. 以下のように設定できます。

設定	表示されるメッセージ
E= エラー	エラー
W= 警告	エラー 警告
R= 推奨	エラー 警告 推奨

すべてのメッセージを表示させたい場合は、***推奨レベル***を使用します。

メッセージのレベルを設定する



1. チェックメッセージテーブル（オプション→設定→チェックメッセージ）を開きます。
2. 各メッセージ毎に**レベル**を ***R= 推奨***、***W= 警告***、***E= エラー***、または ***N= 無視***のどれかに設定します。

エラーリストをカスタマイズし最小レベルに変更すると、チェッカー内では表示させたいメッセージのみが表示されます。

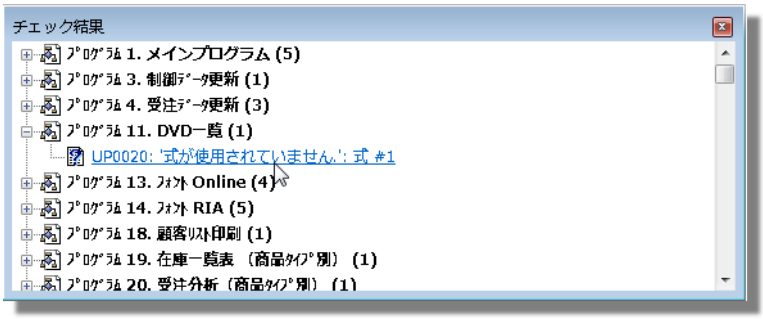
チェック結果を使用するには

構文チェックが終了すると、チェック結果ペインにチェック結果がリスト表示されます。次に、これらのメッセージに対処する必要があります。これを行うには、２つの基本的な方法があります。

- チェック結果ペインの各メッセージをクリックします。
- *Ctrl+F8* を押下してメッセージからメッセージに移動します。

それぞれの方法について説明します。

チェック結果のリストを使用する



1. 修正したいエラー上にカーソルを置きます。
2. ダブルクリックします。
3. エラーの対象となるオブジェクトに移動します。
4. すべてのエラーを修正するまで、上記の操作を続けます。

各タスクまたはデータソースがツリー上に独自のノードを持っている場合、上の図で示されているように、チェック結果はオブジェクトをもとにグループ毎に表示されます。それらが、どのようにグループ分けされるかは、動作環境で設定できます（第 16 章：「表示されたチェックメッセージを制御するには」（389 ページ）を参照してください）。

キーボード操作で移動する

Ctrl+F8（オプション→次のチェックメッセージ）を押下して移動することもできます。

必要条件： 設定→オプション→動作環境→動作設定の**チェック項目の自動位置付**を *Yes* に設定する必要があります。

1. 構文チェックを実行すると、メッセージ内で参照されたタスクが自動的にオープンされ、問題のある箇所にカーソルが移動します。
2. 次のエラーに移動する場合は、*Ctrl+F8* を押下します。

プロジェクトをバックアップするには

作業内容を常にバックアップすることは重要なことです。自動的にバックアップ処理を行うことのできるソース管理用のソフトを使用しているのであれば問題ありませんが、使用していない場合のために、バックアップ用の簡単なオプション機能があります。

- OS 上のファイルとしてバックアップする
- Magic xpa のリポジトリ出力ユーティリティを使用して 1 つの XML ファイルに出力する
- プログラムをコピーする

以下、これらの操作方法について説明します。

OS 上のファイルとしてバックアップする

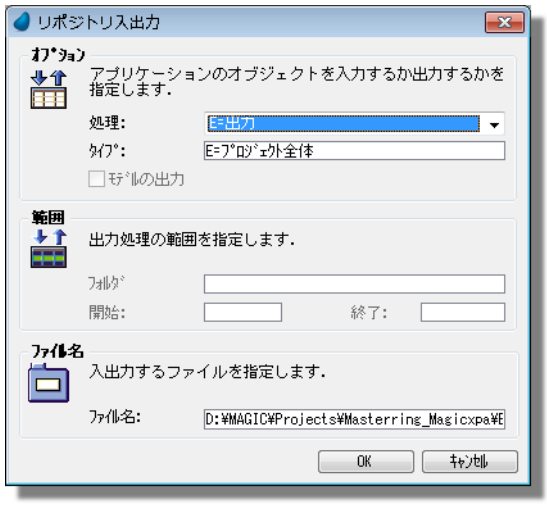
Magic xpa が新しいプロジェクトを作成した場合、プロジェクト構造はルート上にプロジェクト（***.edp***）ファイルといくつかのサブディレクトリが作成されます。実際のプログラムソースは Source サブディレクトリ内に一連の XML ファイルとして格納されます。通常、イメージや HTML テンプレートなどの他のサポートファイルは、別のサブディレクトリに置かれることになります。

従って、プロジェクト全体をバックアップする最も簡単な方法は、ルートレベルのフォルダ（この例では、Utilities）から圧縮するだけです。必要であれば、バージョンごとに圧縮ファイルを作成し、保存することもできます。この方法は、現在のプロジェクトに対するすべてのスナップショットを保管することになります。

ハードウェアが故障した場合に備えて、他のメディア（CD や外部 HDD など）上にこのディレクトリをコピーすることもできます。

ヒント : *Magic.ini ファイルが同じディレクトリ内に存在しているか否かは、インストール方法によります。Magic.ini ファイルもいろいろな変更が行われる可能性があるため、バックアップしておいた方が無難です。*

リポジトリ出力ファイルを作成する



Magic プロジェクトからオブジェクトを出力したり、プロジェクト全体を出力することができます。この方法は、プロジェクトをバージョンごとに 1 つのファイルで管理する場合に利用できます。

1. ファイル→リポジトリ入出力（*Ctrl+Shift+E*）を選択し、リポジトリ入出力ダイアログを開きます。
2. デフォルトでは、**処理**は ***E= 出力***になっています。このままにしておきます。
3. **タイプ**は、デフォルトでは ***E= プロジェクト全体***になっています。プロジェクト全体をバックアップする場合は、このままにしておきます。特定のオブジェクトのみをバックアップする場合は、対象となるリポジトリを指定します。また、出力対象のフォルダを選択したり、一定の連続した出力範囲を指定することもできます。
4. **ファイル名**を指定します。ここには、出力するファイルのパスと名前を指定します。ここからズームして保存先を選択することもできます。ファイルが作成されると「.xml」という拡張子が自動的に付加されます。
5. OK をクリックします。

出力処理が終了すると、ファイルが作成されます。

参照 : 第 2 章 :「別のプロジェクトにオブジェクトを転送するには」（36 ページ）

プログラムをコピーする

プログラムの修正作業を行う前に、**編集→登録→複写登録**（***Ctrl+Shift+R***）を選択してバックアップコピーを作成することができます。この方法は、プロジェクト全体をバックアップするほど全体的なものではありませんが、プログラムを修正前に戻すことができる簡単な方法です。実験的に修正処理を行いたい場合は、有効な方法です。

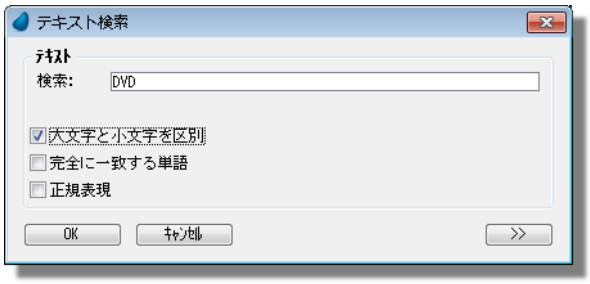
作業手順についての詳細は、第 1 章：「入力行を複写するには」（12 ページ）か、第 1 章：「入力行を他の行の内容に置き換えるには」（14 ページ）を参照してください。

オブジェクト内のテキストを検索 / 置換するには

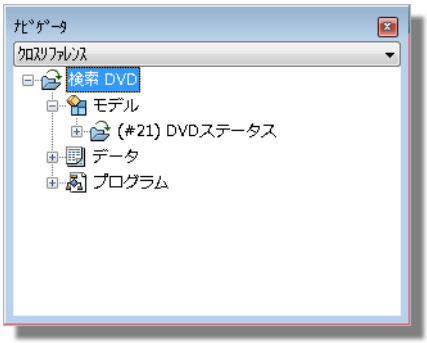
Magic xpa は、プロジェクト内のテキストを検索し、その内容を変更することができます。例えば、帳票処理の出力結果や画面をハードコピーしたものをもっていても、どのタスクに関係するものかが分からない場合があります。また、あるデータの名前を変更する必要が発生した場合を想定してもかまいません。例えばセールスマンの代わりに営業員という名前に変更したり、ハードコートされている会社名を変更することなどが考えられます。

以下では、Magic xpa に備えられているテキストの検索機能や置換機能についての説明を行います。

テキストを検索する



1. **編集→検索と置換→テキスト検索**（*Ctrl+Shift+F*）を選択します。
2. **テキスト検索**ダイアログが表示されます。
3. 検索したいテキストを入力します。この例では DVD と入力しています。
4. 大文字と小文字を区別して検索する場合は、**大文字と小文字を区別**をチェックします。
5. 文字列全体を検索対象にする場合は、**完全に一致する単語**をチェックします。
6. マスク文字を使用する場合は、**正規表現**をチェックします。これらのオプションの説明は、Magic xpa の『リファレンスヘルプ』を参照してください。
7. 下に表示されている >> ボタンを**クリック**すると、検索対象のオブジェクトを指定できる詳細ウィンドウが表示されます。
8. OK を**クリック**します。

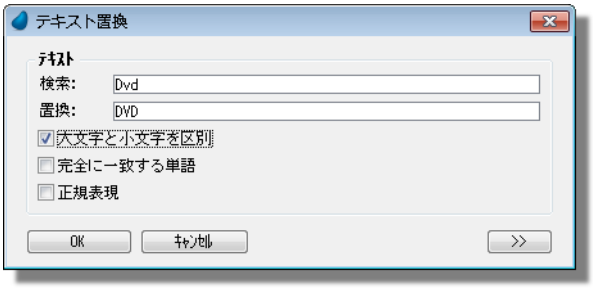


検索処理が終了すると、**ナビゲーション**ペインに検索されたテキストに対するすべての参照リストが表示されます。この検索結果を**クリック**すると該当するテキストを持つオブジェクトに移動します。

さて、検索テキストのリストが表示されたら、このリストを保存したり印刷したりすることができます（「検索結果を保存 / 印刷するには」（684 ページ）を参照してください）。

***ヒント :** 検索リストを使用して作業する場合、必要であれば F3（または編集→行削除）を押下してリスト上の項目を削除することができます。リストが大量に表示される場合は、テキストを修正後にリストを削除することで、作業が効率化する場合もあります。*

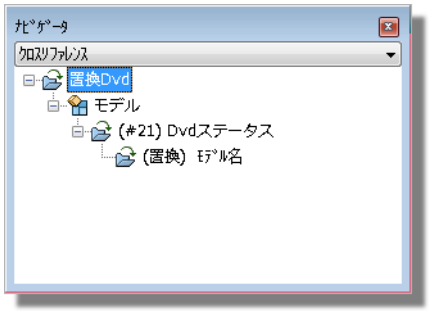
テキストを置換する



1. **編集→検索と置換→テキスト置換**を選択します。
2. **検索**には、検索対象のテキストを入力します。この例では、Dvd が入力されています。また大文字小文字を区別するように指定されています。
3. **置換**には、検索されたテキストを置き換えるテキストを入力します。
4. 前の例で説明されているようにテキストを検索したい場合は、他のオプションを設定します。
5. OK を**クリック**します。

6. テキストが見つかると、検索された最初のテキストに位置付き、**置換の確認**ダイアログが表示されます。ここには以下のオプションがあります。
   - 置換：位置付けられたテキストを置換して次の置換対象に移動します。
   - 全て置換：すべてのテキストを一度に置換します。
   - スキップ：このテキストは置換せず、次の置換対象に移動します。
   - キャンセル：検索処理を中断します。

注： **式**エディタと**フォームデザイナ**で表示される**置換の確認**ダイアログは、他のオブジェクト上で表示されるんものと異なります。



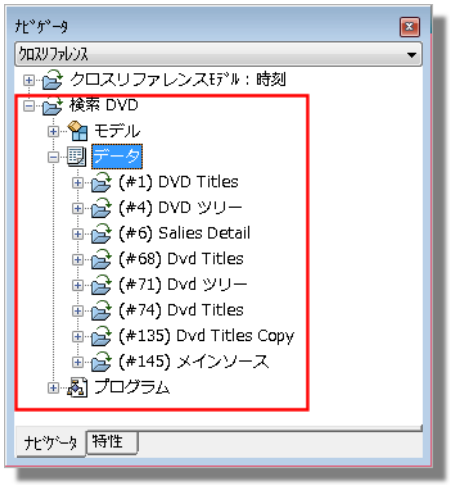
7. 置換が終了すると、**ナビゲータ**ペインに置換されたすべての項目リストが表示されます。これらの項目を**クリック**すると対応するオブジェクトに移動し、置換結果を確認することができます。

検索結果を保存 / 印刷するには

テキストの検索や置換、またはクロスリファレンスを行うと、実行結果はナビゲータペインに表示されます。これらの結果をファイルに保存したり印刷することで便利な場合があるかもしれません。例えば、修正処理にどれだけの工数がかかるかを概算する場合に利用できます。

検索結果を保存する

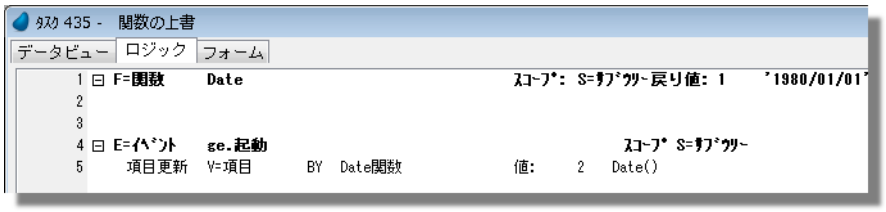
1. 保存したいセクション内のツリー項目にカーソルをパークします。検索 DVD のようなノードの先頭か、ツリー内の項目にパークできます。この例では、データノードにパークしているため、四角でマークされた検索 DVD セクション全体が保存されます。
2. 編集→検索と置換→検索結果の保存を選択します。
3. Windows の名前をつけて保存ダイアログが表示されます。ファイルの保存場所とファイル名を指定します。
4. 保存をクリックします。
5. 選択されたセクションがテキストファイルに保存されます。



検索結果を印刷する

1. 印刷したい項目にカーソルをパークします。この例では、検索 DVD のようなノードの先頭か、ツリー内の項目にパークできます。選択されたセクション全体が印刷されます。
2. 編集→検索と置換→検索結果の印刷を選択します。
3. Windows の印刷ダイアログが表示されます。出力先のプリンタを選択し、印刷をクリックします。
4. 選択されたセクションが印刷されます。

Magic の内部関数を上書きするには



Magic xpa には、様々な機能が利用できる関数があります。しかし、これらの関数をオーバライドすることで簡単にカスタマイズすることができます。

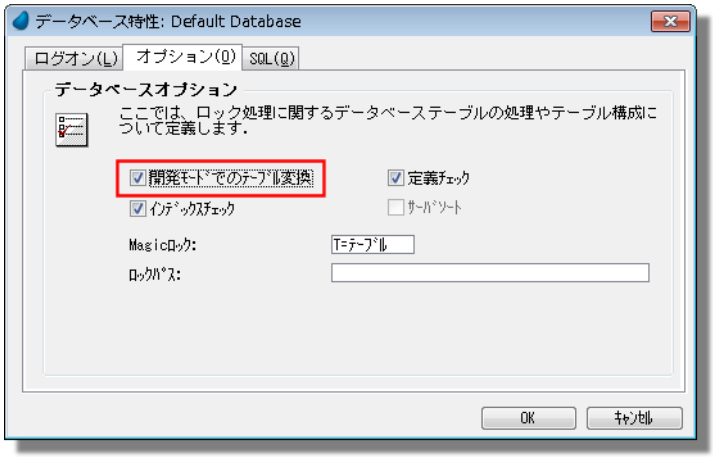
第 10 章 :「プロジェクト全体で有効な関数を作成するには」（224 ページ）の説明を参考にして、内部関数と同じ名前のユーザ定義関数を作成することでオーバライドできます。

上記の例では、Date() 関数がオーバライドされ、常に “1980/01/01“ が返るようになります。

Magic xpa とデータベース間でのテーブル構造の不整合に対応するには

ISAM テーブルを使用している場合、実際のテーブル構造と Magic xpa 側での定義内容とが合っていない可能性があります。このような状態は、Magic xpa 内で変更したために発生する場合と、DBMS 側で変更された場合、２つの異なる Magic プロジェクトで同期を取って変更しなかった場合に発生する可能性があります。

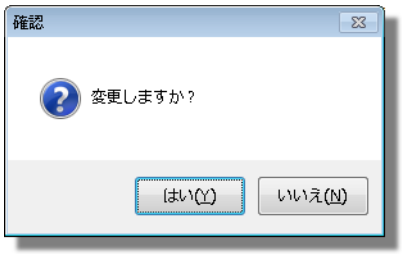
Magic xpa で自動的にテーブルを変換できるようにする



1. プロジェクトが開いている場合、一旦閉じます。
2. **データベース**テーブル（**オプション→設定→データベース**）を開きます。
3. 使用するデータベースを選択し、***Alt+Enter*** を押下して**データベース特性**を開きます。
4. **オプション**タブを**クリック**します。
5. **開発モードでのテーブル変換**特性がチェックされていることを確認します。

開発モードでのテーブル変換特性がチェックされている場合、**データソース**リポジトリでテーブル定義を変更すると、Magic xpa は自動的に実際のファイルの変更処理を実行します。

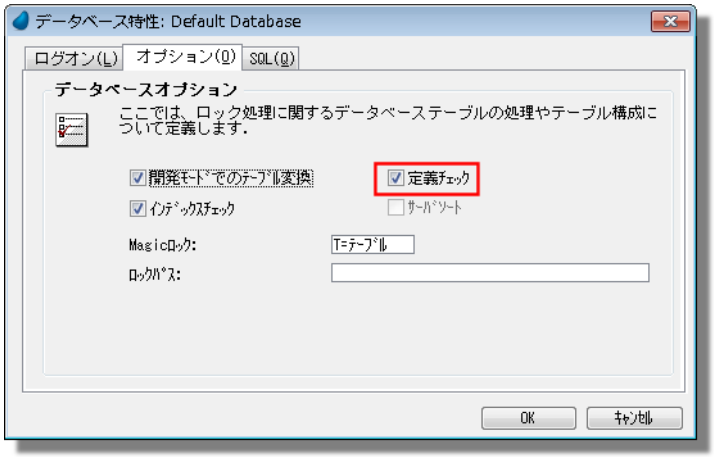
例えば、カラムの型を数値型から文字型に変更した場合、そのファイルが存在していれば以下のようなメッセージダイアログが表示されます。



はいを**クリック**すると、数値型項目は文字型に変更されます。また、項目の位置やインデックスが変更された場合も同様に変更処理が実行されます。

テーブルを変更した場合、毎回データを変更するようにする必要があります。**確認**ダイアログが表示されたときに、**No** を**クリック**するとテーブル定義はデータと同期されなくなります。

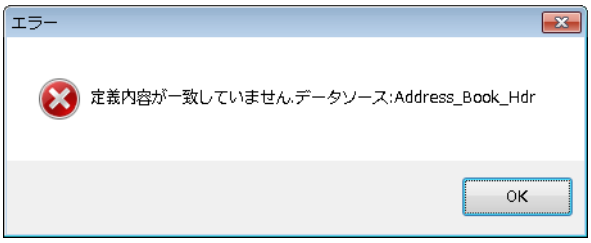
互換性をチェックする



テスト中に、ISAM テーブルに対して同期が取られているかどうかを確認することができます。これを行うには、以下のように設定します。

1. プロジェクトが開いている場合、一旦閉じます。
2. **データベース**テーブル（**オプション→設定→データベース**）を開きます。
3. 使用するデータベースを選択し、***Alt+Enter*** を押下して**データベース特性**を開きます。
4. **オプション**タブを**クリック**します。
5. **定義チェック**特性がチェックされていることを確認します。

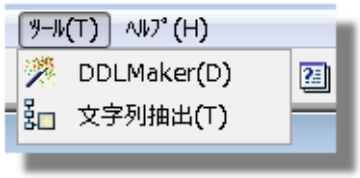
これで、同期していないテーブルにアクセスしようとすると、以下のようなメッセージが表示されます。



また、エラーイベントが発生するため、カスタマイズされたメッセージを表示したり、ログを出力したりすることができます。

外部ツールを Magic xpa Studio に追加するには

Magic xpa Studio には**ユーザ定義開発メニュー**と呼ばれる機能があります。



ユーザ定義開発メニューを使用して、アプリケーション用のツールメニューをカスタマイズすることができます。**Magic xpa Studio** で使用されるウィザードのいくつかは Magic アプリケーションです。

ツールメニューは、Magic.ini の［TOOLS_MENU］セクションに定義することで **Magic xpa Studio** のウィンドウに表示されるようになります。以下の例で説明します。

```
[TOOLS_MENU]
DDLBDR = A,DDLMaker(&D),,Add_On¥DDLMaker¥PVDDLMaker.ecf,,,Add_On¥DDLMaker¥DDLMaker_suf.opr,ImageFor = B +
ToolNumber = 60 ToolGroup = 1 ToolTip = ″ﾃﾞｰﾀ辞書を作成します.″
String Extractor = A, 文字列抽出 (&T),,Add_On¥StringExtractor¥StringExtractor.ECF,,,,Ima+
+ geFor = B ToolNumber = 283 ToolGroup = 1 ToolTip = ″多言語辞書用の文字列を抽出します.″
```

設定できるメニューとして以下の 4 種類があります。

- A（アプリケーション）：Magic xpa の .ecf ファイルを呼び出します。
- O（OS コマンド）：OS のコマンド（バッチファイルや EX ファイル E）を実行します。
- M（サブメニュー）：サブメニュー郡のヘッダ
- L（ライン）：メニューの区切り線

後者の 3 つは、一目瞭然です。しかし、最初のタイプ（***.ecf*** ファイルの呼び出し）は、便利なツールアプリケーションを定義することで開発時に便利に利用することができます。***.ecf*** ファイルのオープン前やクローズ後に実行するマクロ処理を実行するスクリプトを定義することができます。

パラメータの構文はメニュータイプによって異なります。タイプによっては不要なパラメータもあります。

メニュー	構文
アプリケーション （読みやすくするため改行しています。）	< メニュー名 > = A, < タイトル >, < 親のメニュー >, <ECF ファイル >, < アクセスキー >, < 事前処理ファイル >, < 事後処理ファイル >, < アイコン >
OS コマンド	< メニュー名 > = O,< タイトル >,< 親のメニュー >,< コマンド >, < アクセスキー >,,,< アイコン >
サブメニュー	< メニュー名 > = M,< タイトル >,< 親のメニュー >,,,,,
区切り	< メニュー名 > = S,,< 親のメニュー >,,,,,

以下はメニュー項目の概要です。詳細は、『リファレンスヘルプ』を参照してください。

メニューパラメータ		
メニュー名	メニュー定義の名前	ここで指定される名前は、サブメニューの定義の際に親メニューとして指定することができます。実際に表示される名前でありません。
メニュータイプ	• A = アプリケーション • O = OS コマンド • M = サブメニュー • S = 区切り	
タイトル	実際に表示されるメニューの名前	これは、メニュー上に表示されるものです。 ショートカットキーを指定するために '&' を含めることができます。
親のメニュー	親メニューのメニュー名を指定します。	先頭のメニューは空白を指定します。 親メニューのメニュータイプは、M にしてください。

メニューパラメータ		
ECF ファイル／コマンド	• ECF:Magic xpa のキャビネットファイルを指定します。 • OS コマンド：有効な OS のコマンド	メニュータイプにもとづいて、*.ecf* ファイルか OS のコマンドを指定します。
アクセスキー	*Ctrl+1* などのショートカットキー組み合わせ。	
事前処理ファイル	ECF または OS コマンドが処理される前に実行されるコマンドが定義されたスクリプトファイル	コマンドファイルは、いろいろなコマンドを定義できます。詳細は、『リファレンスヘルプ』を参照してください。
事後処理ファイル	ECF または OS コマンドの処理後に実行されるコマンドが定義されたスクリプトファイル	
アイコン	ImageFor = B ToolNumber = 60 ToolGroup 1	メニュー上に表示されるアイコン。 構文で指定される内容は、Magic xpa のメニューリポジトリで使用される内容と同じです。

外部ツールを追加する

次に、外部ツールを現在のメニューに追加する手順を説明します。ここでは、レジストリエディタを追加してみます。

1. 現在のプロジェクトをクローズします。
2. テキストエディタを使用して Magic.ini ファイルを開きます。
3. ［TOOLS_MENU］セクションに移動します。
4. 上記で説明されている構文に従って OS コマンドとして Regedit を追加します。

```
[TOOLS_MENU]
DDLBDR = A,DDLMaker(&D),,Add_On¥DDLMaker¥PVDDLMaker.ecf,,,Add_On¥DDLMaker¥DDLMaker_suf.opr,ImageFor = B +
ToolNumber = 60 ToolGroup = 1 ToolTip = ″ﾃﾞｰﾀ辞書を作成します.″
String Extractor = A, 文字列抽出 (&T),,Add_On¥StringExtractor¥StringExtractor.ECF,,,,Ima+
+ geFor = B ToolNumber = 283 ToolGroup = 1 ToolTip = ″多言語辞書用の文字列を抽出します.″
Menu = O, レジストリの編集 ,,regedit.exe,F12,,,
```

この例では、*F12* をショートカットキーとして定義しています。

5. **Magic.ini** ファイルを保存し、**Magic xpa Studio** を再起動します。これで、追加されたメニューが表示されます。このメニューを選択すると Regedit が起動されます。

自動的に外部プロセスを実行するには

Magic xpa を起動する時に、自動的にコマンドを実行させることができます。これらのコマンドは、Magic xpa の内部でマクロコマンドとして実行されます。このコマンドでは、リポジトリの入出力やテキスト入力などを行うことができます。

外部プロセスを実行するには、２つの処理が必要です。

- 外部コマンドファイルを作成する
- Magic xpa の起動時にコマンドファイルが実行されるように設定する

コマンドファイルを作成する

コマンドファイルには独自の構文でコマンドが設定されています。このコマンド構文は、**ユーザ定義開発メニュー**で使用される**事前／事後処理ファイル**（「外部ツールを Magic xpa Studio に追加するには」（688 ページ）を参照）と同じです。

構文の詳細は『リファレンスヘルプ』を参照してください。Magic xpa 内で同じ処理を繰り返し実行させたい場合、基本的にはコマンドをコピーすることで対応できます。

以下は、コマンドの概要です。

- Export：プロジェクト全体やその中の一部のオブジェクトを出力します。例えば真夜にすべてのプロジェクトをバックアップしたい場合、この機能が利用できます。
- Import：プロジェクト全体やその中の一部のオブジェクトを入力します。
- ECF：プロジェクトからキャビネットファイルを作成します。すべてのプロジェクトのキャビネットファイルを 1 つのバッチ処理で作成したい場合は、この機能が利用できます。
- Getdef：SQL テーブルの定義取得を行います。
- Project：プロジェクトを開きます。
- Simulate：キーボード処理をシミュレートします。

また、プロジェクト内のプログラムやモデルがいくつ定義されているとかといった、プロジェクトに関する情報を参照するためのグローバル変数を使用することもできます。

自動的に実行するコマンドファイルを設定する

コマンドファイルが作成されたら、自動的にそれを実行するための環境設定が必要です。コマンドファイルによって、オープンするプロジェクトを（リポジトリ入力で）作成することもできるため、プロジェクトの外に定義することになります。つまり、Magic.ini ファイルで設定されます。

以下の行を［MAGIC_ENV］セクションに追加します。

```
AutomaticProcessingSequenceFile=%Path%Filename.txt
```

コマンドファイルの指定には、論理名を使用することができます。

バッチ内に自動処理を設定する

アプリケーションをオープンすることなく、バックグラウンドで処理を実行させることもできます。例えば、アプリケーションのバックアップのためにバックグラウンドモードでリポジトリ出力を行うことができます。この場合に、Magic.ini の [MAGIC_ENV] セクションに以下のように設定します。

```
AutomaticProcessingMode= B
```

これにより Magic xpa 起動時に自動的に処理が実行され、自動的に処理が終了します。

Magic xpa ライセンスの使用状況を確認するには

現在のライセンスの利用状況を確認するには、以下のようにします。

Support サブフォルダ内のある MGStaions ユーティリティを使用することで、ライセンスの利用可能数やライセンスが使用数を確認することができます。また利用しているユーザー一覧を取得することができます。

構文：

```
MGStations <license> <license file>
```

例：

```
"C:¥Program Files¥Magic xpa¥Client¥mgstations" MGCSRT4 "C:¥Program
Files¥xpa¥Client¥license.dat"
```

指定されたライセンスが２つの PC で使用されている場合、以下のような結果が返ります。

```
MGCSRT3 (MGCSRT, E171C161A4601B030AC5, 20 users) ... please wait ...
1 : Server1 : 1 users
2 : Server2 : 1 users
MGCSRT, E171C161A4601B030AC5: 2 users consumed
```

1 対 1 の関係を持つデータ ソースの参照プログラムを作成するには

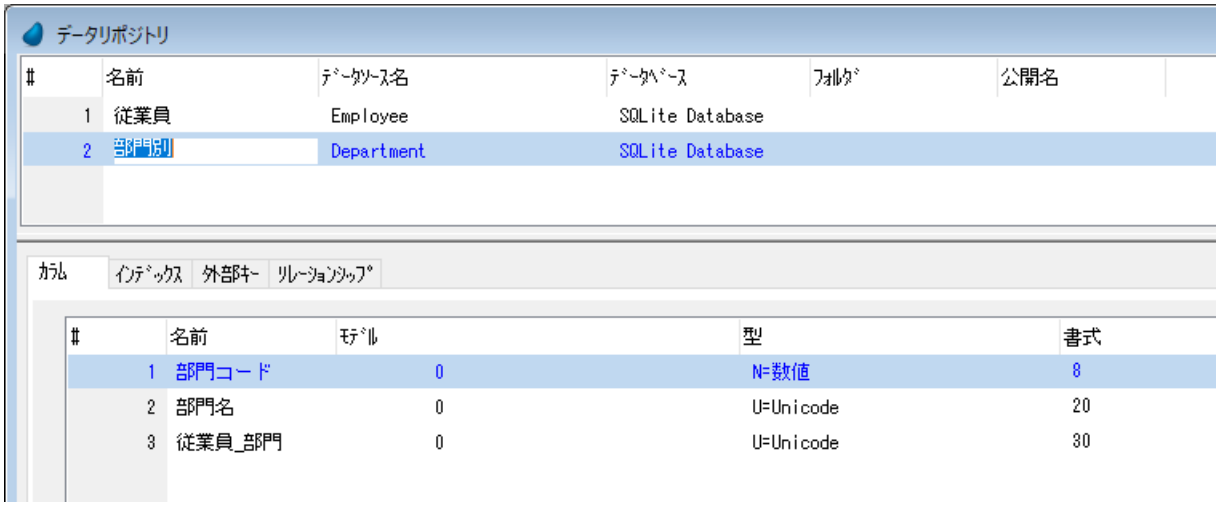
2 つのデータベーステーブルを 1:1 の関係でリンクし、プログラムを自動生成することができます。［ オプション］メニューの ［リレーションシップ プログラムの生成］(Ctrl+R) を使用して、テーブルを含むプログラムを自動的に生成することができます。

指定された手順に従って、テーブルのリレーションシップを持つプログラムを自動的に生成します。

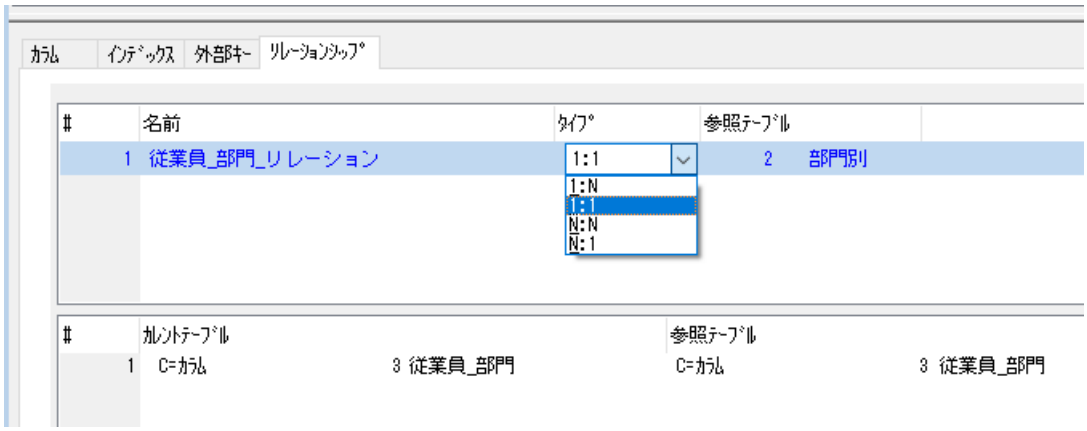
1. 次のように 2 つのテーブルを定義します。

従業員テーブル

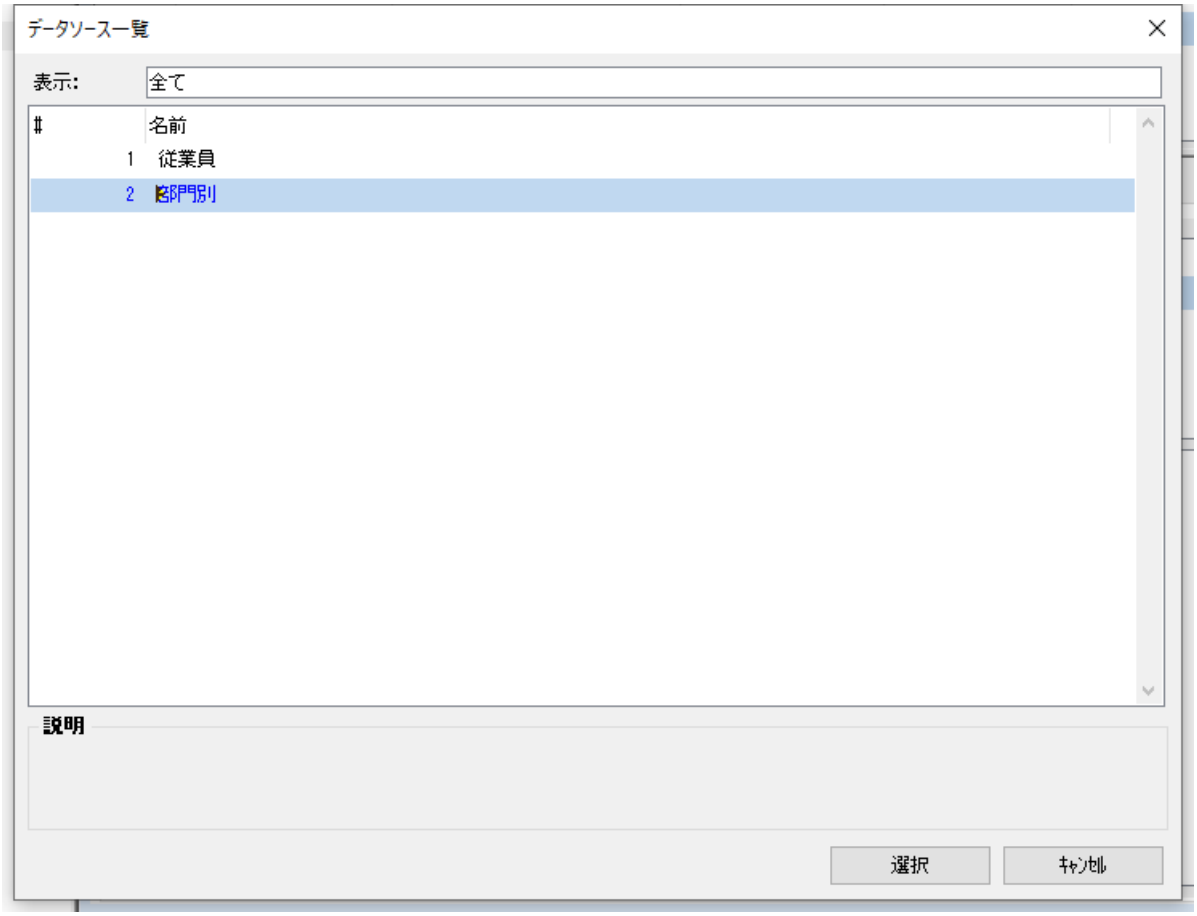
部門別テーブル



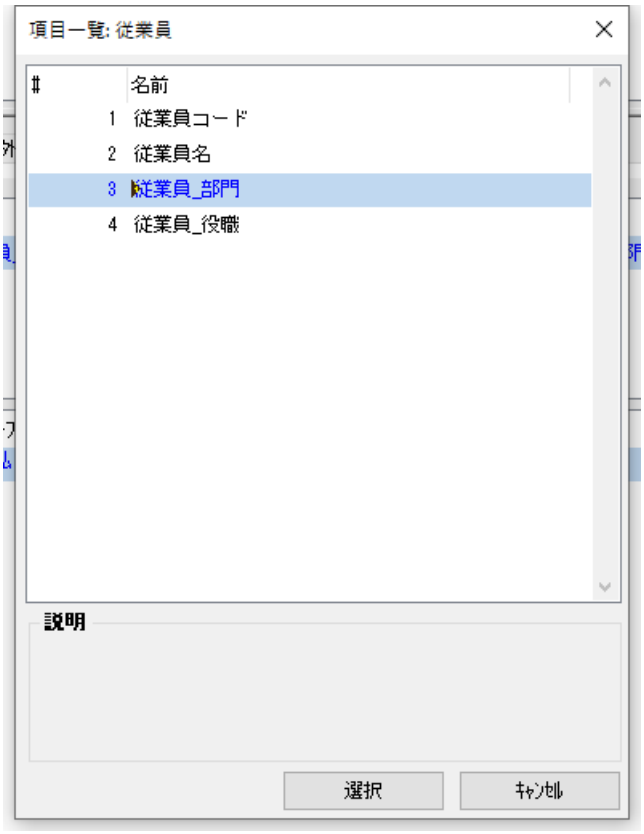
2. " 従業員テーブル " 上にカーソルを置きます。
3. ［リレーションシップ］タブをクリックします。ここでは、リレーションシップ名を入力する必要があります。
4. 一行作成（F4）します。
5. " 従業員 _ 部門 _ リレーション " というようなリレーションシップ名を入力します。
6. ［タイプ］のコンボボックスから、「タイプ」を「1:1」として選択します。



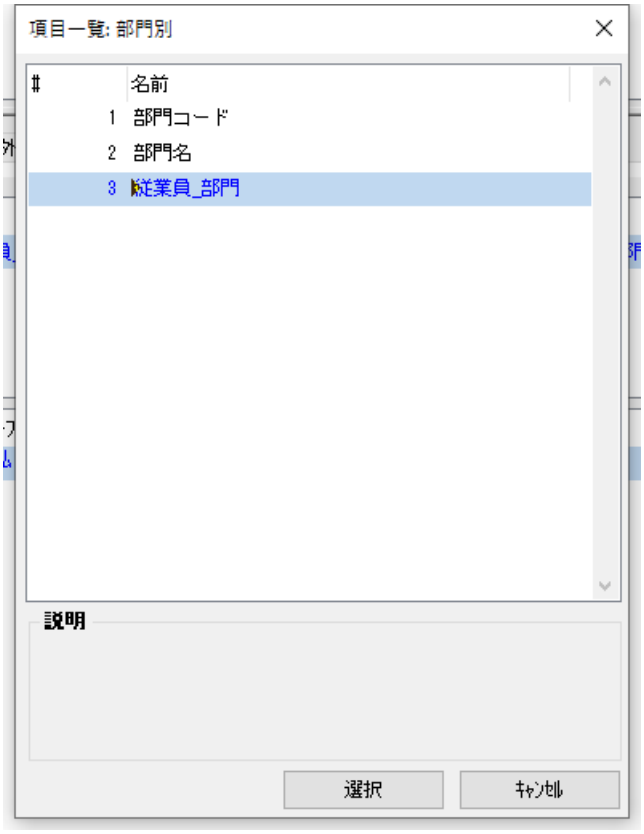
7. 参照テーブル番号をズームします。
8. 現在のテーブルにリンクしたい参照テーブルを選択します。



9. 選択をクリックします。
10. 現在のテーブルをクリックします。ここで、参照されたテーブルと現在のテーブルをリンクするための基準を設定する必要があります。
11. 下のテーブルに移動して［カレントテーブル］でカラム / 変数のオプションから「カラム」を選択します。(テーブルからカラムを選択しても、メインプログラムから変数を選択しても構いません)
12. カラム番号からズームして、現在のテーブルのすべてのカラムのリストを表示します。
13. #3 のカラム、「従業員 _ 部門」を選択します。

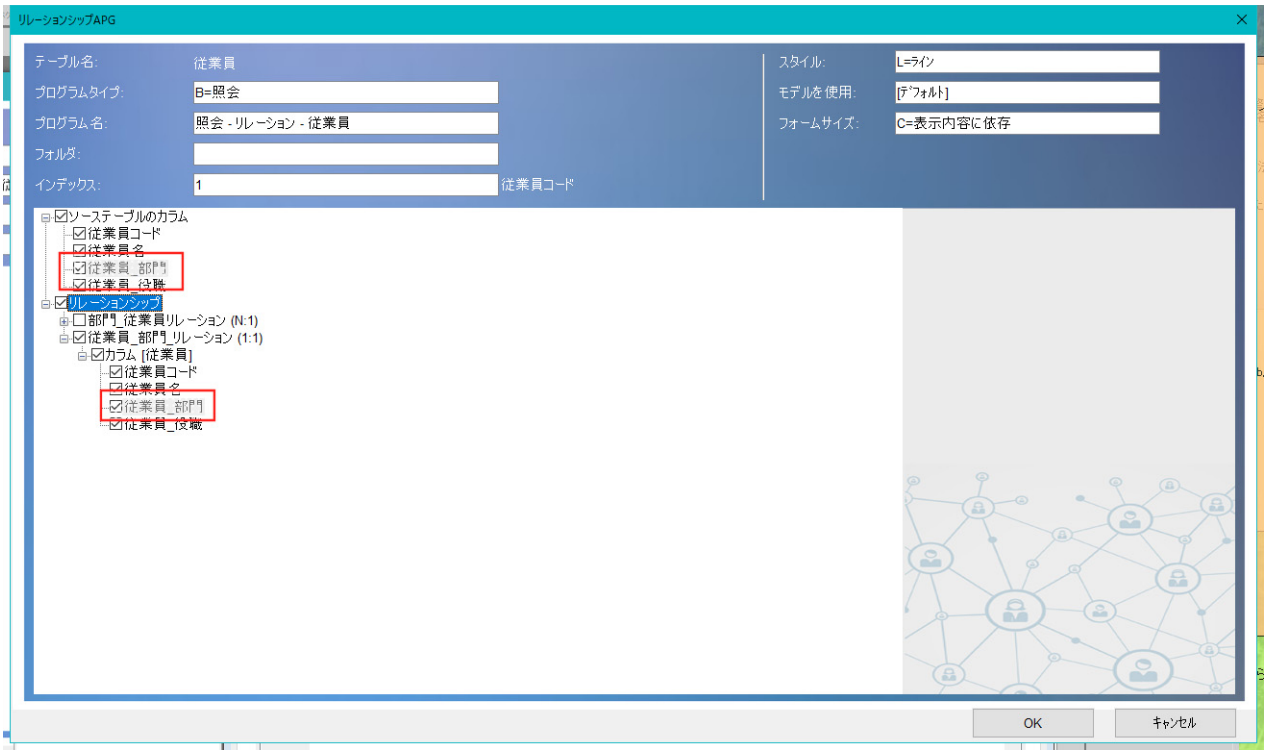


14. [選択] をクリックします。
15. 同様に、参照先テーブルからカラム 「従業員 _ 部門」を選択します。

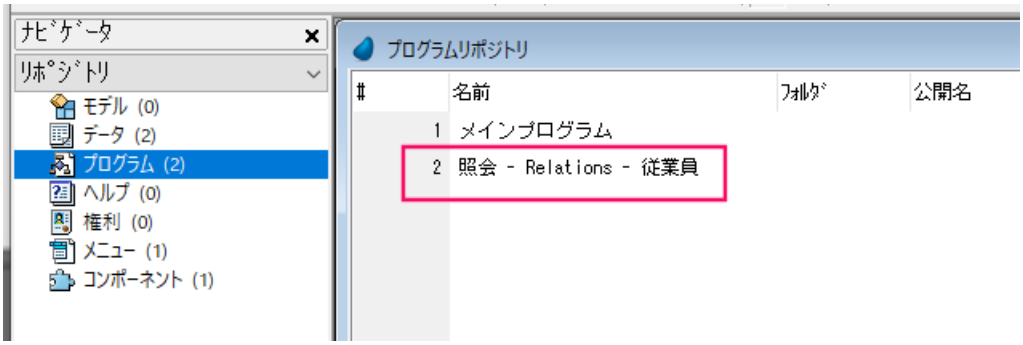


16. [選択] をクリックします。
17. 次に、「従業員テーブル」にパークします。

18. Ctrl+R キーを押下して、" リレーションシップ用の APG" を表示します。［APG リレーションシップ］ダイアログが表示されます。折りたたまれたツリー構造の形でリレーションシップを表示させることができます。列、リレーションシップ、および基準を表示するには、ノードを展開する必要があります。" 従業員 _ 部門 " は、現在のテーブルと参照テーブルがリンクされている共通のカラムであることに注意してください。



19. ダイアログボックスのすべてのオプションをデフォルトのままにしておきます。
20. 参照テーブル側のテーブル名（例：「従業員 _ 部門リレーション (1:1)」をチェックします。
21. 表示フォームに反映するカラムをチェックします。
22. ［OK］をクリックします。
23. ［プログラム］リポジトリ を開きます。プログラムのリストの最後に「照会 - Relations - 従業員」というプログラムが表示されます。



24. プログラムのデータビューを開くと、以下のように照会リンクが定義された従業員テーブルをメインソースとしたプログラムが生成されていることを確認することができます。?

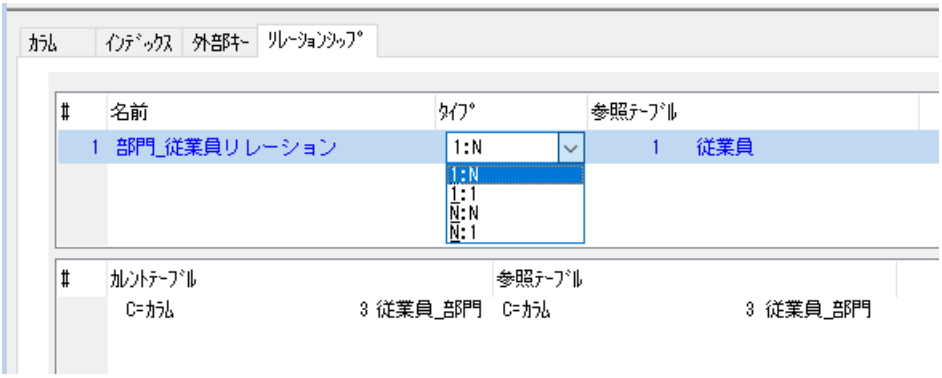
Since version: 4.7

1 対 N の関係を持つデータ ソースの参照プログラムを作成するには

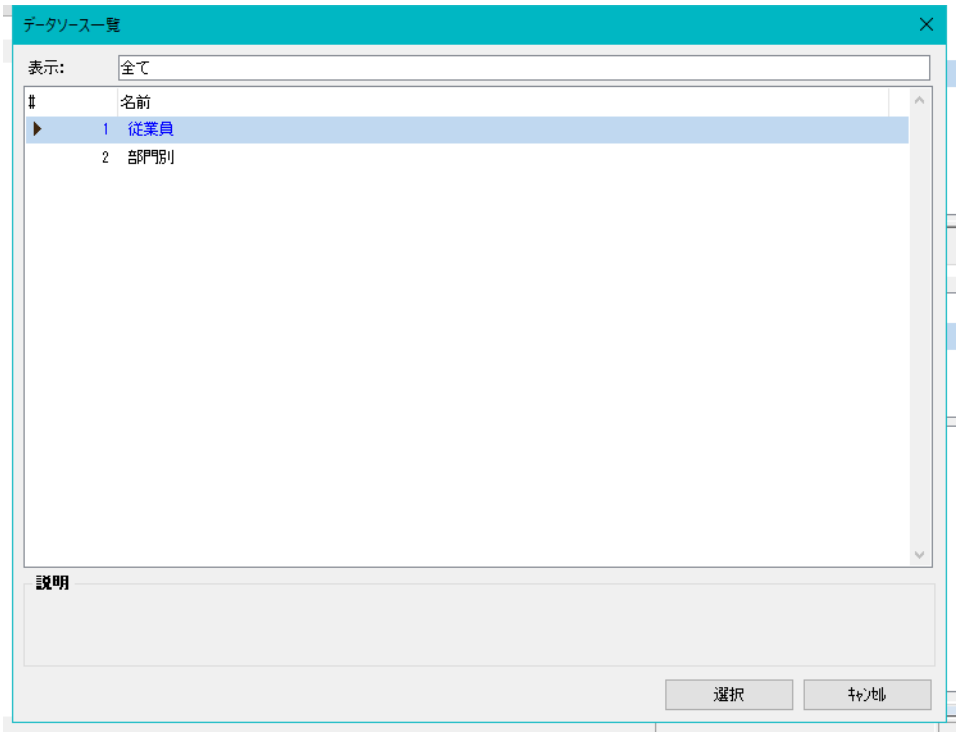
2 つのデータベーステーブルを 1:N の関係でリンクし、プログラムを自動生成することができます。［オプション］メニューの ［リレーションシップ プログラムの生成］(Ctrl+R) を使用して、テーブルを含むプログラムを自動的に生成することができます。

指定された手順に従って、テーブルのリレーションシップを持つプログラムを自動的に生成します。

1. 「1 対 1 の関係を持つデータ ソースの参照プログラムを作成するには」（692 ページ）と同じ 2 つのテーブルを使用します。
- 従業員テーブル
- 部門別テーブル
2. " 部門別テーブル " 上にカーソルを置きます。
3. ［リレーションシップ］タブをクリックします。ここでは、リレーションシップ名を入力する必要があります。
4. 一行作成（F4）します。
5. " 部門 _ 従業員 _ リレーション " というようなリレーションシップ名を入力します。
6. ［タイプ］のコンボボックスから、「タイプ」を「1:N」として選択します。

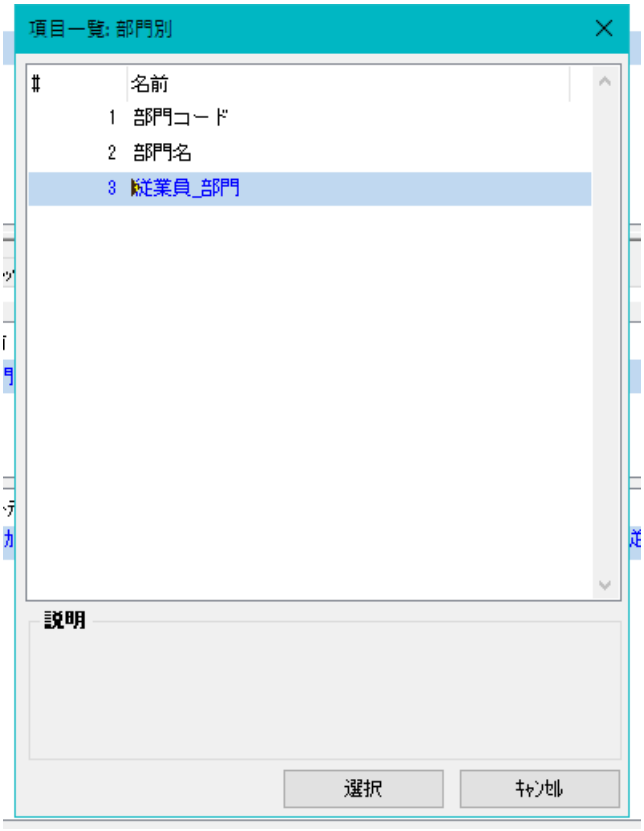


7. 参照テーブル番号をズームします。
8. 現在のテーブルにリンクしたい参照テーブルを選択します。

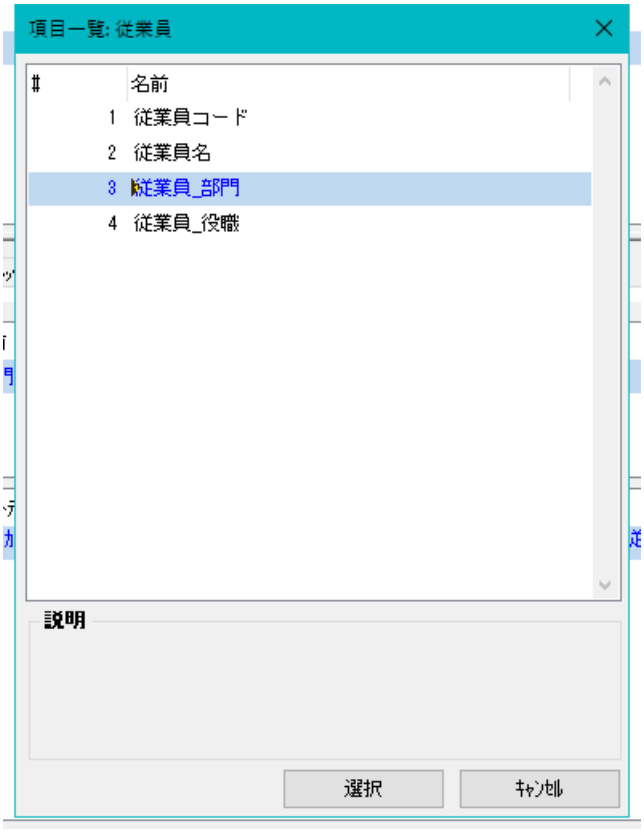


9. 選択をクリックします。
10. 現在のテーブルをクリックします。ここで、参照されたテーブルと現在のテーブルをリンクするための基準を設定する必要があります。
11. 下のテーブルに移動して［カレントテーブル］でカラム / 変数のオプションから「カラム」を選択します。(テーブルからカラムを選択しても、メインプログラムから変数を選択しても構いません)
12. カラム番号からズームして、現在のテーブルのすべてのカラムのリストを表示します。

13. #3 のカラム、「従業員 _ 部門」を選択します。

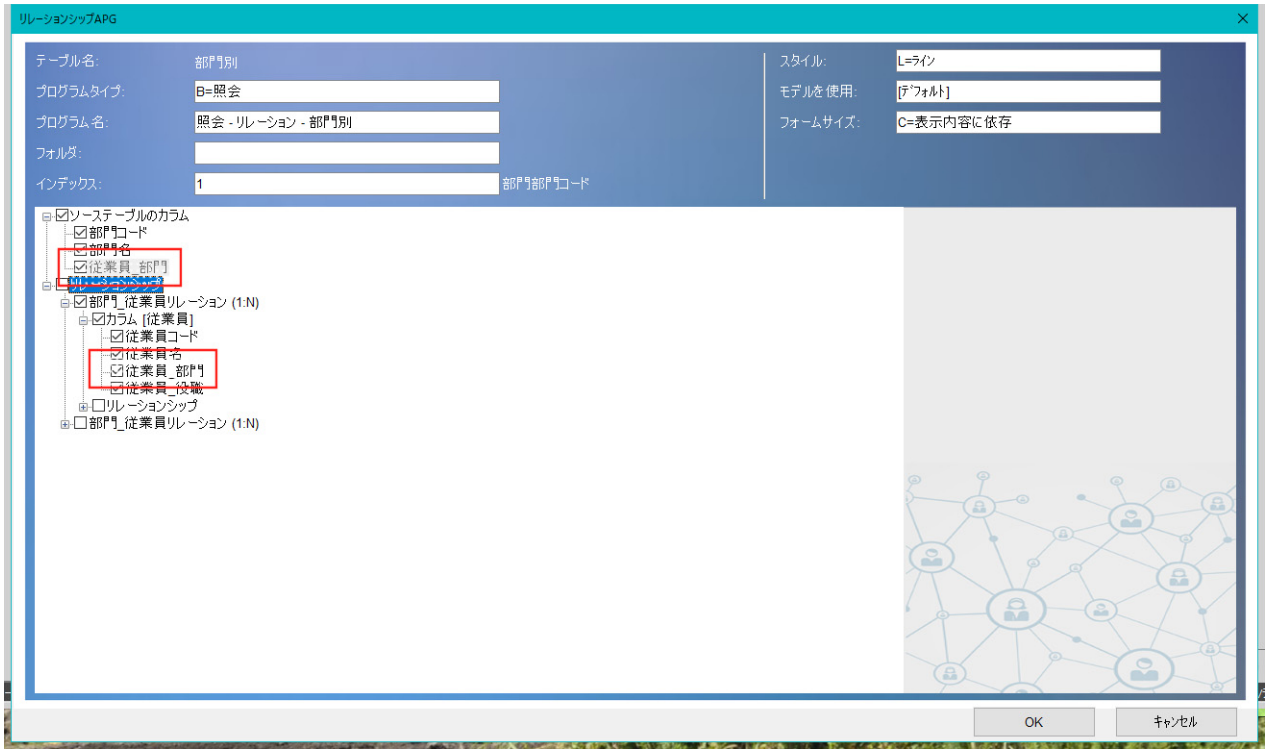


14. [選択] をクリックします。
15. 同様に、参照先テーブルからカラム 「従業員 _ 部門」を選択します。



16. [選択] をクリックします。
17. 次に、「従業員テーブル」にパークします。
18. Ctrl+R キーを押下して、" リレーションシップ用の APG" を表示します。[APG リレーションシップ] ダイアログが表示されます。折りたたまれたツリー構造の形でリレーションシップを表示させることができます。列、リレーション

シップ、および基準を表示するには、ノードを展開する必要があります。" 従業員 _ 部門 " は、現在のテーブルと参照テーブルがリンクされている共通のカラムであることに注意してください。

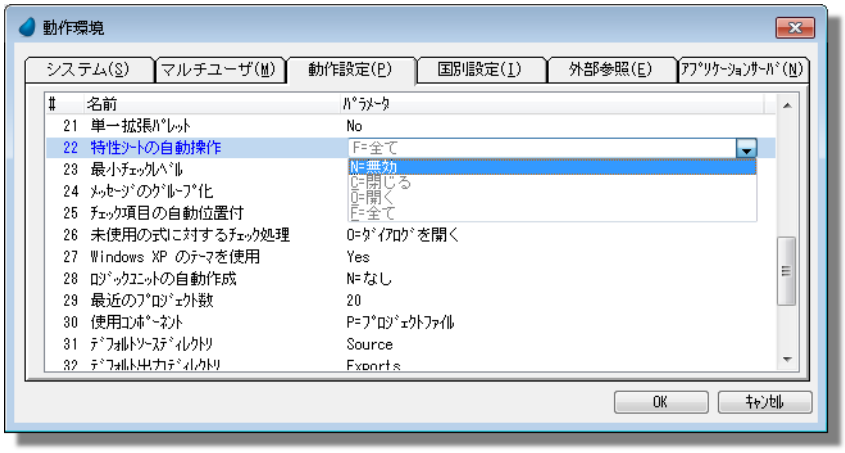


19. ダイアログボックスのすべてのオプションをデフォルトのままにしておきます。
20. 参照テーブル側のテーブル名（例：「部門 _ 従業員リレーション (1:N)」をチェックします。
21. 表示フォームに反映するカラムをチェックします。
22. ［OK］をクリックします。
23. ［プログラム］リポジトリ を開きます。プログラムのリストの最後に「照会 - Relations - 部門別」というプログラムが表示されます。
24. プログラムのデータビューを開くと、以下のように［サブフォーム］コントロールで子タスクを表示するプログラムが作成されていることを確認することができます。?

Since version: 4.7

第 32 章 : 開発環境

特性シートやナビゲータペインが自動的に表示されないようにするには



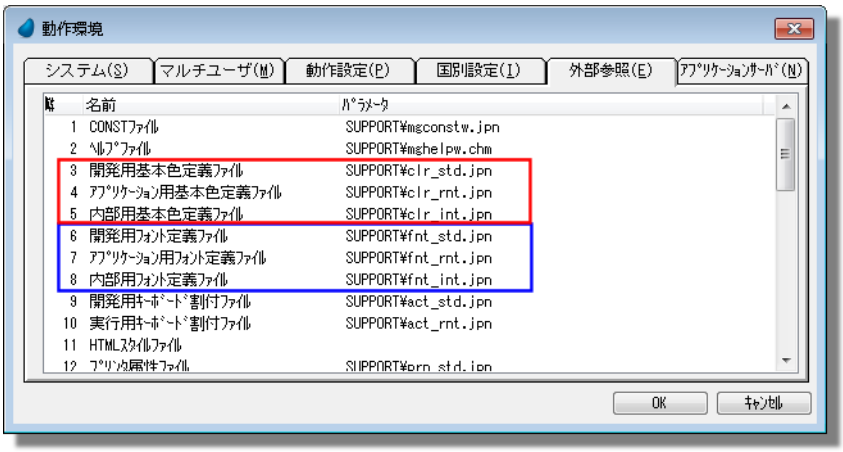
特性シートの表示方法を変更するには、**オプション→設定→動作環境→動作設定**の**特性シートの自動操作**で行います。ここには、以下の４つのオプションがあります。

- ***N= 無効***……特性シートは自動的に開いたり閉じたりしません。
- ***C= 閉じる***……関連しない特性シートが開いている場合、自動的に閉じます。ただし、自動的には開きません。
- ***O= 開く***……関連する特性シートが自動的に開きます。ただし、自動的には閉じません。
- ***F= 全て***……関連する特性シートが自動的に開き、関連しない場合は自動的に閉じます。

特性シートが自動的に開かないようにし、更に自動的に閉じるようにしたい場合は、***C= 閉じる***を選択します。

また、特性シートの手動で操作したい場合は、***N= 無効***を選択します。

使用するフォントや色を変更するには



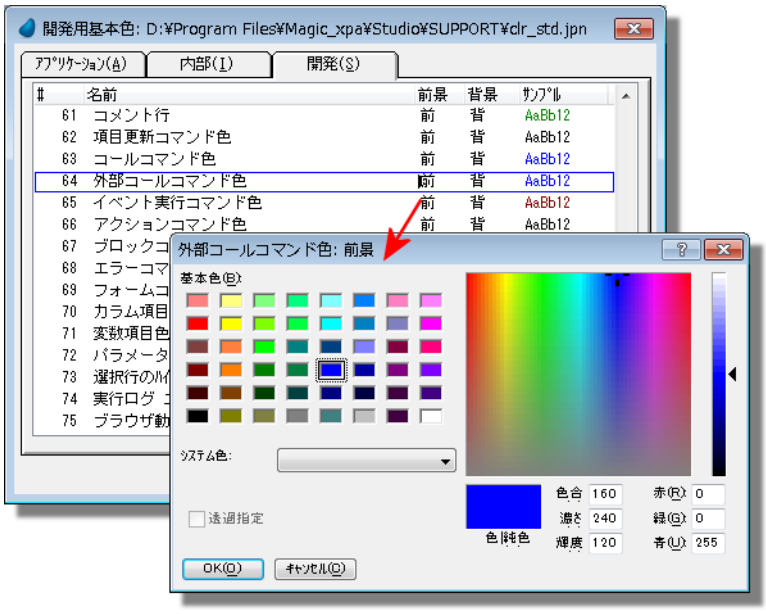
Magic xpa では、基本色定義とフォント定義は３つの別個のファイルに分けて定義されます。

- 開発用：開発用ウィンドウで有効な色とフォント
- アプリケーション用：アプリケーションの作成で使用される色とフォント
- 内部用:実行時に Magic xpa が使用する色とフォント(ポップアップのダイアログやメニューなど Magic xpa の基本構造の一部となっている項目が対象)

これらの定義ファイルのファイル名やパス名は、Magic.ini に指定されます。上の図に示されるように、これらの設定は、**オプション→設定→動作環境→外部参照タブ**で行うことができます。

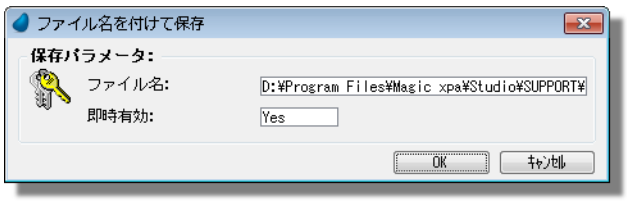
作業内容や、ウィンドウの解像度やサイズに合わせて開発用の基本色やフォントをカスタマイズすることができます。例えば、使用する PC がノート型かデスクトップ型のどちらかによって色とフォントを切り替えることで作業し易くなる場合があります。

開発用基本色を変更する



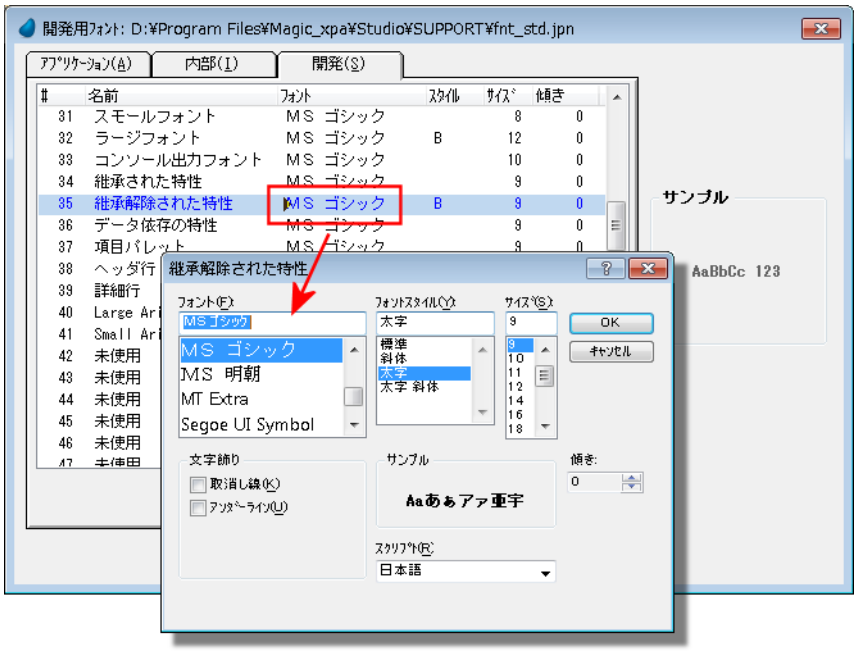
1. **オプション→設定→基本色**を選択し、**基本色**テーブルを開きます。
2. **開発**タブを**クリック**します。
3. 変更したい値にカーソルを置きます。例えば、**処理コマンド色**を変更することでプログラムを読みやすくすることができます。
4. **前景色**（**前景**カラム、文字の色を変更）や**背景色**（**背景**カラム、文字の周りの色を変更）から**ズーム**して、Windows の**カラーパレット**を開きます。
5. **システム色**を選択した場合、OS から引き継がれた色が設定されます。
6. **システム色**を空白にした場合、**基本色**から選択したり、カスタマイズした色を設定することができます。

7. 色を設定したら OK をクリックします。
8. 基本色テーブルを終了すると、保存確認ダイアログが表示されます。即時有効を *Yes* に設定し、OK をクリックします。
9. これで、設定された色が有効になります。

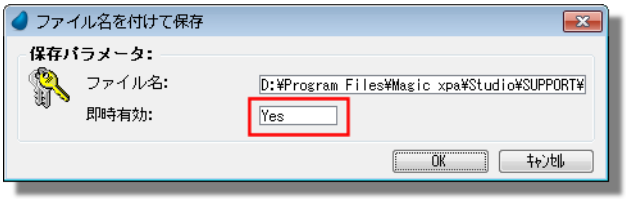


設定された色の情報は、Magic.ini で指定された定義ファイル内に保存され、直ちに有効になります。

開発用フォントを変更する

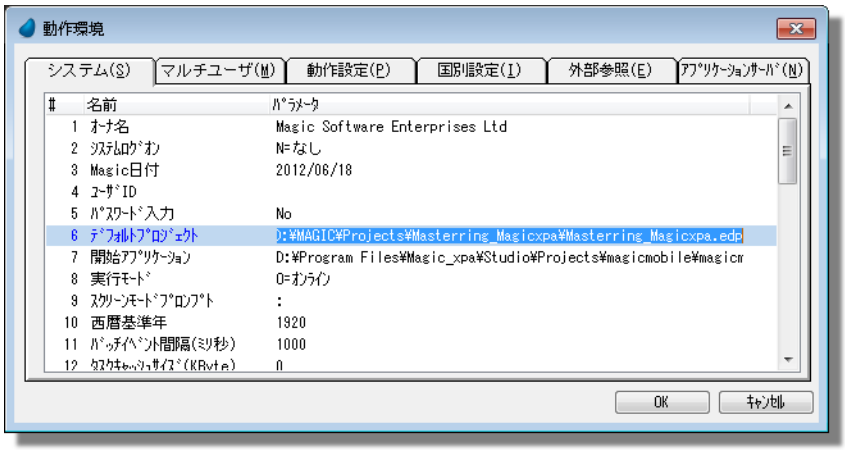


1. オプション→設定→フォントを選択し、フォントテーブルを開きます。
2. 開発タブをクリックします。
3. 変更したい値にカーソルを置きます。
4. フォントカラムからズームして、フォントダイアログを開きます。ここからフォントやスタイル、サイズを選択することができます。選択された内容にもとづいてサンプル欄に例が表示されます。
5. フォントを設定したら OK をクリックします。
6. フォントテーブルを終了すると、保存確認ダイアログが表示されます。即時有効を *Yes* に設定し、OK をクリックします。
7. これで、設定されたフォントが有効になります。



設定されたフォントの情報は、Magic.ini で指定された定義ファイル内に保存され、直ちに有効になります。

Magic xpa Studio 起動時に、自動的にプロジェクトを読み込ませるには



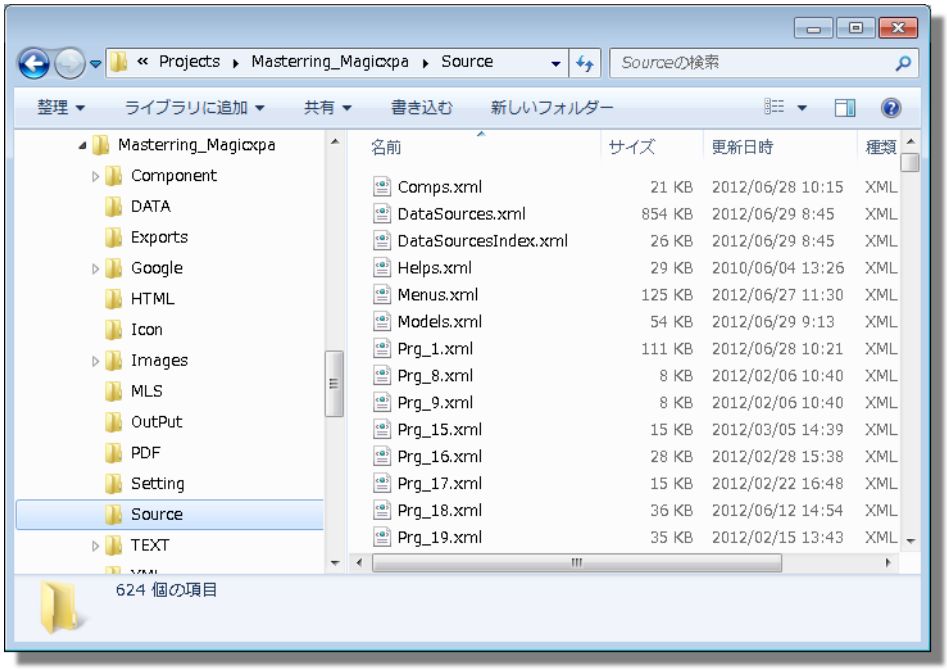
１つのプロジェクトのみを開発している場合、**Magic xpa Studio** 起動時に自動的にそのプロジェクトのみが読み込まれるようにすることができます。

Magic.ini に、デフォルトプロジェクトを設定することで可能になります。設定方法は以下の通りです。

1. **オプション→設定→動作環境→システムタブ**を選択し、**デフォルトプロジェクト**に移動します。
2. ここから**ズーム**してデフォルトプロジェクトとして使用する ***..edp*** ファイルを選択します。
3. OK を**クリック**します。

これで **Magic xpa Studio** を起動すると、指定されたプロジェクトが自動的にオープンされます。

プロジェクトのソースファイルの格納場所を変更するには



Magic プロジェクトのソースコードは一連の XML ファイルとして保持されます。デフォルトでは、ここに示されているように、**Source** と呼ばれるサブディレクトリ内に格納されます。

ソースファイルの格納ディレクトリは、常に ***.edp*** ファイルに関連した場所に定義されます。この例では、***.edp*** ファイルは **C:¥MAGIC¥Projects¥Mastering_xpa** にあるため、ソースディレクトリは、**C:¥MAGIC¥Projects¥Mastering_xpa¥Source** になります。

既存の .edp のソースディレクトリ定義を変更する

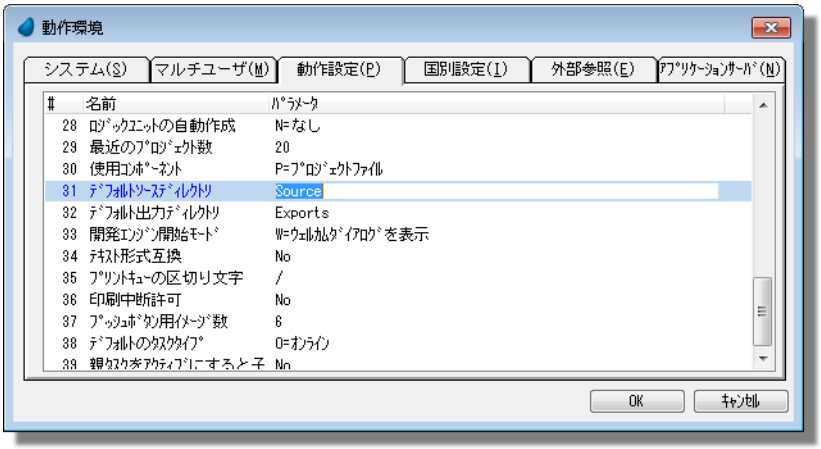
.edp ファイルが作成されると、ソースファイルの位置はその中にコード化されます。この ***.edp*** ファイルは XML ファイルで、編集することができます。この例では、以下の用に設定されています。

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Application>

  <Project>
    <ProjectName val="Masterring_Magicxpa"/>
    <VCActive val="N"/>
    <WorkOffline val="N"/>
    <SourceDirectory val="Source"/>
    <ExportsDirectory val="Exports"/>
    <GUID val="{A0275EFF-1939-49CC-970A-98BA38F752A1}"/>
  </Project>

</Application>
```

この設定を変更することで、ソースファイルの格納先を変えることができます。

デフォルトのソースディレクトリを変更する

今後作成するプロジェクトに対するデフォルトのソースディレクトリを変更することができます。この設定は、**Magic.ini** ファイルの内の DefaultSourceDir の設定で行います。



この内容は、**設定→オプション→動作環境→動作設定タブ**の**デフォルトソースディレクトリ**でも変更することができます。変更された設定内容は、**Magic xpa Studio** が再度起動された時点で有効になります。

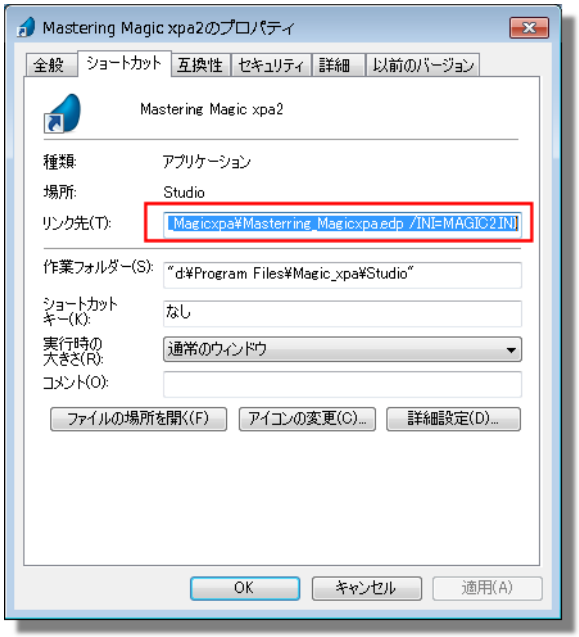
Magic.ini を指定して起動するには

デフォルトでは、***.edp*** ファイルをクリックして **Magic xpa Studio** を起動した場合、***.edp*** ファイルと同じディレクトリに置かれた *Magic.ini* と呼ばれるファイルを使用します。このファイルが存在しない場合、**Magic xpa Studio** のインストールディレクトリ内に置かれた **Magic.ini** ファイルが使用されます（**Magic xpa Studio** の起動メニューをクリックした場合は、Magic xpa のインストールディレクトリ内のファイルが使用されます）。

しかし、任意の場所に格納されている **Magic.ini** を使用して起動させることもできます。この方法は、デフォルトのログイン環境と異なるユーザ（評価担当者や開発者など）用に個別のログイン環境を提供する場合などに使用されます。

この場合、以下のように設定します。

異なる Magic.ini ファイルを使用する



1. 起動用のアイコンを作成またはコピーします。
2. リンク先には、使用したい Magic.ini を読み込むように指定します。この場合、以下の構文で指定できます。

```
<xpa.exe file> /INI=<ini file name>
```

例えば以下のように設定できます。

```
"D:¥Program Files¥Magicxpa¥Studio¥MGxpaStudio.exe" /
ini=D:¥MAGIC¥Projects¥Masterring_xpa¥magic2.ini
```

バッチファイルやその他のスクリプトを使用して指定することもできます。

Magc.ini の環境情報を追加指定するには

デフォルトでは、***.edp*** ファイルをクリックして **Magic xpa Studio** を起動した場合、***.edp*** ファイルと同じディレクトリに置かれた *Magic.ini* と呼ばれるファイルを使用します。このファイルが存在しない場合、**Magic xpa Studio** のインストールディレクトリ内に置かれた **Magic.ini** ファイルが使用されます。

しかし、Magic.ini の特定の項目だけを変更したり追加して **Magic xpa Studio** を起動することができます。例えば特定の基本色定義ファイルを指定したり、評価のために別のファイルを使用する場合にこの方法が利用できます。

この場合、以下のように設定します。

Magic.ini の設定をオーバライドする

最初に、変更したい情報だけを格納したファイルを作成します。各項目の先頭には、/（スラッシュ）を設定する必要があります。

例えば、以下のように設定します。

```
/[MAGIC_ENV]InputPassword=N
/[MAGIC_ENV]User = Supervisor
/[MAGIC_ENV]Password =
/[MAGIC_ENV]StartApplication=1
/[MAGIC_ENV]ColorDefinitionFile=usrclr2.jpn
/[MAGIC_LOGICAL_NAMES]TEMP=C:¥temp¥
/[MAGIC_LOGICAL_NAMES]Testing=Y
```

この例では、ユーザ ID やパスワードなどのデフォルト設定のいくつかが無効にしています。プロジェクトの開発時には、これによって時間の節約になります。

スラッシュが付加された各アイテムは、[]（ブラケット）内に Magic.ini のセクション名を指定します。

オーバライドを使用する

次に、**Magic xpa Studio** 起動時にこの設定ファイルを読み込ませる指定が必要です。以下の構文で指定します。

```
<xpa.exe file> @<Override File Name>
```

例えば以下のように設定できます。

```
"C:¥Program Files¥xpa¥Studio¥uniStudio.exe" @C:¥xpa_Projects¥Testing.ini
```

これで、Testing.ini で設定された値によって、Magic.ini に設定されている値がオーバライドされます。

第 33 章 : Web サービス（コンシューマ）

注 : Magic xpa 3.3 から Systinet Server for Java がインストールされなくなりました。以下の項目は、Systinet Server for Java が必要になりますが下位互換のために有効になっています。利用する場合は、手動で Systinet Server for Java をインストールしてください。

- 設定／サービス／ Web サービス特性
- ［外部コール］処理コマンド
- SoapSpy

Systinet Server for Java を利用して Web サービスを呼び出す処理については、Magic xpa 3.2 以降のマニュアルを参照してください。

簡易的に Web サービスを呼び出すには

Magic xpa が Web サービスを呼び出す標準の方法は、Systinet Server for Java をフレームワークとして使用することです。しかし、**外部コール**処理コマンドの Web サービスライトオプションを利用することで、より簡単に呼び出すこともできます。この方法は、Magic eDeveloper Ver9 と同じ方法です。この場合は、Java の実行環境や Systinet Server for Java をインストールする必要がありません。ただし、SOAP1.2 や WS-Security などを利用することができません。

コール Web サービスライトを定義する

Country information という Web サービスを呼び出して、ISO の国コードから国名（英語）を求めるプログラムを作成します。この作成作業にはインターネットに接続できる環境が必要です。

オンラインプログラムの作成

1. **プログラム**リポジトリにオンラインタスクを作成します。
2. 以下のような変数項目を定義します。

名前	型	書式
Country Code	A= 文字	UU
Result	A= 文字	200
XML To Send	B=BLOB	
XML Result	B=BLOB	
Success?	L= 論理	5
Error Message	B=BLOB	

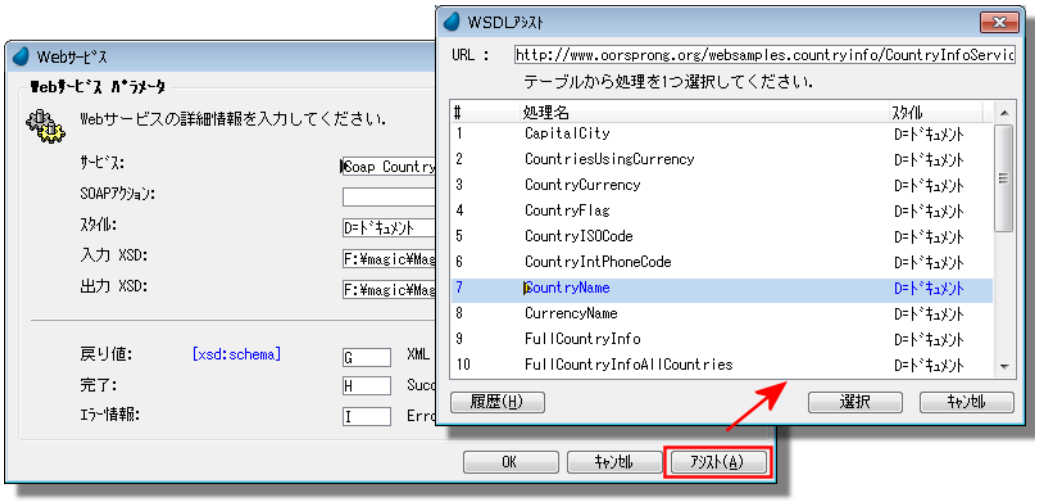
3. メインフォームに変数（Country Code と Result）を配置します。
4. **イベント**テーブルを開いて（*Ctrl+U*）、Start というイベントを作成します。**トリガタイプ**は、***なし***に設定します。OK をクリックして**イベント**テーブルを閉じます。
5. **ロジック**エディタを開き（*Ctrl+2*）、*Ctrl+H* を押下して**イベント**ロジックユニットを追加します。
6. 1 行追加して、**コール Web サービスライト**処理コマンド（**外部コール／*L=Web サービスライト***）を定義します。
7. L=Web サービスライトと表示されている次のカラムに移動して、ズームします。**Web サービス**ダイアログが開きます。
8. 新規の場合はさらに **WSDL アシスト**ダイアログが表示されます。ここで、既存の WSDL を指定することで Web サービスを指定することができます。

WSDL アシスト

1. ここで、既存の WSDL を指定することで Web サービスを指定することができます。
2. **URL** 欄に WSDL の URL を指定します。ここでは、以下のように入力して、*Enter* を押下すると、Web サービスとして利用可能な処理が一覧表示されます。
   http://www.oorsprong.org/websamples.countryinfo/CountryInfoService.wso?WSDL

（上記の URL は、外部のサービスですのでアクセスできなくなる場合があります。）

3. 利用したい処理にカーソルを置き、**選択**ボタンをクリックします。ここでは、CountryName を選択します。

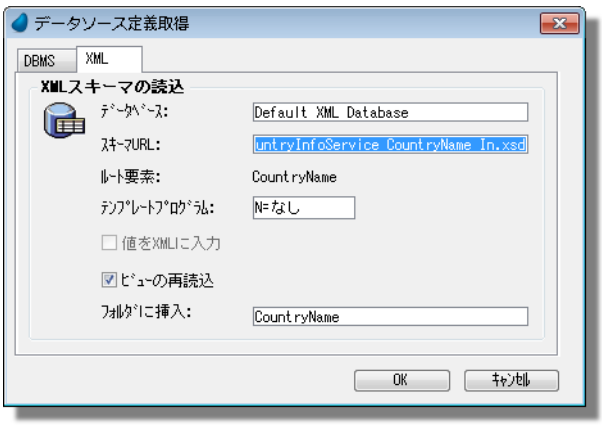


4. **WSDL アシスト**ダイアログが閉じ、**Web サービス**ダイアログの上位パラメータに値が設定されます（**入力 XSD** と**出力 XSD** は、後で使用します）。
5. **Web サービス**ダイアログの下位のパラメータ以下のように設定します。
   - **戻り値** …… XML Result
   - **完了** …… Success?
   - **エラー情報** …… Error Message
6. OK をクリックして **Web サービス**ダイアログを閉じます。

データソースの定義

Web サービスに渡すパラメータや戻り値は、XML 形式になるため、XML 形式のデータソースを定義します。

1. **データ**リポジトリを開きます（***Shift+F2***）。
2. **データ**リポジトリのヘッダ行にカーソルを置き、**オプション／定義取得**（***F9***）を実行します。



3. **データソース定義取得**ダイアログが表示されたら、**XML** タブをクリックします。以下のようにパラメータを設定します。
   - **データベース**…… ***Default XML Database***
   - **スキーマ URL** ……前述の **Web サービス**ダイアログに設定された、**入力 XSD** と**出力 XSD** のファイル名を指定します。
     上記の場合は、以下のようなファイル名を指定します。
     - 入力 XSD …… <Magic xpa のインストールフォルダ>¥SoapClientModules¥SOAPLITE¥CountryInfoService_CountryName_In.xsd
     - 出力 XSD …… <Magic xpa のインストールフォルダ>¥SoapClientModules¥SOAPLITE¥CountryInfoService_CountryName_Out.xsd
   - **テンプレートプログラム**…… ***N= なし***
4. OK をクリックしてダイアログを閉じると**データ**リポジトリにデータソースが追加されます。上記の例では、以下のような名前のデータソースが２つ追加されます。
   CountryInfoService_CountryName

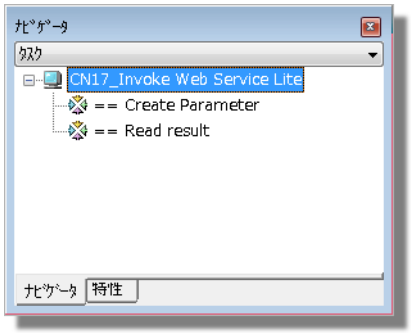
名前が同じですが、データソース名（XML ファイル名）が異なります。

• CountryName.xml …… **入力 XSD 用**

• CountryNameResponse.xml …… **出力 XSD 用**

作成されたデータ
ソースは、Web サービスを呼び出す際のパラメータとして利用します。

パラメータの定義ロジックの作成



1. プログラムリポジトリに戻り、最初のプログラムを開きます。
2. ナビゲータペインを開き、*F4* を押下して、以下の 2 つのサブタスクを作成します。
   • Create Parameter

タスクタイプ	***B= バッチ***
初期モード	***C= 登録***
タスク終了条件	***Yes***
チェック時期	***A= 後置***
メインソース	入力 XSD 用データソース (CountryInfoService_CountryName)
タスクデータビュー	#3:sCountryISOCode
レコード後ロジックユニット	目更新／ sCountryISOCode ／親タスクの "Country Code"

• Read result

タスクタイプ	***B= バッチ***
初期モード	***M= 修正***
タスク終了条件	***Yes***
チェック時期	***A= 後置***
メインソース	出力 XSD 用データソース (CountryInfoService_CountryName)
タスクデータビュー	#3:CountryNameResult
レコード後ロジックユニット	項目更新／ sCountryISOCode ／親タスクの "Country Code"

3. 親タスクのロジックエディタを開き、イベントロジックユニット内にカーソルを置きます。
4. 外部コール処理コマンドの前に以下のロジックを定義します。

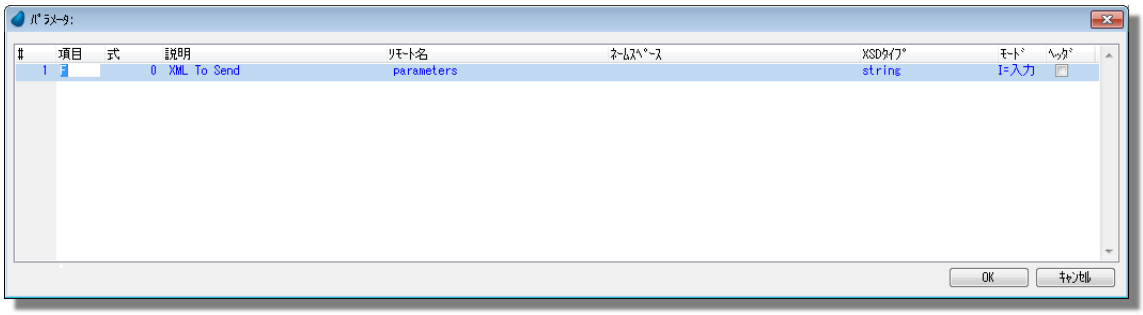
処理コマンド		タスク名	式
アクション			DbDel(入力 XSD 用データソース),'')
アクション			DbDel(出力 XSD 用データソース),'')
コール	サブタスク	Create Parameter	
項目更新	XML To Send		File2Blb(DbName(入力 XSD 用データソース))

5. 外部コール処理コマンドの後に以下のロジックを定義します。

処理コマンド		タスク名	式
アクション			Blb2File(変数 <XML Result>,DbName(出力 XSD 用データソース))
コール	サブタスク	Read result	

パラメータの設定

Web サービスプロバイダに渡すパラメータを設定します。プロバイダ側のパラメータの書式に合った値を渡す必要があります。



1. **コール Web サービスライト**処理コマンドの**パラメータ**特性でズームすると**パラメータ**テーブルが表示されます。**リモート名**、**Xsd タイプ**、**モード**にはすでに WSDL アシストによって設定されていますので、これに対応した Magic 側の値（[項目] または、[式]）を設定します。

ここでは**項目**カラムで*ズーム*して、変数「**XML to Send**」を設定します。

コール Web サービスライトの実行

作成したプログラムを実行し、Web サービスプロバイダを呼び出します。

1. 上記で作成したプログラムを実行します。
2. **Country Code** に JP と入力します。（JP は、日本の国コードです。）
3. 正常に処理された場合は、**戻り値**に、Japan が返ります。

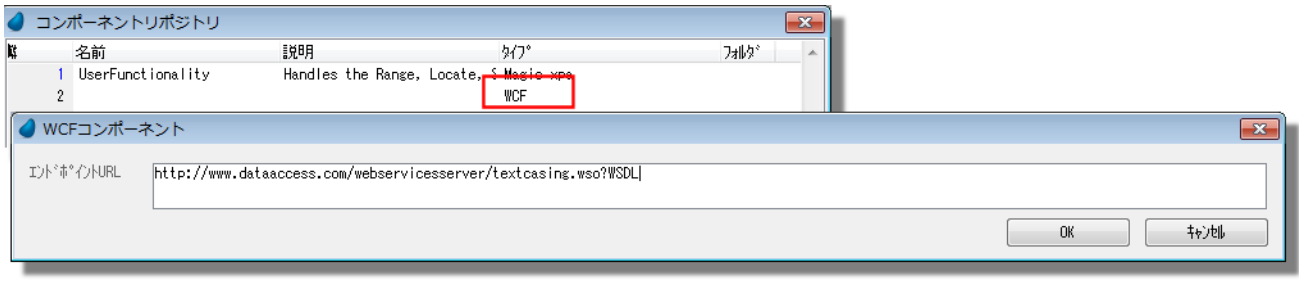
WCF Web サービスのコンシューマとして設定するには

注： Magic xpa Studio で WCF コンポーネントの作成処理を行うには、ServiceModel メタデータ ユーティリティ ツール (Svcutil.exe) が必要です。Windows SDK をインストールすると利用できます。

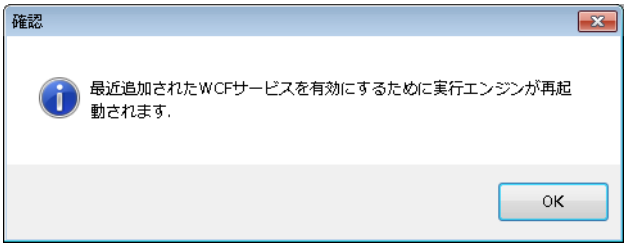
Windows Communication Foundation （WCF）を使用することで、Web エンドポイントを公開するサービスを作成することができます。これによって、ワールドワイドな基準に基づいたサービスの大きなライブラリに対する容易でセキュアなアクセスと利用を提供します。Magic xpa では、コンポーネントを作成することで利用できるようになります。

コンポーネントの登録

1. **プロジェクト**メニューから、**コンポーネント**を選択します。
2. 一行追加し（*F4*）、**タイプ**カラムで ***WCF*** を選択します。
3. ズーム（*F5*）します。コンポーネントが選択されていなければ、**WCF コンポーネント**ダイアログが開きます。このダイアログで、Web サービスステップを呼び出すサービスのエンドポイント URL を入力してください。
   例：http://www.dataaccess.com/webservicesserver/textcasing.wso？ WSDL



4. OK をクリックしてダイアログを閉じます。しばらく時間がかかる場合がります。この時点で、WCF 用のファイルが、<Magic xpa のインストールフォルダ>¥WCFClientModules にフォルダが作成され、**名前**カラムにコンポーネントの名前としてフォルダ名と同じ名前が自動的に設定されます。
5. **コンポーネント**リポジトリを閉じると、以下のメッセージダイアログが表示され、実行エンジンが再起動されます。



6. コンポーネントがすでにロードされている場合は、**WCF コンポーネント特性**ダイアログが開きます。

WCF を利用した Web サービス用プログラム

このように Web サービスを定義すると、他の .NET コンポーネントと同じような方法で利用することができます。

例えば、上記の Web サービスを定義した場合、以下の式を使用した**エラー**処理コマンド定義することができます。

```
DotNet.TextCasingSoapTypeClient('TextCasingSoap').InvertStringCase('My String')
```

プログラムを実行すると、テキスト（”mY sTRING”）が返るメッセージボックスが表示されます。

パラメータの "TestCastingSoap" は、.Net オブジェクトのインスタンスを作成する際に必要になります。この名前は、<Magic xpa のインストールフォルダ >¥WCFClientModules¥system.servicemodel.client.config ファイル内の エンドポイントの名前を指定します。

1 つのサービスに対して複数のエンドポイントが定義されている場合、インスタンスの作成時にエンドポイント名を指定する必要があります。1 つのエンドポイントしか定義されていない場合は、指定する必要はありません。

system.servicemodel.client.config ファイルが以下の場合、以下の赤字のどれかを指定することになります。

```
<?xml version="1.0" encoding="utf-8" ?>
<client>
    <endpoint address="http://www.dataaccess.com/webservicesserver/textcasing.wso"
        binding="basicHttpBinding" bindingConfiguration="TextCasingSoapBinding"
        contract="TextCasingSoapType" name="TextCasingSoap" />
  <endpoint address="http://www.dataaccess.com/webservicesserver/textcasing.wso"
        binding="customBinding" bindingConfiguration="TextCasingSoapBinding12"
        contract="TextCasingSoapType" name="TextCasingSoap12" />
</client>
```

注： Windows 認証を使用する WCF サービス設定を利用する場合、WCF コンシューマプロジェクトの出力ファイルを (別の PC 上に) インポートした後、コンポーネントを再読み込みするか、WCFClientModules¥system.servicemodel.client.config ファイル内の userPrincipalName プロパティ内のホスト名を変更する必要があります。WCF コンポーネントが作成される際、PC（Windows 認証が使用された場合のログオンユーザ）特有の設定が作成されますが、プロジェクトまたは WCF コンポーネントをインポートする時にはこの設定が変更されないため、この処理が必要になります。

上記の例の Web サービスの内容は、以下を参照してください。

注： 外部のサービスのため、内容については保証できません。

http://www.dataaccess.com/webservicesserver/textcasing.wso

サポートバージョン：3.1

WCF Web サービスでメッセージコンストラクトとして設定するには

注： Magic xpa Studio で WCF コンポーネントの作成処理を行うには、ServiceModel メタデータ ユーティリティ ツール (Svcutil.exe) が必要です。Windows SDK をインストールすると利用できます。

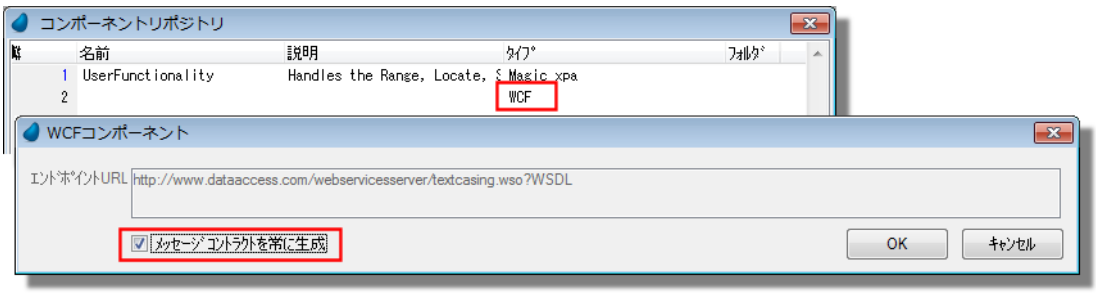
Windows Communication Foundation (WCF) を使用した場合、データ送信するメッセージの構成は考慮する必要がなくなります。ただし、SOAP メッセージセのセキュリティを制御するなどの場合、メッセージ構成を指定できるメッセージコンストラクタを作成することで対応できます。

ここでは、「WCF Web サービスのコンシューマとして設定するには」（712 ページ）で利用した同じ Web サービスをメッセージコンストラクトを作成して呼び出す方法について説明します。

メッセージコンストラクトを作成する場合、.NET オブジェクトにアクセスするする際のシグニチャが変わることに注意してください。

コンポーネントの登録

1. **プロジェクト**メニューから、**コンポーネント**を選択します。
2. 一行追加し（*F4*）、**タイプ**カラムで ***WCF*** を選択します。
3. ズーム（*F5*）します。コンポーネントが選択されていなければ、**WCF コンポーネント**ダイアログが開きます。このダイアログで、Web サービスステップを呼び出すサービスのエンドポイント URL を入力してください。
   例：http://www.dataaccess.com/webservicesserver/textcasing.wso？ WSDL



4. **メッセージ今ストラクトを常に生成**のチェックボックスをチェックします。
5. OK をクリックしてダイアログを閉じます。しばらく時間がかかる場合がります。この時点で、WCF 用のファイルが、<Magic xpa のインストールフォルダ>¥WCFClientModules にフォルダが作成され、**名前**カラムにコンポーネントの名前としてフォルダ名と同じ名前が自動的に設定されます。
6. **コンポーネント**リポジトリを閉じると、メッセージダイアログが表示され、実行エンジンが再起動されます。
7. コンポーネントがすでにロードされている場合は、**WCF コンポーネント特性**ダイアログが開きます。

WCF を利用した Web サービス用プログラム

このように Web サービスを定義すると、他の .NET コンポーネントと同じような方法で利用することができます。

1. **プログラム**リポジトリにオンラインタスクを作成します。
2. **データビュー**エディタを開き、以下のような変数項目を定義します。

#	名前	型	書式	オブジェクトタイプ
A	net	.=Net		TextCasingSoapTypeClient
B	req	.=Net		InvertStringCaseRequest
C	resp	.=Net		InvertStringCaseResponse
D	string	A= 文字	30	
E	Return	A= 文字	30	

3. ロジックエディタを開き、以下のようなロジックを定義します。

処理コマンド	変数項目	タスク名	式
項目更新	A: net		DotNet.TextCasingSoapTypeClient('TextCasingSoap')
項目更新	B: req		DotNet.InvertStringCaseRequest()
項目更新	C: resp		DotNet.InvertStringCaseResponse()
アクション			DNSet(B.Body,DotNet.InvertStringCaseRequestBody(D))
項目更新	C: resp		A.InvertStringCase(B)
項目更新	E: Return		C.Body.InvertStringCaseResult

4. プログラムを実行し、**変数項目 C（string）**に My String を入力すると、**変数項目 E（Return）**にテキスト mY sTRING が返ります。

実行結果は、「WCF Web サービスのコンシューマとして設定するには」（712 ページ）と同じになりますが、プログラムの内容が異なることに注意してください。

サポートバージョン：3.1a

WCF を使用して、Web サービスの障害を処理するには

Web サービスのコンシューマとして動作する際、Web サービスから障害が返り、それに対する対応を行う必要があります。ここでは、どのようにそれを行うかについて説明しています。

1. コール .NET 処理コマンドを定義し、.NET コードダイアログボックスにズームしてください。

以下の例には、Web サービスの障害情報を取得する .NET コードが定義されています。

2. 最初に、リクエストを送るコードを追加します。これは、紫色の枠でマークされたコードです。
3. さらに、障害情報を取得するコードを追加します。これは、青色の枠でマークされたコードです。

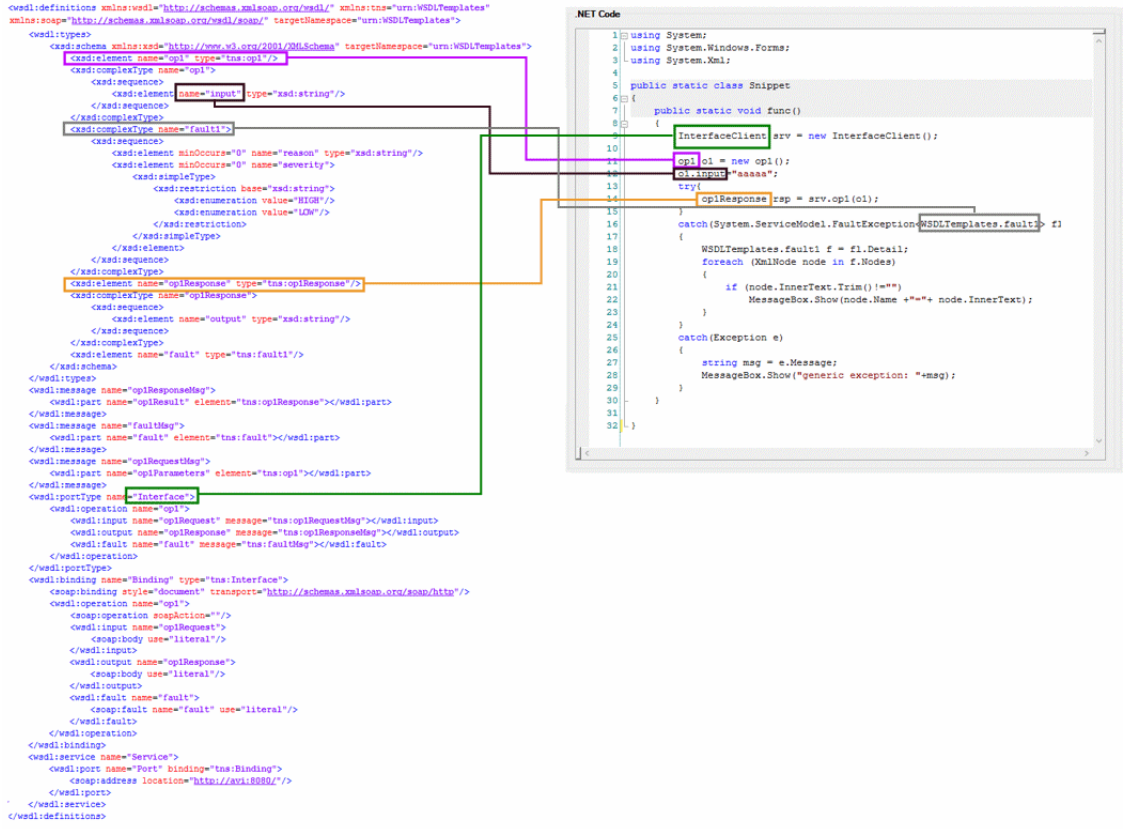
.NET Code

```
using System;
using System.Windows.Forms;
using System.Xml;

public static class Snippet
{
    public static void func()
    {
        InterfaceClient srv = new InterfaceClient();

        op1 o1 = new op1();
        o1.input="aaaaa";
        try{
            op1Response rsp = srv.op1(o1);
        }
        catch(System.ServiceModel.FaultException<WSDLTemplates.fault1> fl
        {
            WSDLTemplates.fault1 f = fl.Detail;
            foreach (XmlNode node in f.Nodes)
            {
                if (node.InnerText.Trim()!="")
                    MessageBox.Show(node.Name +"="+ node.InnerText);
            }
        }
        catch(Exception e)
        {
            string msg = e.Message;
            MessageBox.Show("generic exception: "+msg);
        }
    }

}
```

上記のコードの中の特定の項目は、指定されたサンプルを参照します。?

4. スキーマから項目をコードに合うように調整してください。

以下の表示のように、その項目は、以下のイメージのスキーマに対応しています。


.NET Code

Java を使用して Web サービスのコンシューマとして使用するには

Apache Axis2 とは、Java と XML 技術に基づいた Web サービスのフレームワークです。

Magic xpa 3.3 では、Web サービスのコンシューマとして、Apache Axis2 を利用するようになりました。

Java を使用して Web サービスを使用するには、次の手順を実行します。

1. ［プロジェクト］メニューから［コンポーネント］を選択します。
2. 一行追加します（*F4*）。
3. ［タイプ］カラムで、「Apache Axis2」を選択します。
4. ズーム（*F5*）します。
5. コンポーネントが選択されていない場合、［Apache Axis2 コンポーネント］ボックスが開きます。
6. このボックスで、Web サービスのステップが呼び出すサービスのエンドポイント URL を入力します。例：http://www.dataaccess.com/webservicesserver/textcasing.wso?WSDL
7. コンポーネントが既にロードされている場合は、［Apache Axis2 コンポーネント特性］ダイアログが開きます。
8. Web サービスを定義したら、ほかの Java コンポーネントと同じ方法で Web サービスを使用することができます。

例：

［コンポーネント］リポジトリに Java エイリアスを定義します。例：TextCasingStub = com.dataaccess.www.webserviceserver.TextCasingStub

（ヒント：Web サービス用のモジュールが、〈Magic xpa のインストールフォルダ〉¥JavaClientModules¥〈コンポーネント名〉に作成されます。この中の Wsdl2JavaOutput¥src サブフォルダ内の com フォルダ以降のパスと Java ファイルをもとにエイリアスの Java タイプを定義することができます。）

次のようにプログラムを定義します。

		型	オブジェクトタイプ
変数	A:textCasingStub	JAVA	TextCasingStub
変数	B: invertStringCaseRequest	JAVA	TextCasingStub.InvertStringCase
変数	C: invertStringCaseResponse	JAVA	TextCasingStub.InvertStringCaseResponse

項目更新	A	Java.TextCasingStub()	
項目更新	B	Java.TextCasingStub.InvertStringCase()	
アクション		B.setSAString('My String')	
項目更新	C	A.invertStringCase(B)	
エラー	エラー	JavaException().toString()	条件 : JavaExceptionOccured()
エラー	警告	C.getInvertStringCaseResult()	

プログラムを実行すると、('mY sTRING') がメッセージボックスで表示されます。

参照 : **上記の Magic xpa のロジックは、以下のネイティブな Java プログラムのロジックに従っています。**

```
import com.dataaccess.www.webservicesserver.TextCasingStub;
public class TestClient
{
 public static void main(String[] args) throws Exception
 {
  try
   {
    TextCasingStub textCasingStub=new TextCasingStub();
    TextCasingStub.InvertStringCase invertStringCaseRequest = new TextCasingStub.InvertStringCase();
    invertStringCaseRequest.setSAString("My String");
    TextCasingStub.InvertStringCaseResponse invertStringCaseResponse =
            textCasingStub.invertStringCase(invertStringCaseRequest);
    System.out.println(String.format("Response: %s", invertStringCaseResponse.getInvertStringResult()));
   }
  catch (Exception ex)
```

```
    {
     system.out.println(ex);
    }
   }
}
```

参照 : 「WCF Web サービスのコンシューマとして設定するには」（712 ページ）

サポートバージョン : 3.3

［このページは意図的に空白にしています。］

第 34 章 : Web サービス（プロバイダ）

注： Magic xpa Ver3.3 から Systinet Server for Java（SOA Enablement Server for Java) がバンドルされなくなりました。また、Web サービスインタフェースビルダもなくなりました。

Magic エンジンを Web サービスのプロバイダとして利用するには、アプリケーションサーバとして Apache-Tomcat と Axis2 が必要です。またミドルウェアとしてインメモリ・データグリッド（IMDG) が必要です。Axis2 のサービスとして動作する aar(Axis Application Archive) を作成する必要があります。

Magic xpa で Web サービスを提供するには

Web サービスは、異なるプラットフォームやフレームワークで動作するアプリケーション間で相互運用するための標準的な方法です。Web サービスを利用することで、Magic アプリケーションを外部に公開することができます。Web サービスを使用することで、世界のあらゆる場所にある様々な言語で作成された、色々なタイプのアプリケーションとアクセスすることができます。しかし、このサービスを手動で作成するには非常に手間がかかる場合があります。Magic xpa を使用することで、このようなサービスの作成が自動化され作業が容易になります。

Magic プロジェクトを Web サービスとして提供することができます。Web サービス用のプログラムは、バッチタスクにパラメータと戻り値を定義することで作成できます。これらのプログラムが Web サービスで使用できるように、Magic xpa はインタフェースを作成します。Web サービスを提供するための手順を以下で説明します。

Magic xpa のセットアップ

Web サービスを作成するには、Magic xpa の 64bit のサーバ製品が適切にセットアップされる必要があります。

Magic xpa をインストールする際に、以下のコンポーネントを選択する必要があります。

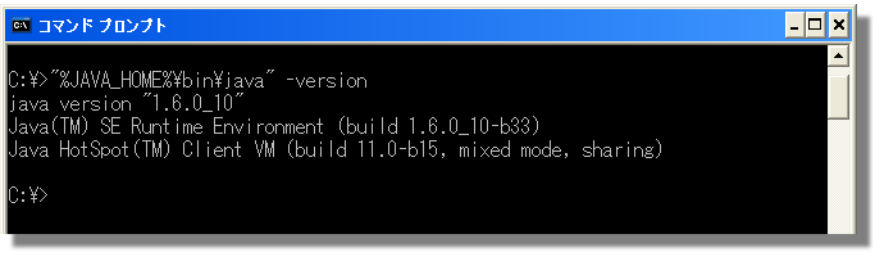
- ミドルウエア／ GigaSpace
- インターネットリクエスタ／ ASP.NET(Microsoft)
- インターネットリクエスタ／ Apache Tomcat リクエスタ
- Web サービス／ Web サービス　プロバイダ用フレームワーク

この設定により、Apache Tomcat Web サーバがインストールされます。また Java フォルダに Open JDK もインストールされます。

Java の確認

Magic xpa を DVD からインストールした場合、Java フォルダに Open JDK がインストールされます。

インストールされていない場合は、64bit 版の Open JDK 1.8 を取得したうえで、Magic xpa のインストールフォルダ内にコピーしてください。



Open JDK を別途インストールした場合は、以下の設定ファイルにインストールパスを設定してください。

- Magic.ini の [MAGIC_JAVA] セクション
  JAVA_HOME_64=<JDK のインストールパス >¥
- <Magic xpa のインストールパス >¥GigaSpaces-xpa¥bin¥setenv.bat の 11 行目
  set JAVA_HOME=<JDK のインストールパス >¥
- apache-tomcat-9.0.20¥bin¥setenv.bat の 10 行目

set JAVA_HOME=D:¥Magicxpa¥Server47¥Java¥64¥

- <Magic xpa のインストールパス>¥GigaSpaces-xpa¥DotNet¥Config の 16 行目
  <XapNet.Runtime.JavaHome>JavaHomePath</XapNet.Runtime.JavaHome>

注： Java のバージョンによっては動作しない場合があります。

Apache-Tomcat を操作するには

Magic xpa のインストール時に、インターネットリクエスタ /Apache Tomcat を選択すると Apache Tomcat がインストールされます。これは、Web サービスを実行するための Web サーバです。

ここでは、Apache Tomcat の管理画面の操作方法について説明します。

Web アプリケーションマネージャの設定

tomcat-users.xml （<Magic xpa のインストールフォルダ >¥apache-tomcat-9.0.20¥conf¥tomcat-users.xml）を編集します。

Web アプリケーションマネージャのユーザ名、パスワードが以下の場合を想定します。

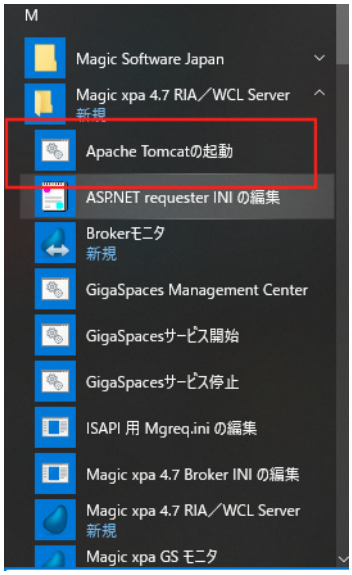
- ユーザ名：admin
- パスワード：pass

以下のように追加します。

```
<tomcat-users>
<role rolename="manager-gui"/>
<user username="admin" password="pass" roles="manager-gui"/>
<!--
  NOTE:  By default, no user is included in the "manager-gui" role required
  to operate the "/manager/html" web application.  If you wish to use this app,
  you must define such a user - the username and password are arbitrary.
-->
```

Tomcat の起動

Magic xpa のスタートメニュー内に ”Apache Tomcat の起動 ” のショートカットがあります。これをクリックすると起動されます。



Tomcat フォルダ配下の bin フォルダ内の「startup.bat」をダブルクリックしても起動させることができます。

正常に起動していれば、最後に「情報 : Server startup in xxxxms」というような起動にかかった時間が表示されます。

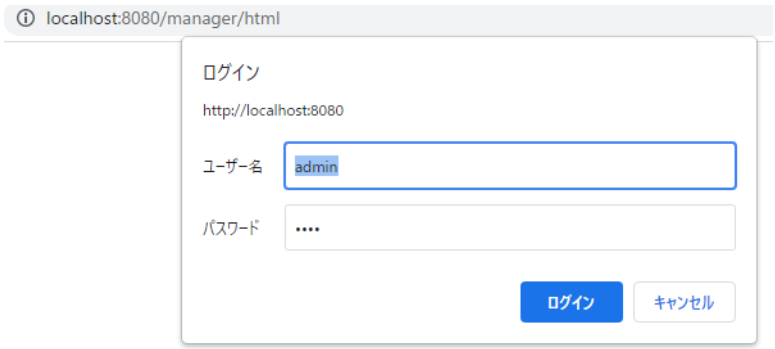
Web アプリケーションマネージャを起動

Tomcat が起動されたら、確認のために、Web アプリケーションマネージャを起動します。

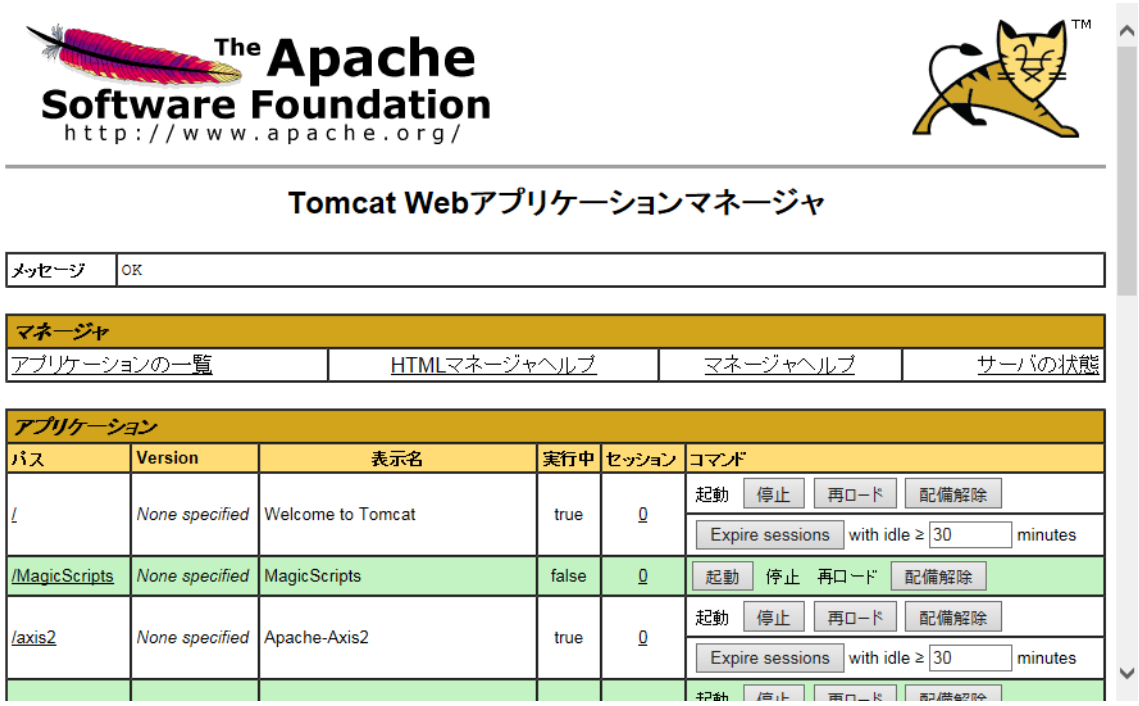
Web ブラウザから以下の URL を開きます。

```
http://localhost:8080/manager/html
```

BASIC 認証のログインダイアログが表示されるので指定したユーザ名、パスワードを入力します。



［OK］をクリックすると、「Web アプリケーションマネージャ」が開きます。



これで、正しくインストールされ Web サービスをデプロイする用意ができたことになります。

Tomcat の停止

bin フォルダ内「shutdown.bat」をダブルクリックすると停止します。

Tomcat で Axis2 を実行

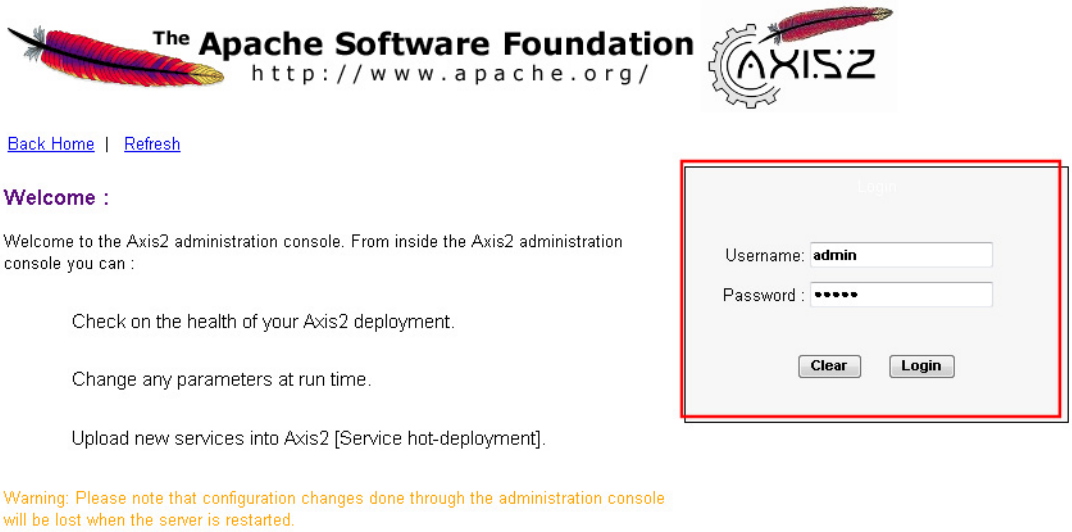
Apache Axis2 は、Java と XML 技術に基づいた Web サービスのフレームワークです。

Tomcat を起動してから、Web ブラウザを開いて以下の URL を開き、Werlcome 画面で「Administration」をクリックします。

```
http://localhost:8080/axis2
```

Axis2 管理コンソールのログイン画面が表示されます。

デフォルトのユーザー名「admin」とパスワード「axis2」を入力してログインします。



ログインに成功すると、管理コンソールが表示されます。

Web サービスをアップロードするためのリンクをクリックします。

Web サービスのプロバイダとしてビルドするには

サンプルの Web サービス

以下の機能を提供する Web サービスを作ります。

- hellow() …… "Hello" という文字列を返す。
- getUpperString …… パラメータとして渡した英字の文字列を全て大文字に変換する
- add …… 足し算を行う

Java プログラムの作成

上記の機能を提供する Java プログラム (Axis2Sample.java) を作成します。

```
public class Axis2Sample {
   public String hello() {
      return "hello";
   }
   public String getUpperString(String s) {
      return s.toUpperCase();
   }
   public int add(int a, int b) {
      return a + b;
   }
}
```

コマンドプロンプトを開きプログラムをコンパイルします。

```
Javac Axis2Sample.java
```

aar ファイルを作成する

まず、services.xml を作成します。

```
<service name="Axis2Sample" scope="application">
    <messageReceivers>
        <messageReceiver
            mep="http://www.w3.org/ns/wsdl/in-only"
class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"/>
        <messageReceiver
            mep="http://www.w3.org/ns/wsdl/in-out"
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </messageReceivers>
    <parameter name="ServiceClass">
        Axis2Sample
    </parameter>
</service>
```

Axis2Sample をルートとしたディレクトリ構成を以下ように定義します。

```
Axis2Sample
   - META-INF
      - services.xml
   - lib
   - Axis2Sample.class
```

コマンドプロンプトでカレントを Axis2Sample ディレクトリ に移動し次のコマンドを実行します。

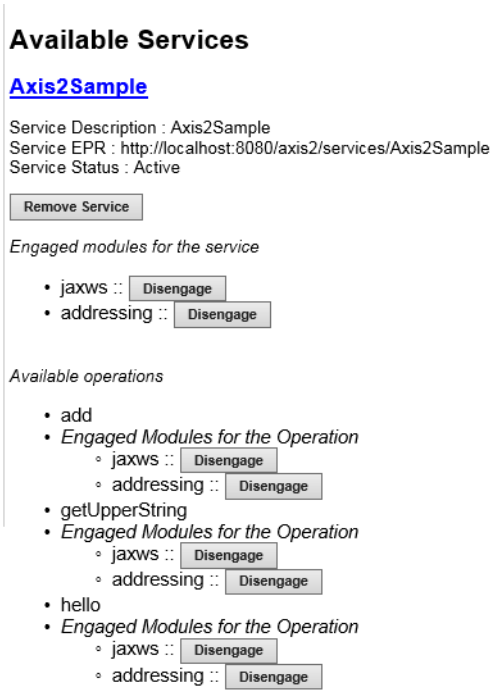
```
jar cvf axis2sample.aar META-INF/ lib/ Axis2Sample.class
```

これで、"axis2Sample.aar" というファイル (Apache Axis2 Archive) が作成されます。

Web サービスを登録する

作成された aar ファイルをサービスとして登楼します。

1. Axis2 （http://localhost:8080/axis2）の管理コンソール（Administration）を開いて、Tools/Update Service をクリックします。
2. "Upload an Axis Service Archive File" 画面が表示されます。
3. ［参照］ボタンをクリックして "axis2Sample.aar" ファイルを選択し、［Upload］をクリックします。
4. 読み込みに成功したら、"File axis2sample.aar successfully uploaded." と表示されます。
5. Axis2 の管理コンソールの System Components/Available Services をクリックします。ここに読み込まれたサービス名が表示されます。
（aar ファイルは、%TOMCAT_HOME%/webapps/axis2/WEB-INF/services にコピーされます）



サービス名（この例では、"Axis2Sample"）のハイパーリンクをクリックすると、Wsdl の内容（URL:http://localhost:8080/axis2/services/Axis2Sample?wsdl）が表示されます。

Magic プログラムから呼び出す

上記で作成した Web サービスを Magic プログラムから呼び出す場合は、以下のようにします。

コンポーネントの定義

1. ［コンポーネント］リポジトリで一行作成します。［タイプ］カラムで、¢Apache Axis2£ を選択します。
2. ［エンドポイント URL］で Wsdl の URL（http://localhost:8080/axis2/services/Axis2Sample?wsdl）を指定します。
3. ［OK］をクリックすると Web サービスのコンシューマ用モジュールが定義されます。（<Magic xpa のインストールフォルダ >¥JavaClientModules）
4. ［Java エイリアス］をクリックします。［Java エイリアス］ダイアログが開きます。
5. ［エイリアス］カラムには、任意の名前。［java タイプ］には「org.apache.ws.axis2.Axis2SampleStub」と入力します。

プログラムの作成

add を呼び出すプログラムを作成します。

オンラインプログラムを作成します。

［データビュー］エディタ

以下の変数を定義します。（Axis2SSatestStub は、エイリアス名です）

		型	オブジェクトタイプ
A	Axis2SSatestStub	JAVA	Axis2SSatestStub
B	Axis2SSatestStub.Add	JAVA	Axis2SSatestStub.Add

		型	オブジェクトタイプ
C	Axis2SSatestStub.AddResponse	JAVA	Axis2SSatestStub.AddResponse
D	Int1	数値	
E	Int2	数値	
F	Return	数値	

［ロジック］エディタ

以下の処理コマンドをを定義します。

[タスク前]		式	条件
項目更新	A （Axis2SSatestStub）	Axis2SSatestStub	
[イベント]	（ユーザイベント）		
項目更新	B （Axis2SSatestStub.Add）	Java.Axis2SSatestStub.Add()	
項目更新	C (Axis2SSatestStub.AddResponse)	Java.Axis2SSatestStub.AddResponse()	
アクション		B.setArgs0(D)	
アクション		B.setArgs1(E)	
項目更新	C (Axis2SSatestStub.AddResponse)	A.add(B)	
エラー		JavaException().toString()	JavaExceptionOccurred()
項目更新	F (Retuen)	C.get_return()	

［フォーム］エディタ

変数 D (Int1)、E (Int2)、F (Return) をフォームに配置します。［プッシュボタン］コントロールを配置し、ロジックで発行するイベントを定義します。

Magic アプリケーションを Web サービスのプロバイダにするには

Magic xpa 3.2 までは、Web サービスビルダを使用して、Magic アプリケーションを Web サービスのプロバイダにすることができました。Ver3.3 からは、Systinet や Web サービスビルダ が利用できないため、Apache Axis2 で実行する aar で Magic アプリケーションの中継することになります。

この機能を利用する場合は、ミドルウエアとして GigaSpace が必須となります。、

Magic アプリケーション

Web サービスのプロバイダとする Magic アプリケーションを以下のように作成します。

これは、SOAP によって各データ型のパラメータを受け取り、文字列で返すプログラムです。

注： パラメータに２バイトコードの文字列を含める場合は、Unicode 型を使用してください。

タスク特性

アプリケーション名：	Sample_Provider	**タスクタイプ：**	バッチ
プログラム名：	DataType	**タスク終了条件：**	Yes
公開名：	DataType	**チェック時期：**	B= 前置
公開：	チェック	**戻り値：**	returnedString

［データビュー］エディタ

パラメータ：	Alpha	文字型	10
パラメータ：	Unicode	Unicode 型	10
パラメータ：	Int	数値型	5
パラメータ：	Float	数値型	5.2
パラメータ：	Logical	論理型	5
パラメータ：	Blob	Blob 型	
変数	returnedString	文字型	200

［ロジック］エディタ

［タスク前］		**式**
項目更新	returnedString	StrBuild('String == @1@ | Unicode == @2@ | int == @3@ | float == @4@ | Logical == @5@ | Blob == @6@ bytes', B, Str(C,'5'), Str(D,'5.2'), IF(E,'true','false'), Str(BlobSize(F),'9'))

中継用 aar ファイルの作成

以下のディレクトリ構成を作成します。（赤字部分は、任意です）

```
Root
  - src
    - com
```

```
            -sample
        - META-INF
```

src¥com¥sample フォルダに SOAP リクエストをもとに Magic プログラムを呼び出す Java プログラム（Service1.java）を作成します。

```
package com.sample;
import com.magicsoftware.xpa.requester.Requester;
import com.magicsoftware.xpa.model.Arguments;
import com.magicsoftware.xpa.model.SoapRequest;
import com.magicsoftware.xpa.model.SoapResponse;
public class Service1
{
   static final long REQUESTER_TIMEOUT = 20;
    /**
    * @param a
    * @param b
    * @param c
    * @param d
    * @param e
    * @param f
    * @return a formatted string of the sent parameters.
    */
   public String dataTypes(String a, String b, int c, float d, boolean e, byte[] f)
   {
      String result = "";

      Arguments arguments = new Arguments(6);
      arguments.set(0, a, false);
      arguments.set(1, b, false);
      arguments.set(2, c, false);
      arguments.set(3, d, false);
      arguments.set(4, e, false);
      arguments.set(5, f, false);

      SoapRequest request = new SoapRequest("Axsia_Provider", "DataTypes", arguments);

      SoapResponse response = (SoapResponse)Requester.executeRequest(request,
REQUESTER_TIMEOUT);
      if (response != null)
         result = (String)response.getReturnValue();
      return result;
   }
}
```

ここで、"Sample_Provider" は、Magic のアプリケーション名、"DataTypes" は公開プログラム名です。

src¥META-INF フォルダに Service.xml を作成します。

```
<service name="com.sample.Service1" scope="application">
    <description>
        Xpa POJO Service #1
    </description>
    <messageReceivers>
        <messageReceiver mep="http://www.w3.org/ns/wsdl/in-only"
                         class="org.apache.axis2.rpc.receivers.RPCInOnlyMessageReceiver"/>
        <messageReceiver mep="http://www.w3.org/ns/wsdl/in-out"
                         class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </messageReceivers>
    <parameter name="ServiceClass">com.sample.Service1</parameter>
</service>
```

Root フォルダに aar ファイルを作成するための build.xml ファイルを作成します。

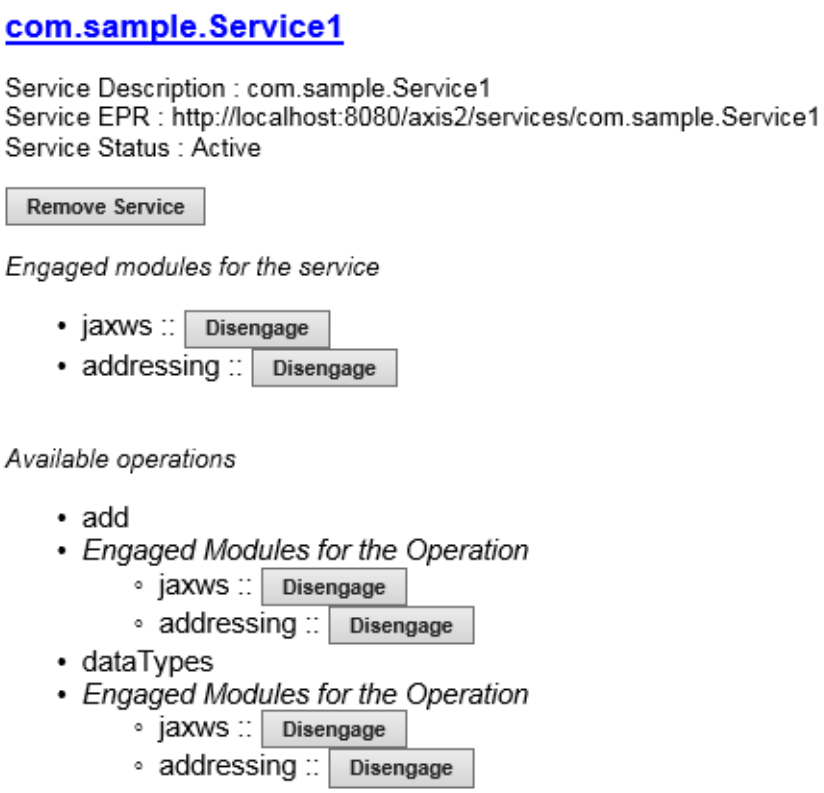
```
<project name="sample.Service1" basedir="." default="generate.service">
   <property name="service.name" value="com.sample.Service1" />
   <property name="dest.dir" value="build" />
   <property name="dest.dir.classes" value="${dest.dir}/${service.name}" />
   <property name="dest.dir.lib" value="${dest.dir}/lib" />
   <property environment="env"/>

   <path id="build.class.path">
      <fileset dir="${env.AXIS2_HOME}/lib">
         <include name="*.jar" />
      </fileset>
      <fileset dir="${env.MAGIC_XPA_HOME}/GigaSpaces-xpa/lib/xpa">
         <include name="*.jar" />
      </fileset>
   </path>
   <target name="clean">
      <delete dir="${dest.dir}" />
      <!--delete dir="src" includes="com/sample/stub/**"/ -->
   </target>
   <target name="prepare">
      <mkdir dir="${dest.dir}" />
      <mkdir dir="${dest.dir}/lib" />
      <mkdir dir="${dest.dir.classes}" />
      <mkdir dir="${dest.dir.classes}/META-INF" />
   </target>
   <target name="generate.service" depends="clean,prepare">
      <copy file="src/META-INF/services.xml" tofile="${dest.dir.classes}/META-INF/
services.xml" overwrite="true" />
      <javac srcdir="src" destdir="${dest.dir.classes}" includes="com/sample/**">
         <classpath refid="build.class.path" />
      </javac>

      <jar basedir="${dest.dir.classes}" destfile="${dest.dir}/${service.name}.aar" />
      <copy file="${dest.dir}/${service.name}.aar" tofile="${env.TOMCAT_HOME}/webapps/
axis2/WEB-INF/services/${service.name}.aar" overwrite="true" />

   </target>
</project>
```

Root フォルダに aar ファイルを作成するための build.cmd ファイルを作成します。

```
 echo off
    call "<Magic xpa のインストールフォル>¥GigaSpaces-xpa¥bin¥setenv.bat"
    call "%ANT_HOME%¥bin¥ant"
 pause
```

build.cmd を実行して aar ファイルを作成します。実行すると、Root フォルダ内に "build" フォルダが作成され、"com.sample.Service1.aar" ファイルが作成され、Axis にサービスとして配布されます。

サービスが配布されると Axsis2 の管理コンソールには、以下のように表示されます。



Java プログラムから呼び出す

上記で作成した Web サービスを Java プログラムから呼び出す場合は、以下のようにします。

Java プログラムの作成

上記の機能を提供する Java プログラム (TestClient.java) を作成します。

注： 全角文字が含まれているため、Java ファイルは、Shift-JIS で保存してください。

```
import com.sample.ComSampleService1Stub;
public class TestClient
{
   public static void main(String[] args) throws Exception
   {
      try
      {
         // Create the stub object
         ComSampleService1Stub stub = new ComSampleService1Stub();
         // 'DataTypes':
         //=============
         System.out.print("DataTypes ...¥t");
         ComSampleService1Stub.DataTypes dataTypesRequest = new
ComSampleService1Stub.DataTypes();
         dataTypesRequest.setArgs0("abc");
         dataTypesRequest.setArgs1("あいうえお");
         dataTypesRequest.setArgs2(123);
         dataTypesRequest.setArgs3((float)123.45);
         dataTypesRequest.setArgs4(true);
         System.out.println(stub.dataTypes(dataTypesRequest).get_return());
      }
      catch (Exception ex)
```

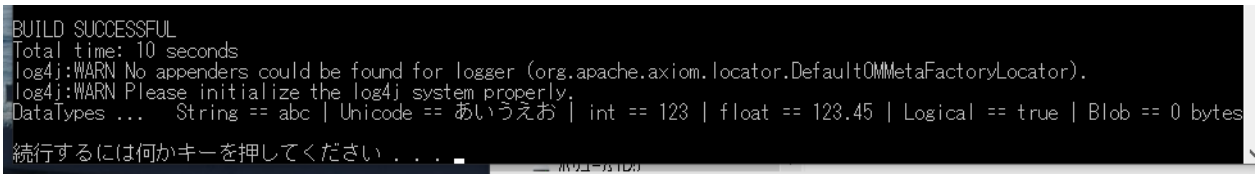
```
            {
                System.out.println(ex);
            }
        }
    }
```

Java プログラムのビルドと実行

ビルドと実行を行うためのファイルを定義します。

```
REM goto :TEST
   call ..¥..¥..¥bin¥setenv.bat
   call "%AXIS2_HOME%¥bin¥wsdl2java.bat" -uri http://localhost:8080/axis2/services/
com.sample.Service1?wsdl -d adb -o Wsdl2JavaOutput
   cd Wsdl2JavaOutput
   call "%ANT_HOME%¥bin¥ant"
   cd ..

:TEST
   "%JAVA_HOME%¥bin¥javac" -cp Wsdl2JavaOutput¥build¥lib¥com.sample.Service1-test-
client.jar;"%AXIS2_HOME%/lib/*" -d . TestClient.java
   "%JAVA_HOME%¥bin¥java" ?-cp .;Wsdl2JavaOutput¥build¥lib¥com.sample.Service1-test-
client.jar;"%AXIS2_HOME%/lib/*" TestClient

pause
```

実行して、サーバの処理が正常に終了すると以下のような結果が表示されます。



第 35 章 : データベース

データベースとの接続を定義するには

Magic xpa は、Oracle、MS-SQL Server、ODBC、および Pervasive.SQL などの複数のデータベースと接続することができます。また、同時に複数のサーバ上の複数のデータベースにリンクすることもできます。

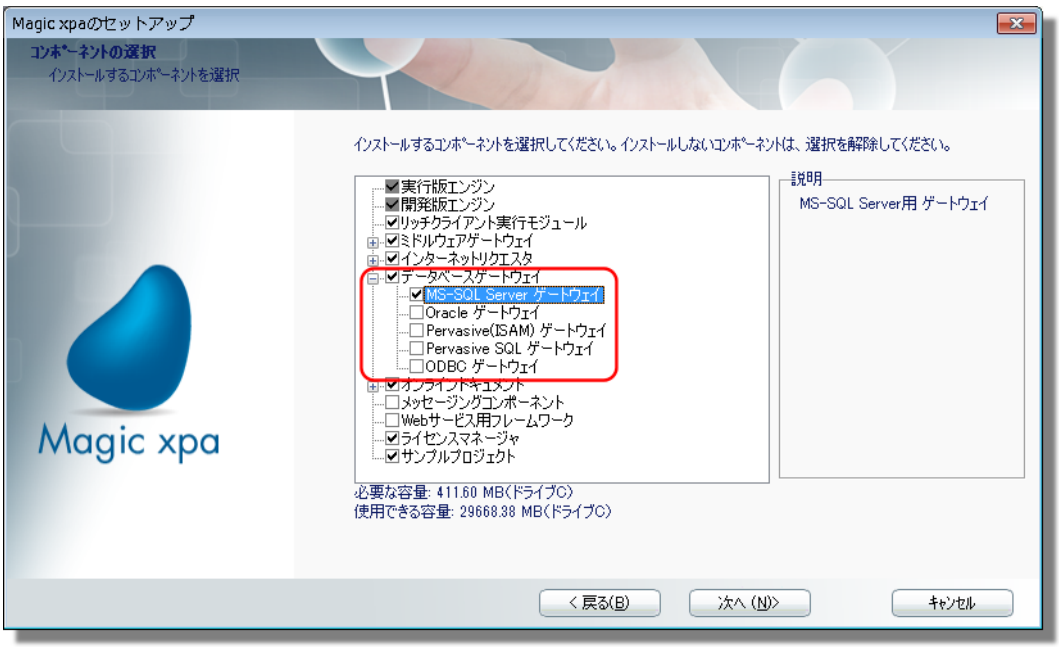
これらのデータベースを使用するには、事前に Magic xpa でこれらの RDBMS に接続するための環境設定を行う必要があります。

環境設定は、以下の手順で行います。

- データベースゲートウェイをロードできるように設定します。
- データベースを定義します。
- 接続を確認します。

各手順の詳細を以下で説明します。接続環境の内容は、使用するデータベースに依存するため、RDBMS 毎に説明します。

1. データベースゲートウェイの設定を行う



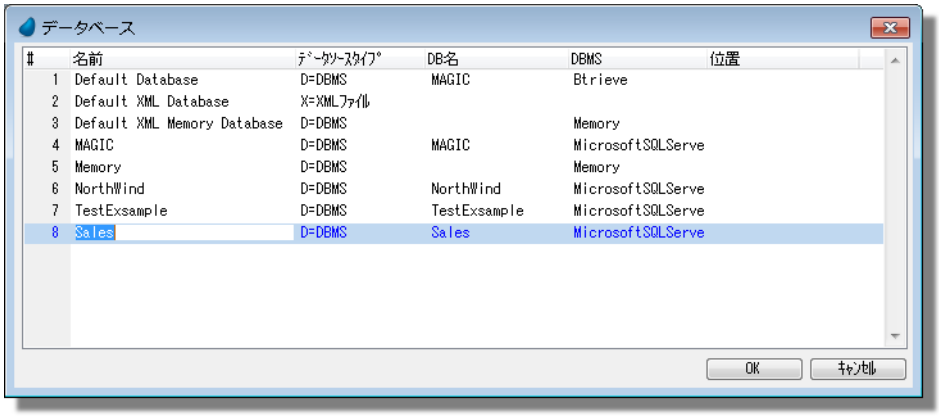
1. Magic xpa のインストール時に、使用するデータベースゲートウェイを選択することで、対応する DLL ファイルがインストールされます。Magic xpa のインストール時にゲートウェイのインストール選択を行わなかった場合、メンテナンスモードで Magic xpa のインストーラを起動する（**スタート→コントロールパネル→プログラムの追加と削除**を選択し、Magic xpa の製品名を選択した後、変更と削除を**クリック**する）ことで追加することができます。

```
308
309 [MAGIC_GATEWAYS]
310 ;MGCOMM01=mgwsock.dll
311 MGDB00=Gateways¥MGbtrieve.dll
312 ;MGDB06=Gateways¥MGdb2400.DLL
313 MGDB13=Gateways¥MGoracle.dll
314 ;MGDB16=Gateways¥MGeac32.dll
315 ;MGDB18=Gateways¥MGdb2.DLL
316 ;MGDB19=Gateways¥MGobc.dll
317 MGDB20=Gateways¥MGmssql.dll
318 MGDB21=Gateways¥MGmemory.dll
```

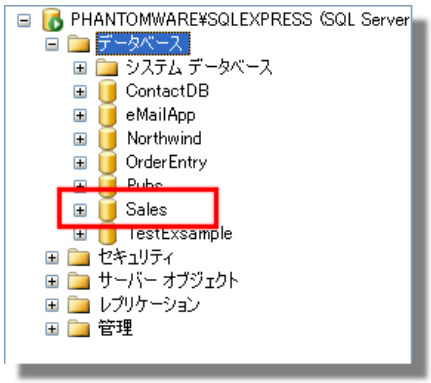
2. 次に、使用する Magic.ini ファイルの［MAGIC_GATEWAYS］セクションでデータベースゲートウェイの設定行がコメントになっていないことや、DLL ファイルのパスが正しいことを確認します（通常は、ゲートウェイをインストールすることで Magic.ini の設定も自動的に行われます）。
3. また、Magic.ini ファイルの［MAGIC_DATABASES］セクションで使用するデータベースの設定行がコメントになっていないことを確認します。
4. Magic.ini を編集したら、Magic xpa を起動します。

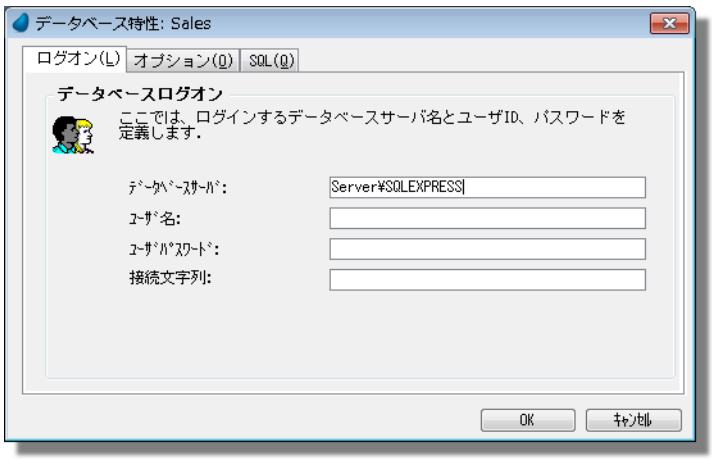
これで、データベースを定義する用意ができました。

2. データベースを定義する

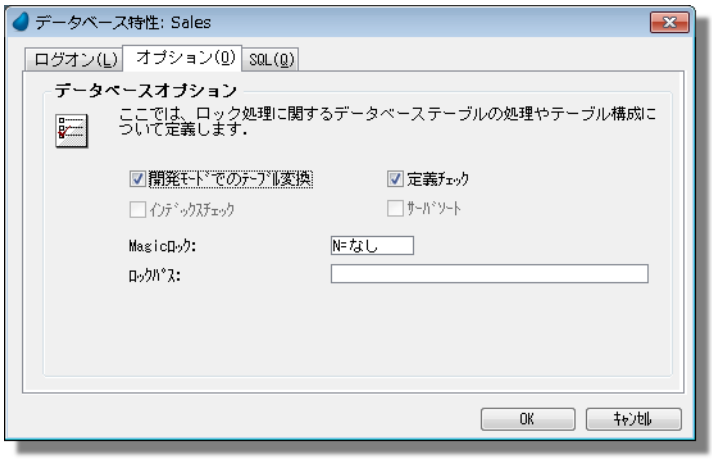


1. プロジェクト（またはアプリケーション）が開いている場合は、一旦閉じます。
2. **データベース**テーブル（**オプション→設定→データベース**）を開きます。
3. ***F4*** を押下して１行追加し、**名前**カラムに任意のデータベース名を入力します。この名前は、データベースを選択する際の表示用としてのみ使用されます。
4. **データソースタイプ**カラムで ***D=DBMS*** を選択します。
5. **データベース名**カラムでは、アクセスするデータベースの名前を入力します。
   データベース名はデータベースマネージャを使用して事前に定義しておく必要があります。この例では、MS-SQL Server に定義されている ***Sales*** というデータベースを使用するように定義されています。
6. **DBMS** カラムから**ズーム**します。Magic.ini に定義されたデータベースの名前が一覧表示されます。この例では、***Microsoft SQLServer*** が選択されています。次に、***Alt+Enter*** を押下して **DBMS 特性**を開きます。

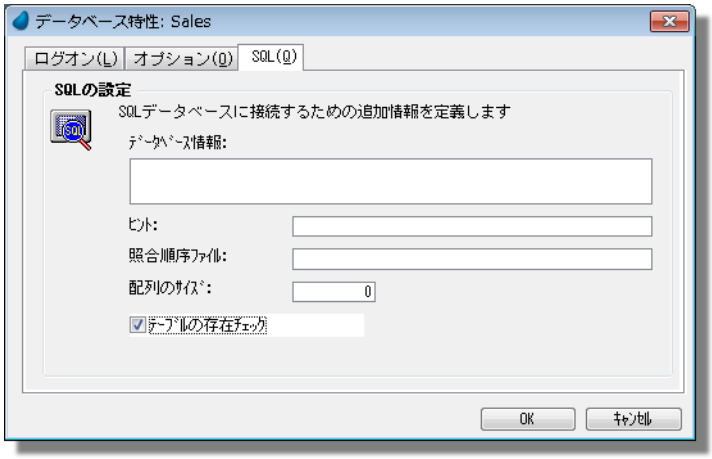




7. ログオンタブをクリックし、データベースサーバ名を設定します。必要であればユーザ ID やパスワードも指定します。この例では、Windows 認証を使用しているため、ユーザ ID やパスワードは必要ありません。



8. オプションタブをクリックします。Magic xpa 内でテーブル定義の変更を行いたい場合は、**開発モードでのテーブル変換**特性をチェックします。このテーブルが別のアプリケーションによって作成されたもので、Magic xpa 側で変更しないようにする場合は、チェックを外します。



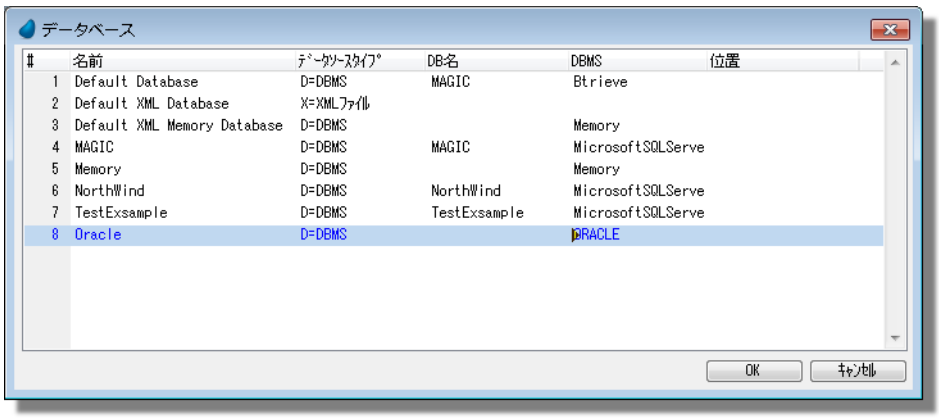
9. SQL タブをクリックします。**データベース情報**特性には、テーブルと接続するために使用する SQL コマンドを追加することができます。Magic xpa 側でテーブルを作成できるようにする場合は、**テーブルの存在チェック**特性をチェックします。

3. 接続を確認する

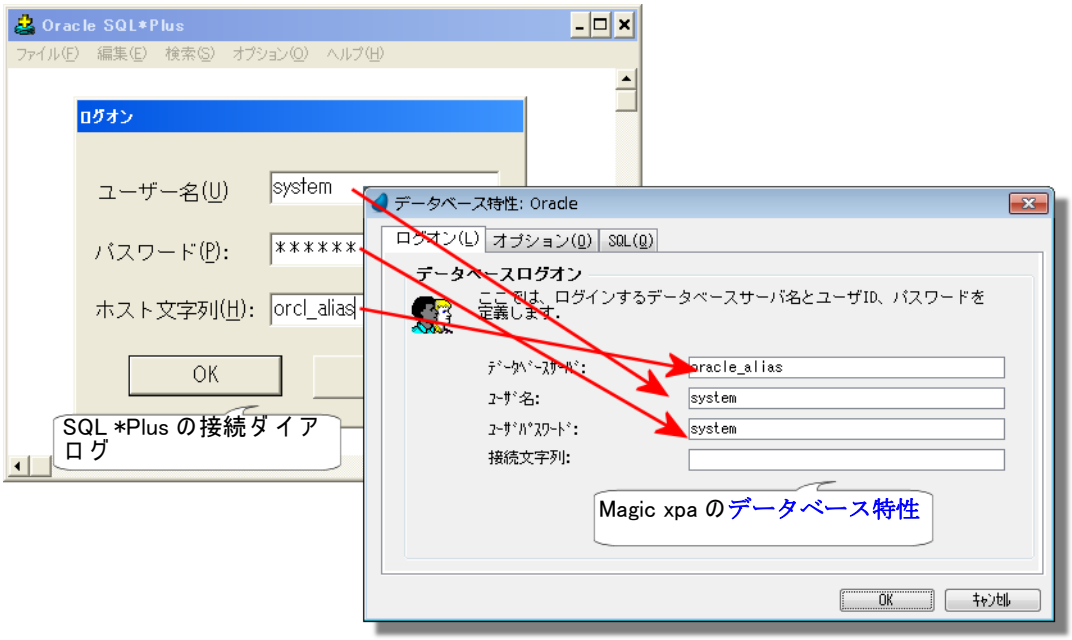
データリポジトリ内で定義取得を実行することで、データベースが正しく設定されたことを確認することができます。定義取得の方法については、第 17 章：「既存のデータベーステーブルにアクセスするには」（400 ページ）を参照してください。

DBMS 別の設定方法

Oracle



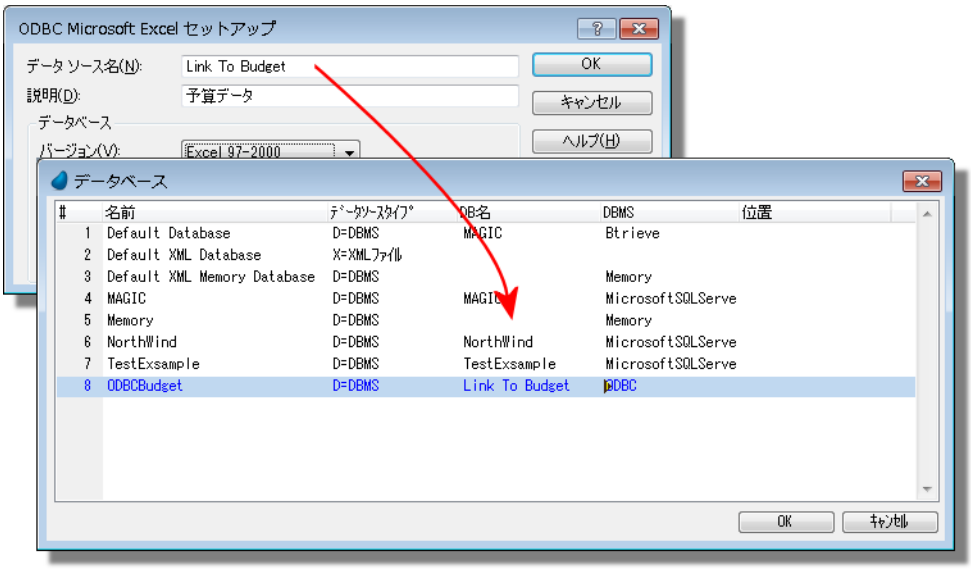
Oracle データベースを定義するには、**DBMS** カラムで ***Oracle*** を設定します。Oracle のエイリアスはすでにデータベースを示しているため、**DB 名**カラムには何も指定する必要がありません。



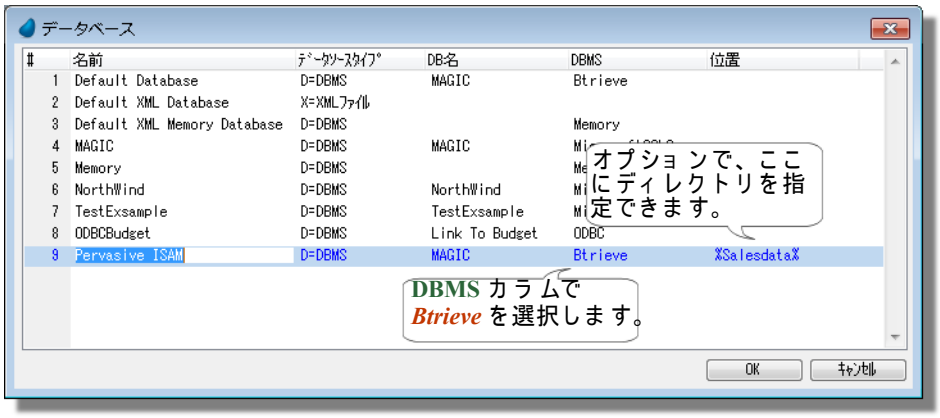
データベース特性で、**ユーザ ID** と**パスワード**が同じ場合、ホスト文字列を指定する必要があります。

ODBC ／ Pervasive SQL

ODBC データベースを定義するには、**DB 名**カラムで ODBC ／ Pervasive SQL のデータ ソース名を設定します。**データベース特性**には何も設定する必要がありません。



Pervasive ISAM



Pervasive SQL（ISAM）データベースを定義するには、データベーステーブルを設定する必要がありません。ISAM ファイルが通常の OS ファイルのように作成されます。

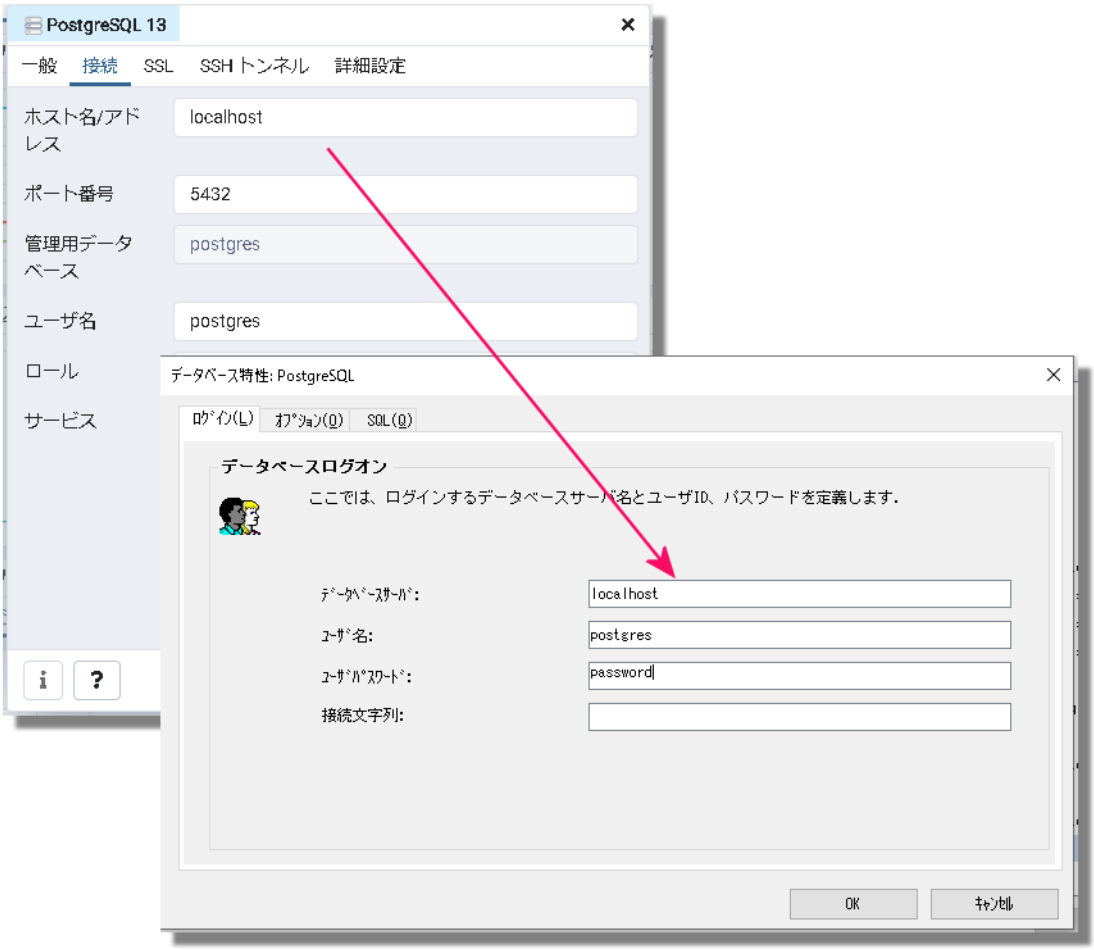
DBMS カラムで ***Btrieve*** を設定します。必要であれば、**位置**カラムにパスを指定することができます。**データ**リポジトリの**データ ソース名**カラムでパスを指定することもできます。

SQlite

SQlite は、シングルユーザ用の SQL データベースです。Magic xpa では、サンプルアプリやモバイル用データとして利用できます。

SQLite データベースを定義するには、データベーステーブル用のファイルを**位置**カラムに設定する必要があります。

PosgreSQL



PosgreSQL データベースを利用するには、以下の条件が必要です。

- ライセンス …… FUTRUE 名："MGPOSTGRE" のライセンスを登録する必要があります。
- クライアントライブラリ …… Studio や Client§32bit 版のサーバ製品では 32bit 版の クライアントライブラリ (libpq.dll) にアクセスできるようにする必要があります。
  psqlODBC（Application Stack Builder）をインストールして、＜インストールフォルダ＞\bin を Windows の 論理変数 "Path" に追加することで有効になります。

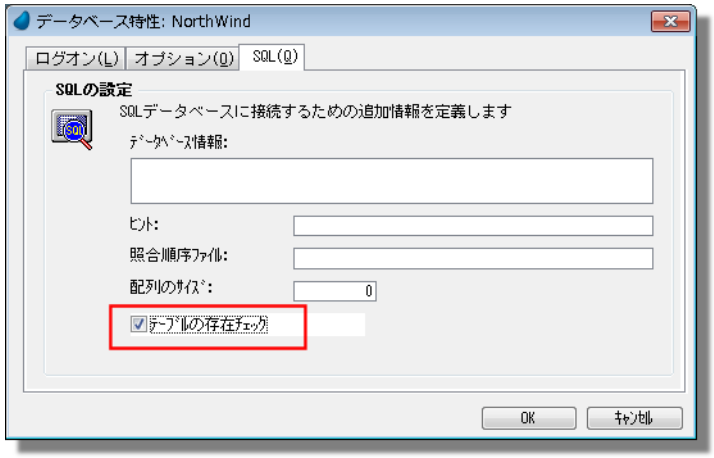
PosgreSQL のインストール方法

PostgreSQL のインストールを含めたインストール手順は以下の通りです。

1. PostgreSQL の情報は、関連サイトなどを参照してください。
2. サイトから最新の PostgreSQL を入手します。（PostgreSQL 3.x【Windows x86-64】）
3. 保存したファイルを実行してインストールを実行します。
4. インストール後に Stack Builder を起動して、コンポーネントの追加インストールを実行します。
5. "Database Drivers" から "psqlODBC(32bit) V13.xxxxx" を選択して、インストールを実行します。Server(x64) 製品に対しては、"psqlODBC(64bit) V13.xxxxx" を選択します。
6. Windows の システムの詳細設定 を開き、［環境変数］ダイアログで、"Path" に 以下のパスを追加します。
   ＜psqlODBC のインストールパス＞¥bin
   デフォルト："C:¥Program Files (x86)¥PostgreSQL¥psqlODBC¥XXX¥bin" （xxx: バージョン番号）または、"C:¥Program Files¥PostgreSQL¥psqlODBC¥XXX¥bin"）

Magic xpa からデータベーステーブルを作成するには

第 17 章：「Magic xpa でデータベーステーブルを作成するには」（395 ページ）で説明されているように、Magic xpa でデータベースを作成することで、Magic xpa によって DBMS に自動的にテーブルを作成したり、構成を変更したりすることができます。



Magic xpa のこのような機能を有効にさせるには、**テーブルの存在チェック**特性をチェックする必要があります。この特性は、以下のようにして設定できます。

1. プロジェクト（またはアプリケーション）が開いている場合は、一旦閉じます。
2. **データベース**テーブル（**オプション→設定→データベース**）を開きます。
3. 設定を変更したいデータベースを選択します。
4. ***Alt+Enter*** を**クリック**して**データベース特性**を開きます。
5. **SQL** タブを**クリック**します。
6. **テーブルの存在チェック**特性を**チェック**します。

これで、データソース内で新しいテーブルを作成したり、既存のデータソースを変更した場合、**データベース**テーブルが DBMS 内で変更されるようになります。

既存のデータベーステーブルまたはビューにアクセスするには

データベーステーブルがすでに DBMS の中に存在している場合、Magic xpa で**定義取得**を行うことで自動的にデータソースを作成することができます。**定義取得**については、第 17 章：「既存のデータベーステーブルにアクセスするには」（400 ページ）を参照しています。

データベースに送信される SQL ステートメントを参照するには

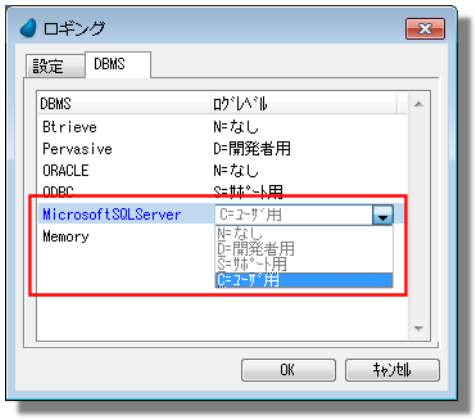
Magic xpa は背後で SQL ステートメントの作成／送信を行っていますが、実際に送信される SQL ステートメントを参照することができます。

- Magic xpa のデバッグ機能を有効にする
- 使用する DBMS のログ採取機能を使用する

２番目の方法は、DBMS に依存するためここでは説明しません。Magic xpa で SQL ステートメントを参照することは、以下で説明するように簡単に行うことができます。

Magic xpa 内での SQL ステートメントのロギング機能を有効にする

1. 最初に、ロギング機能を有効にする必要があります。**ロギング**テーブル（**オプション→設定→ロギング**）でこの設定を行います。
2. **設定**タブの**ゲートウェイ**を ***Yes*** に設定します。
3. **DBMS** タブに切り替え、利用する DBMS に対して必要なロギングレベルを設定します。
   - ***N= なし*** ：ロギングを行いません。
   - ***C= ユーザ*** ：発行される SQL コマンドのみを出力します。
   - ***D= 開発者*** ：MSE/MSJ の開発向けの最大レベルの情報を出力します。
   - ***S= サポート*** ：アプリケーション開発者用の情報を出力します。



Magic xpa で送信される SQL ステートメントを参照するだけであれば ***ユーザ***レベルで十分ですが、開発者またはサポートレベルを指定することで更に多くの情報を確認することができます。以下の出力例は、***サポート***レベルで出力されたものです。

```
,09484  session - 0, in use - 1, write - 1, reusable - 1, results pending - 0,not only for cursors - 1
,09500  ms7_trans(): OPEN_READ
,09500  ms7_trans(): <<<<< ctxID = -1.000000, retcode = 0
ﾄﾗﾝｻﾞｸｼｮﾝ読込 0
,09500  ms7_trans(): >>>>> ctxID = -1.000000, transmode = 8, db = 20
,09500  Hold thru ms7_trans(): >>>>> ctxID = -1.000000, serverID = 1, dbconn_hdl = 0
,09500  Hold thru ms7_trans(): <<<<< ctxID = -1.000000, errcode = 0, RC = RC_OK
,09500  session - 0, in use - 1, write - 1, reusable - 1, results pending - 0,not only for cursors - 1
,09500  ms7_trans(): <<<<< ctxID = -1.000000, retcode = 0
ﾄﾗﾝｻﾞｸｼｮﾝのｺﾐｯﾄ 0
```

4. 必要であれば、**デバッグ**モード（**デバッグ→デバッグモード**）を有効にします。
5. **アクティビティモニタ**（**表示→アクティビティモニタ**）を開きます。

これで、デバッガを起動しながらプログラムを実行することで、**アクティビティモニタ**に SQL ステートメントが表示されます。

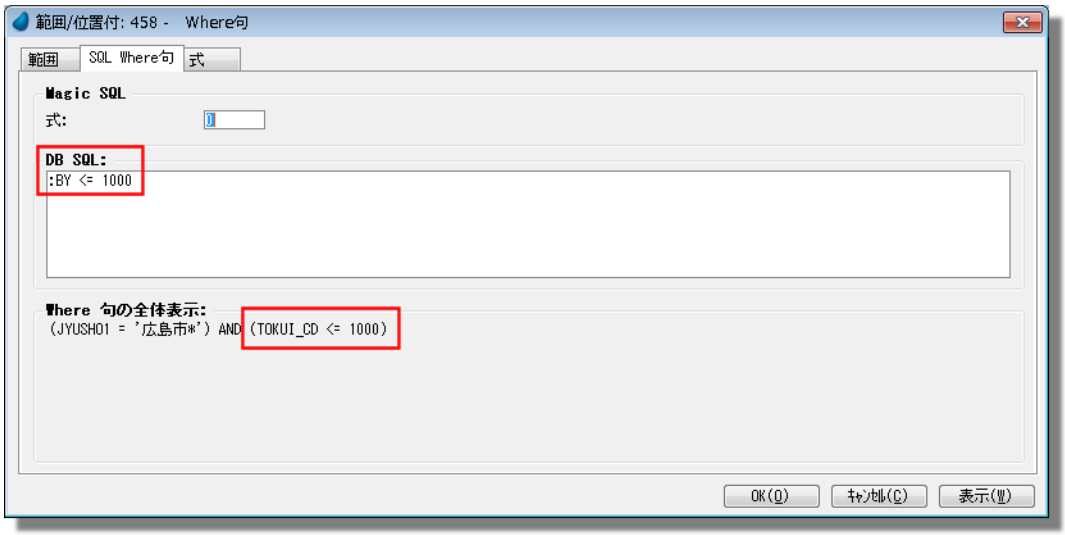
参照： 第 28 章：「デバッガを使用してデバッグするには」（613 ページ）を参照してください。

独自の SQL ステートメントをデータベースに送るには

SQL データベースを利用して Magic xpa を使用する場合、Magic xpa は必要な SQL コードを生成します。しかし、デフォルトのコード生成機能を無効にすることもできます。例えば、ストアドプロシジャを呼び出したり、レコードグループの取得方法をより詳細に制御したい場合に利用できます。

このようなことは、Magic プログラム内で簡単に行うことができます。SQL ステートメントの送信方法には、以下に示すようなものがあります。

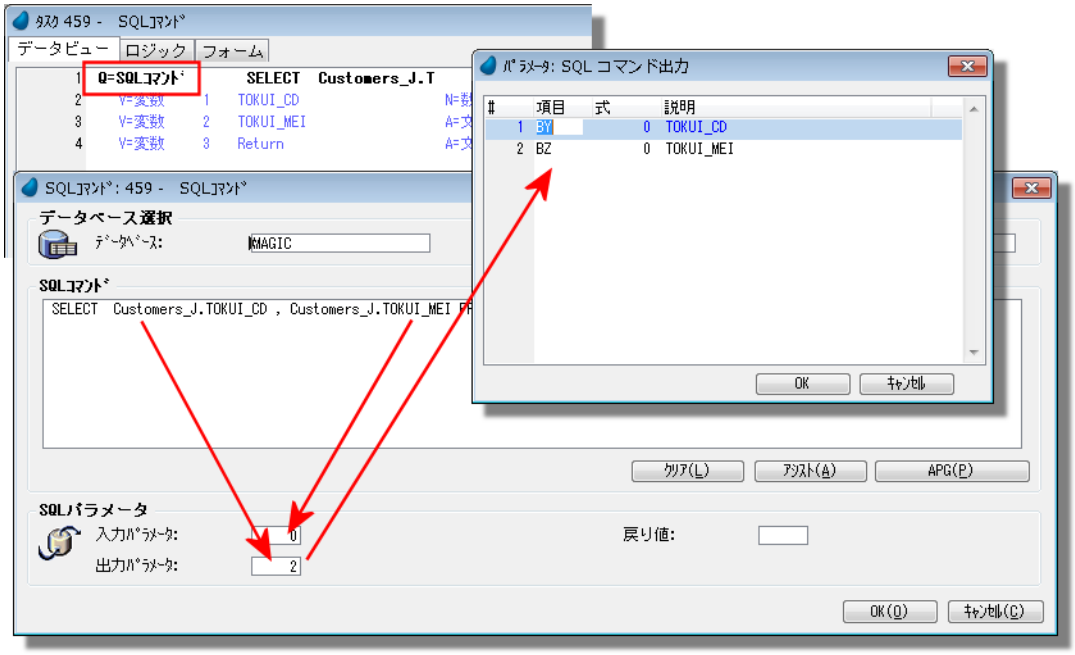
WHERE 句を手動で入力する



WHERE 句を送信したいだけの場合、**範囲 / 位置付**ダイアログを使用することで実現できます。**DB SQL** で Where 句ステートメントを入力することで、**Where 句の全体表示**に Magic xpa で生成される Where 句が表示されます。

1. **範囲 / 位置付**ダイアログ（**タスク→範囲 / 位置付**）を開きます。
2. **SQL Where** タブを**クリック**します。
3. **DB SQL** に直接入力したり、**式**から**ズーム**して**式エディタ**に必要な式を入力することで、任意の Where 句を入力します（**範囲**タブで範囲式を定義した場合も有効です）。
4. **Where 句の全体表示**に送信される Where 句の内容が表示されます。

他の SQL ステートメントを手動で入力する



WHERE 句ステートメントより複雑なものを入力したい場合、ダイレクト SQL ステートメントを使用することで実現できます。

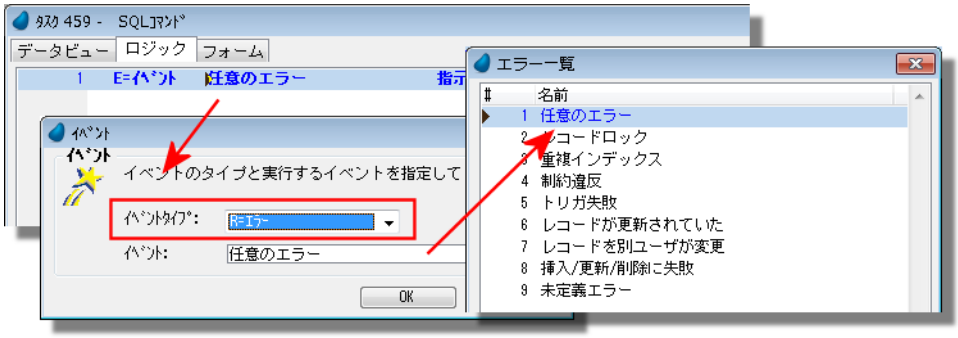
1. **データビューエディタ**を開きます。
2. １行目のヘッダ行で ***Q=SQL コマンド***を選択し、**Tab** 移動します。
3. **SQL コマンド**ダイアログが表示されます。ここで SQL コマンドを入力することができます。アシストボタンをクリックすることで SQL コマンドを入力を支援するための**アシスト**ダイアログが表示されます。
4. **出力パラメータ**から**ズーム**して、タスクに定義された変数項目と SQL コマンドで定義されたフィールドを割り当てます。

メインソースのカラム項目と同じように変数項目をプログラム内で使用することができます。

データベースエラーまたは例外を処理するには

プログラムがデータベースで発生するエラーを処理する場合、Magic xpa のデフォルト機能を使用しないで、プログラム内で独自のエラー処理を行うようにすることができます。**イベントタイプ**特性が ***R= エラー***と定義されている［イベント］ロジックユニットを使用することで実現できます。

エラーロジックユニットを作成する



1. ロジックエディタを開きます。
2. *Ctrl+H* を押下してヘッダ行を作成します。
3. *E* を入力して***イベント***を選択します。カーソルが次のカラムに移動します。
4. ズームしてイベントダイアログを開きます。
5. **イベントタイプ**で ***R= エラー***を選択します。
6. **イベント**からズームして取得したいエラーを選択します。
7. **指示**特性を設定し、Magic エンジンに対してどのようなエラー対応処理を行うかを指定します。エラー対応とは、エラーロジックユニットが実行された後にエンジンが実行する処理を意味します。***S= タスク特性に依存***を選択すると、**タスク特性**の**エラー発生時**特性（*Ctrl+P* →**データ**タブ）の設定内容にもとづいて動作します。***I= 無視する***や ***B= ロールバックして再起動***などの他のオプションを指定することで処理を明示的に指定できます。各エラーに対するサポートされたエンジンの動作。異なるエラー処理については、『リファレンスヘルプ』を参照してください。
8. DBMS の実際のメッセージを Magic xpa で自動的に表示させるには、**メッセージ**特性を *Yes* に設定する必要があります。これにより、**動作環境**（**オプション→設定→動作環境→動作設定**）の**メッセージの詳細表示**の設定内容を上書きすることができます。

いくつかの異なるエラーを処理することができまが、どのようなデータベースエラーが発生した場合でも処理させたい場合は、***任意のエラー***を選択します。

これで、選択されたエラーに対応するハンドラが定義できました。次に、このエラーを処理するための処理コマンドを追加することができます。

処理コマンドが何も定義されず、**指示**特性が ***I= 無視する***に設定されている場合、エラーが発生しても何も行われません。レコードの重複エラーを無視させたい場合など、重要でないエラーと認識した上であればこのような設定を行うこともできます。

ハンドラ内で行うことができる事の１つは、ユーザにエラーメッセージを表示したり、エラーメッセージをログに書き込んだりすることです。Magic xpa には、エラーに関する情報を表示したり保存するために使用できる **ErrDatabaseName()**、**ErrDbmsCode()**、**ErrTableName()**、および **ErrDbmsMessage()** などの一連の関数があります。

しかし、**メッセージ**特性を *Yes* に設定した場合、メッセージが表示された後でメッセージバッファはクリアされるため **DbERR()** と **ErrDbmsMessage()** 関数は空白を返します。従って、プログラム内で DBMS エラー情報を使用し、メッセージを表示させたい場合は、**メッセージ**特性を *No* に設定して**エラー**処理コマンドを使用して手動でメッセージを表示してください。

エンドユーザのアクセスとデータ操作を制限するには

APG を使用してプログラムを作成した場合、デフォルトでは、ユーザはすべてのカラムにアクセスし、すべてのレコードを変更することができます。しかし、タスクの様々な特性を変更することで Magic xpa によるテーブルデータのアクセスを制限させることができます。ここでは、テーブルへのアクセスを制限するためのオプションの設定方法について説明します。

- **初期モード**特性（**タスク特性**→**汎用**タブ）：プログラムの初期モードを設定します。照会と設定された場合、カラム内のデータを修正することができません。
- **オプション**（**タスク特性**）：ここには、一連の特性があり、*Yes*、*No*、または***式***が指定できます。例えば、**修正**が *No* に設定された場合、ユーザは、タスクを修正モードに変更することができなくなります。**削除**が *No* に設定された場合、レコードを削除することができなくなります。
- **アクセス**特性（**メインソース特性**または**リンク特性**）：この特性は、テーブルに対するアクセスモードを設定します。読込に設定された場合、テーブルは読込モードでオープンされ、タスク環境がどのように設定されていてもテーブルのデータは保存されません。他の特性によって修正が許可されていてレコードが不注意に更新しようとした場合、エラーメッセージが表示され、レコードは更新されません。
- 上記の設定を全く行わない場合は、タスク内に必要なカラムのみを定義するようにします。
- **照会モード時の更新許可**（**動作環境**→**システム**タブ）を *No* に設定します。タスクが照会モードの場合、データが入力できなくなります。この設定を、*Yes* にすると**項目更新**処理コマンドによるデータ修正ができるようになります。

データベースのレコードの取得順を指定するには

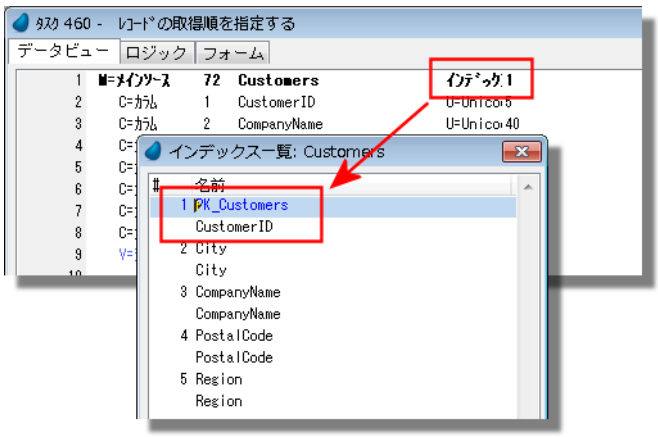
レコードをテーブル形式で表示させる場合、レコードが検索される順序が重要になってきます。

エンドユーザは、最も理解し易い方法レコードを参照することを望むため、レコードの並び順はユーザインタフェースを考える上で重要な要因です。また、レコードをフィルタリングする上で最も効率的な方法で検索する必要があるという点からも、レコードの並び順は機能面を考える上で重要な要因になります。

Magic xpa では、レコードの並び順を制御する上でいくつかの方法があります。そのうち以下の主要な２つの方法について説明します。

- テーブルのインデックス
- タスクソートの使用

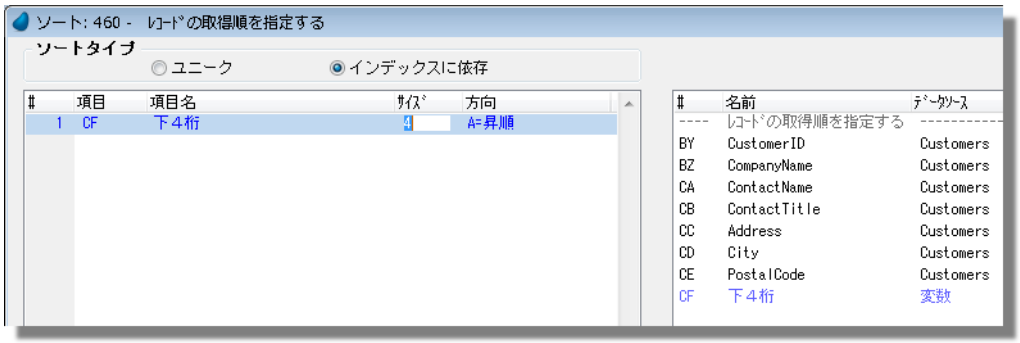
テーブルインデックスを設定する



メインソースやリンクテーブルをタスクで定義する場合、インデックス特性にて使用するインデックスを指定します。この例では、Customers というデータソースの PK_Customers インデックスを指定しています。

メインソースに対しては、使用する範囲条件に最も有効なインデックスを使用することで効率的に動作します。リンクされたソースに対しては、リンクで使用される位置付条件に最も有効なインデックスを使用することで効率的に動作します。この例では、Customer_ID が A001 から A003 までの範囲で検索する必要があるので、PK_Customers をインデックスとして使用することが最も効率的となります。

タスクソートを使用する



タスクソートは、最初にレコードが表示された後にその並び順を変更することを可能にする機能です。タスクソートを使用することでとても柔軟に表示させることができます。例えば、何らかの値で初期設定された変数項目を定義し、ソート処理時にこの変数項目を使用することができます。

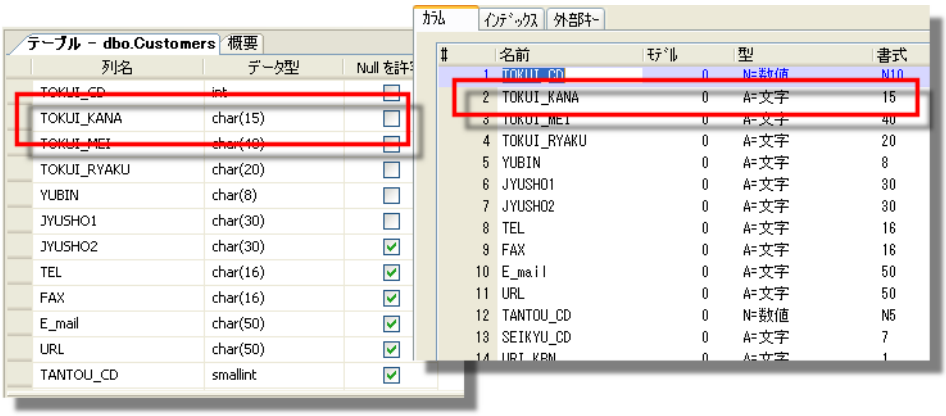
タスクソートを設定するには以下のようにします。

1. ソートテーブル（タスク→ソート、または *Ctrl+T*）を開きます。
2. 左側のカラムで **F4** を押下して１行追加します。
3. 右側のリストからソート処理で使用する項目を選択します。
4. 項目が長い場合、**サイズ**カラムの値を変更することでソートに使用する文字の長さを指定することができます。
5. ソートの順序を逆にしたい場合、**方向**カラムで降順を選択します。
6. 同様な方法で、必要な項目を選択します。

これで、タスクが実行されるとウィンドウに表示される前にレコードはソートされます。この例では、PostalCode の下 4 桁の文字によってソートされています。

参照 : 第 5 章 :「レコードの表示順を動的に変更するには」（81 ページ）
第 17 章 :「決められた順番でデータベーステーブルからレコードを取得するには」（410 ページ）を参照してください。

特定の SQL タイプにアクセスするには



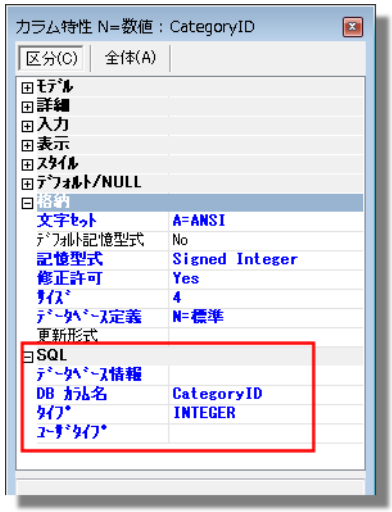
Magic xpa の各カラムは、データベース内の対応するデータを表しています。すべてのデータベースサーバは、独自のデータ型を持っています。しかし、Magic xpa はデータベースゲートウェイを使用して、それらのタイプを Magic xpa で処理できるように変換しています。

データベースサーバと Magic xpa とのデータのタイプの対応については、『リファレンスヘルプ』（データ管理 >SQL に関する考慮事項 > 構成とパフォーマンス >SQL ゲートウェイでサポートされるパラメータ）を参照してください。

しかし、既存のテーブルで定義取得を行った場合や、Magic xpa で作成されたテーブルの内容を参照することで、変換処理内容を確認することができます。

上記の例では、２つのテーブルが表示されています。左側は、MS-SQL Server の管理ツールにて **Customers** というテーブルのカラム内容を表示しています。右側では、同じテーブルを Magic xpa で定義取得した場合のカラムテーブルを表示しています。

各カラムの特性を見てみると、NULL の扱いなど、各カラムがどのように定義されているかを確認することができます。オリジナルのデータベース定義タイプは、**カラム特性**の **SQL** セクションのタイプ特性に設定されています。



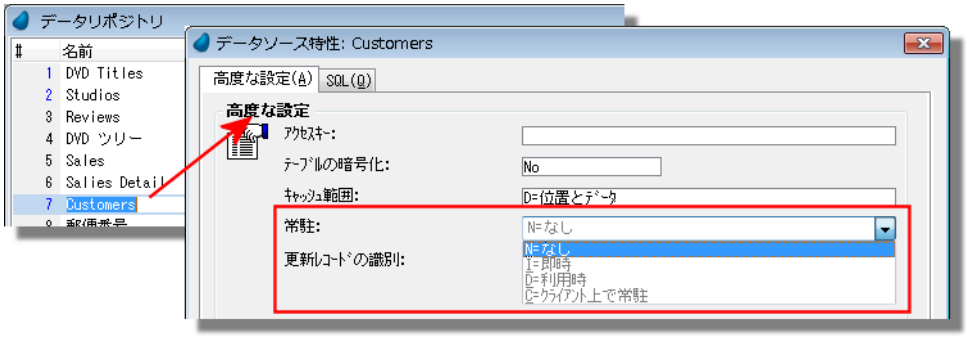
いくつかのカラムは特別な処理が行われます。DATETIME タイプのカラムは、Magic xpa で 2 つの個別の項目として処理できるように日付型と時刻型の２つのカラムに分割されます。しかし、データベースに対しては DATETIME カラムとして格納されます。

読込専用データのアクセスを最小化するには

データにアクセスするためのリクエストをデータベースサーバに送る必要があるため、データベースアクセスはプログラムの処理速度を落とす傾向があります。アクセス方法によっては、より時間がかかるものがあります。ロックされたデータを読み込む処理は、読込専用のデータを処理する場合より時間がかかります。

しかし、（参照用テーブルのように）頻繁にアクセスするが変更が少ないような場合、アプリケーションがオープンされた時点（または、テーブルが最初にアクセスされた時点）で一回だけ読み込むようにして、実行中は読み込んだ内容を利用し続けるようにすることができます。

これは、テーブルを***常駐テーブル***として定義することで実現できます。



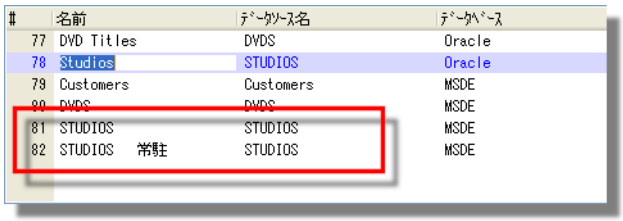
常駐テーブルの設定を変更する

1. データリポジトリで、変更したいデータソース上にカーソルを置きます。
2. *Alt+Enter* を押下してデータソース特性にアクセスします。デフォルトとして高度な設定タブが表示されます。
3. **常駐**特性で設定したいオプションを選択します。

ここには以下のオプションがあります。

- ***N= なし***：デフォルト設定です。タスクが起動されるたびにテーブルはオープンされます。
- ***I= 即時***：アプリケーションの起動時に、テーブルは 1 回だけオープンされます。このテーブルを使用するタスクは、最初にオープンされた時点でのデータを利用することができます。
- ***O= 利用時***：このテーブルを使用するタスクが最初に起動された時に、１回だけオープンされます。この後、他のタスクはオープンされた時点でのデータを利用することができます。
- ***B= クライアント上で常駐***：これはブラウザクライアントタスクでのみ有効です。テーブルの内容がクライアント側のブラウザにダウンロードされます。

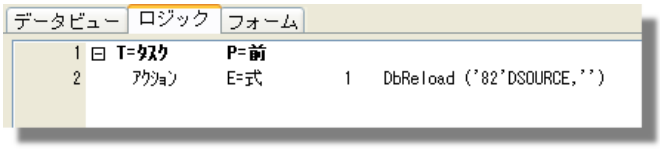
常駐テーブルの内容を更新する



データソースを常駐テーブルとして定義した場合、アプリケーション内のどのプログラムもそれを更新することができなくなります。

同じアプリケーション内でそのデータソースを更新したい場合は、データリポジトリでデータソースのコピーを作成し、このコピーの**常駐テーブル**特性を *No* に設定する必要があります。この例では、STUDIOS テーブルの 2 つのコピーが定義されています。一方は常駐テーブルで、他方は常駐にはなっていません。

次に、データソース **#81** を更新するプログラムを作成することで、このプログラムによる更新内容は STUDIO テーブルに格納されます。しかし、他のプログラムが使用している STUDIOS テーブルのコピー（データソース **#82**）は常駐テーブルとして設定されているため、この内容には反映されません。



この場合、**DBReload()** と呼ばれる Magic 関数を使用することで STUDIOS テーブルのコピーをリフレッシュする必要があります。**DBReload()** 関数は、データソース番号（とオプションでデータソース名）をパラメータとして指定します。**DBReload()** 関数が実行されると、メモリ上に読み込まれているデータソースの内容がリフレッシュされます。

データベースのアクセス頻度を減らすには

Magic xpa は自動的にデータベースへのアクセス処理を行うため、レコードをいつフェッチするかを指定するステートメントを記述する必要がありません。しかし、アクセス頻度に影響する要素を制御する必要があります。アプリケーションに最適な方法で Magic xpa がデータベースにアクセスする特性を設定することは、プログラムの処理速度に最も大きな影響の与える要因の 1 つになります。

ヒント : *テーブルのオープン中に、データベースへのアクセス回数を確認するには、デバッガやネイティブな SQL プロファイラーツールを使用することができます。プログラムがリソースを効率的に使用しているかどうかを確認するために、どのように処理されているかを確認することを推奨します。*

読込専用テーブル

アプリケーションが使用するテーブルの中には、主にコードの参照用として使用されるものがあり、これらは頻繁に更新されるようなことはありません。このようなテーブルに対しては、常駐特性のような特殊なアクセス特性を使用することができます。この特性については、「読込専用データのアクセスを最小化するには」（750 ページ）を参照してください。

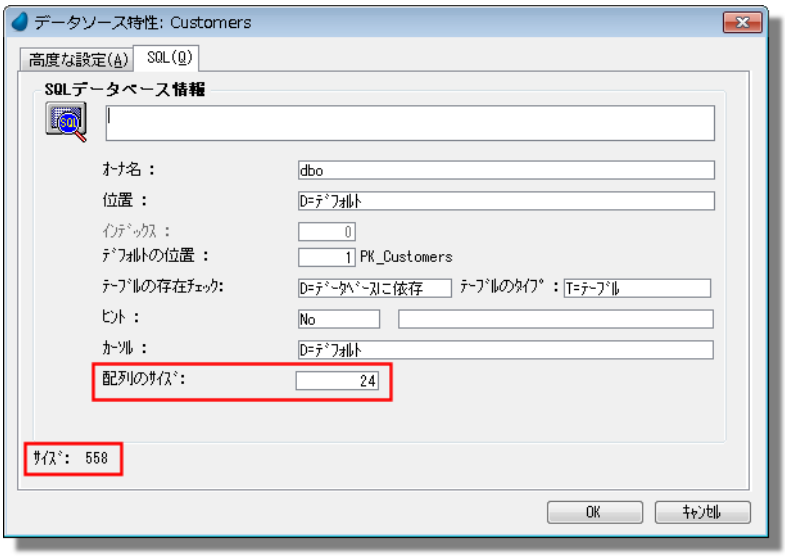
レコードのフェッチの制御

レコードからいつ、どれだけのレコードをフェッチするかを指定するための設定もあります。以下の設定内容について説明します。

- 配列サイズ
- キャッシュ
- ビューの事前読込
- 範囲やダイレクト SQL の使用

配列サイズ

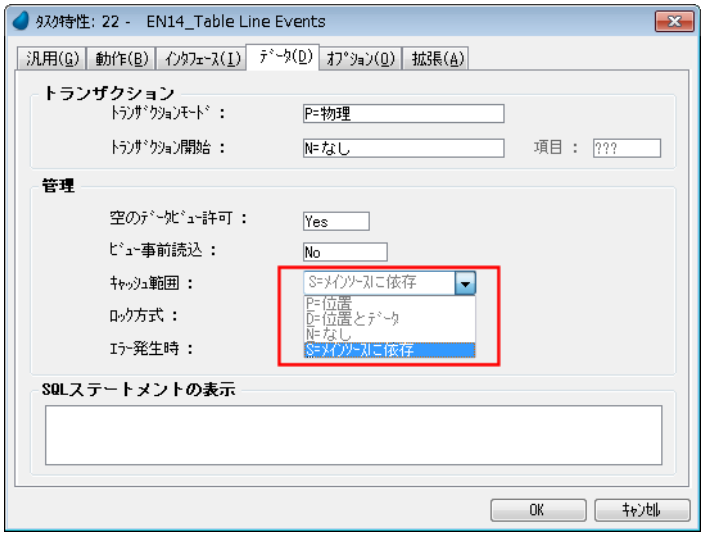
Magic xpa がレコードをフェッチする際、一度に全てのレコードをフェッチせず、ブロック毎にフェッチします。**データソース特性**の**配列サイズ**特性によって、一回にフェッチするレコード数を指定することができます。



配列サイズが *0* に設定された場合、**データベース特性**に設定された値がデフォルト値として使用されます。**データベース特性**の**配列サイズ**が *0* の場合も Magic xpa のデフォルトが使用されます。

Magic xpa のデフォルト値は、1200/ レコードです。これは、レコードが 200 バイト長の場合、6 つのレコードを一度にフェッチすることができます。SQL ファイルの場合、実際のレコード長はフェッチされるカラム数に応じて、実行時に変動します。

キャッシュ



データベースのアクセス頻度を減らすためにキャッシュを使用することができます。

キャッシュは、すべてのテーブルに対してオンラインタスクで使用することができます。バッチタスクでは、リンクテーブルのみ有効です。

メインソースのキャッシュは、**タスク特性**の**キャッシュ範囲**特性で設定します。この特性が***位置とデータ***の場合、Magic xpa はメインソースのアクセス時にデータベースからレコードをフェッチしません。

リンクテーブルのキャッシュは、**リンク特性**の**キャッシュ**特性で設定します。この特性が *Yes* の場合、Magic xpa は一度データベースからレコードを読み込み、同じタスク内で他のレコードをリンクする必要があった場合も、この読込内容を再利用します。

タスクがテーブルをオープンしている間は、キャッシュ内容は保持されます。

ビュー事前読込

ビュー事前読込特性（**タスク特性→データ**）は、タスクのウィンドウがオープンされる前にフェッチされるレコード数に影響します。この特性が Yes に設定されている場合、ウィンドウがオープンされる前に全てのコードがフェッチされます。この設定によって、スクロールバーを正確に表示させることが可能になります。しかし、レコード数が非常に多くユーザが最後までスクロールする事がないような場合、必要以上のレコードをフェッチする結果になってしまいます。

範囲とダイレクト SQL の使用

フェッチされたレコード数を減らすためには、データベースサーバ側でできるだけフェッチするレコード数を絞るように指定するようにしてください。

例えば、期限切れの総額が 12,000 ドルを越える全ての顧客レコードを印刷したい場合、フォーム出力処理コマンドの条件を使用することで実現することができます。しかしこの場合は、条件を評価するためにすべての顧客レコードをフェッチする必要があります。場合によっては、数百レコードのフェッチ処理が必要になるかもしれません。このような場合は、フェッチするデータに範囲条件を設定することで、Magic xpa は 1 つの SQL ステートメントを発行し、サーバは期限切れの総額が 12,000 ドルを越えるレコードのみを返すようにします。何人のユーザがお金を借りているかに依存しますが、場合によっては 1 回のフェッチ処理だけで済むことにもなります。

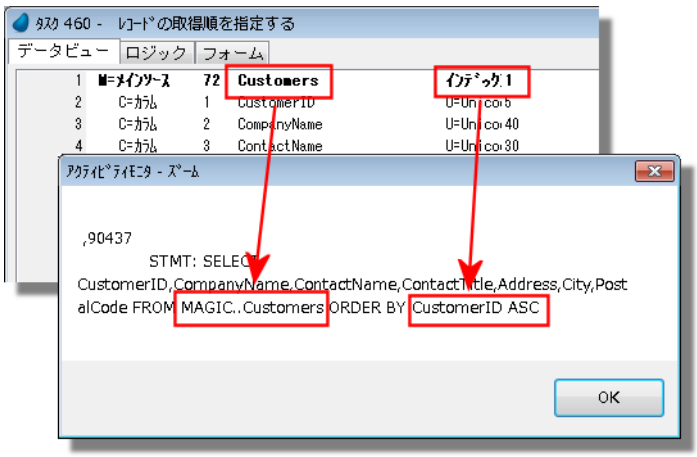
ダイレクト SQL ステートメントが、データを処理する方法として最も効率的に扱われるケースとして、インスタンスがあります。この詳細については、「独自の SQL ステートメントをデータベースに送るには」（744 ページ）を参照してください。

データベースに送られる SELECT ステートメントを制御するには

タスクを作成するために **Magic xpa Studio** を使用する場合、データベースへのクエリを迅速に記述するために Studio インタフェースを使用します。データベースゲートウェイが処理するため、使用する SQL、ISAM、XML、またはメモリテーブルに対して、実際どのようにアクセスするかを知る必要がありません。

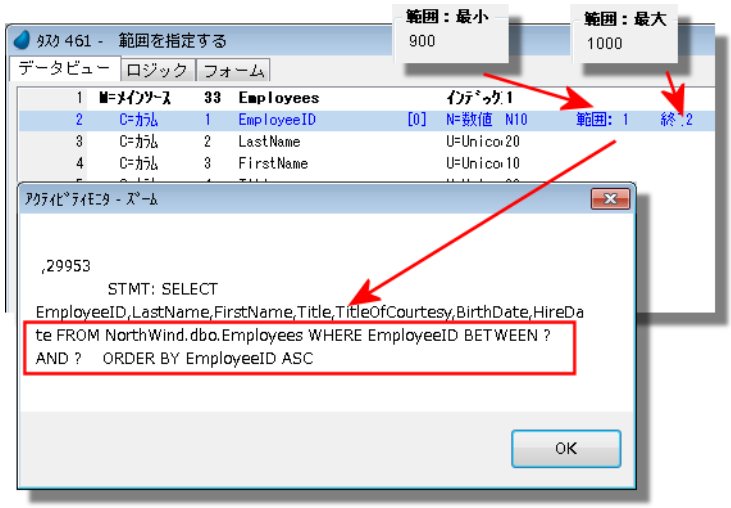
しかし、Magic プログラムの構成によって発行される SQL クエリが、どのように影響されるかは知っておく必要があります。

注： デバッガを使用することで、作成される SQL クエリを確認することができます。詳細は、「データベースに送信される SQL ステートメントを参照するには」（743 ページ）を参照してください。



SQL クエリに影響する要因を以下に列挙します。

- カラムの選択 …… 選択されたカラムはクエリに含まれ、データビューには選択された順番にカラムがフェッチされます。
- Magic xpa 側でのインデックスの指定（ユニークであるかどうかにかかわらず）…… インデックスは ORDER BY 句に変換されます。



- タスク範囲 …… 範囲条件は、Where 句として含まれます。
- タスクソート …… 定義されたタスクソートは、ORDER BY 句に変換されます。

そして、ダイレクト SQL ステートメントをクエリに追加することもできます。詳細は、「独自の SQL ステートメントをデータベースに送るには」（744 ページ）を参照してください。

複数のユーザが同じレコードを修正した場合の Magic xpa の動作を決定するには

Magic xpa は指定されたトランザクションの内容にもとづいて、修正されたレコードに対して異なる動作を実行します。

遅延トランザクションが最初に Magic xpa で処理され、次に個別の手順でデータベースにコミットする間、物理トランザクションが処理されます。

物理トランザクション使用時の修正行の識別

物理トランザクションを使用している場合、行がどのように識別されるかは使用する DBMS の設定に依存します。

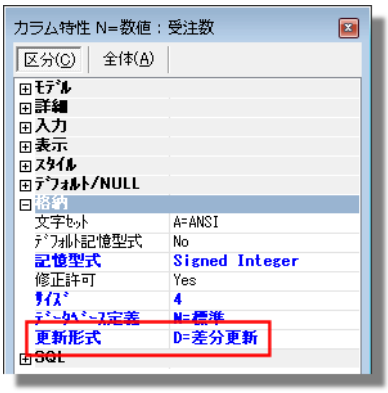
遅延トランザクション使用時の修正行の識別

最初に、Magic xpa は行が修正されたことを識別する必要があります。この処理は、メインソース特性の**更新レコードの識別**特性に依存します。この特性には、３つの異なるオプションがあり、設定内容によって以下のように動作が異なります。

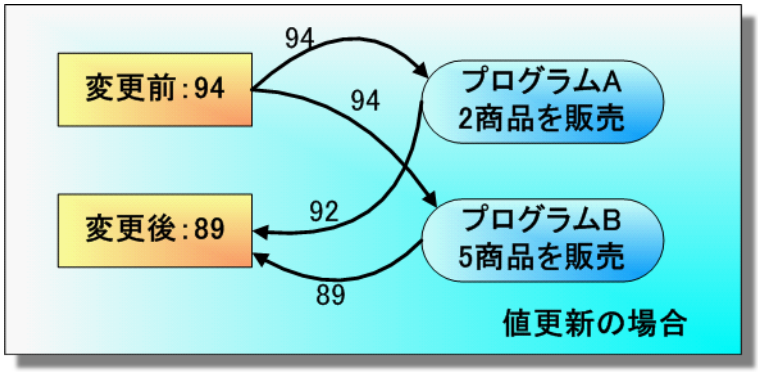
2 つのユーザ（ユーザ A、およびユーザ B）が SQL テーブルで同じレコードを更新している場合を例にとって説明します。どちらのユーザも更新のために同じレコードをフェッチします。「ユーザ B」が更新する前に「ユーザ A」がレコードを更新します。

- ***位置*** ：修正されたレコードのユニークな ID のみを Magic xpa が参照します。従って、ユーザ A によって書き込まれたレコードは、ユーザ B によるレコード更新によって上書きされます。ユーザ B による更新のみが保存され、エラーは発生しません。
- ***位置と更新項目*** ：修正されたレコードのユニークな ID と、どの項目が更新されたかを Magic xpa が参照します。従って、ユーザ A とユーザ B が異なる項目を更新した場合、両方の更新内容が保存されエラーは発生しません。もし同じ項目が更新された場合、ユーザ B に対してエラーが発生し、変更内容は保存されません。
- ***位置と選択項目*** ：修正されたレコードのユニークな ID と、（更新されたかどうかに関わらず）タスクで定義された項目を Magic xpa が参照します。従って、ユーザ A とユーザ B が実行しているタスクに同じ項目が定義されている場合、ユーザ B に対してエラーが発生し、変更内容は保存されません。
- ***メインソースに依存*** ：データ ソース特性の**更新レコードの識別**特性に依存します。

更新形式の効果



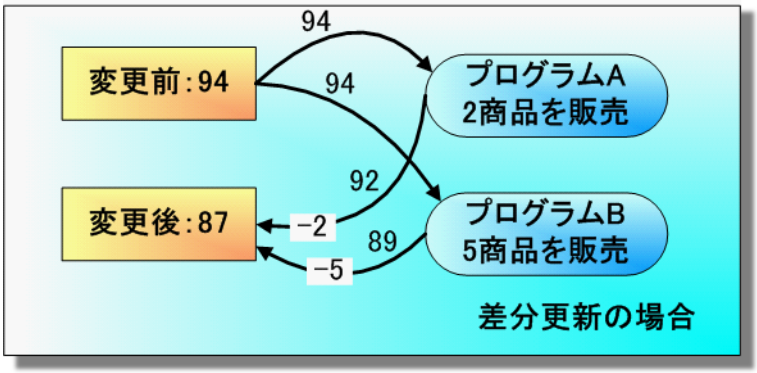
同じ数値データを更新する際に発生する問題を最小化するために**更新形式**特性を使用することもできます。



2 つのショッピングカート・プログラムを実行しているものとします。プログラム A は、在庫内に 94 の商品があることを確認しています。プログラム B も同じです。プログラム A は、2 つの商品を販売し在庫数を 92 に変更します。

プログラム B は、5 つの商品を販売したことで在庫数を 89 に変更します。この場合、現在の在庫数が 89 になり、正確な値ではなくなります。

しかし、データリポジトリのカラム特性で更新形式特性を差分に設定した場合、異なる更新結果になります。



これは、前述のようなプログラムを実行させることで確認することができます。

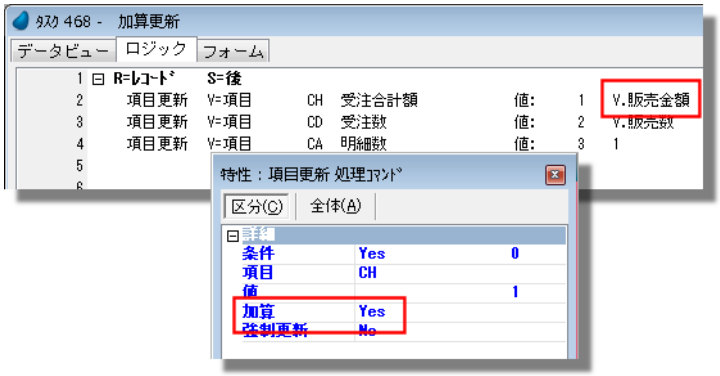
しかし、項目の内容を 92 または 89 に変更する代わりに、項目の値は、読み込まれた値と書き込まれる値の差分をもとに算出されます。従ってプログラム A は 92 という値を書き込む代わりに、項目の値を 2 だけ減算します。また、プログラム B は 89 という値を書き込む代わりに、項目の値を 5 だけ減算します。

プログラム A またはプログラム B は、プログラムを全く変更していません。データソースのカラム特性のみを変更しています。

差分更新と加算更新の違い

差分更新と加算更新は、まったく異なることに注意してください。差分更新はデータソースのカラム特性に設定するのに対して、加算更新は項目更新処理コマンドの特性に設定します。加算更新をタスクで使用する場合、実際の項目更新処理コマンドには異なる定義が行われます。通常、以下に示されたようなサブレコードを使用して実行した場合、合計の追加や、修正、削除を行うための明示的なロジックを定義する必要があります。

しかし、加算更新を設定することで、レコード後で自動的にすべて処理されます。開発者は、追加、または削除される値を指定するだけになります。



この例では、レコードが作成されると、V. **販売金額**は**受注合計額**に追加されます。レコードが削除されると、**受注合計額**から V. **販売金額**の値が削除されます。また、V. **販売数**の値が変更された場合、**受注数**は、変更された V. **販売数**の値にもとづいて変更されます。明細レコードの合計を算出する方法として、加算更新は「スマートな加算方式」といえます。

データベース制約がある場合に、1 対多の関係を実現するには

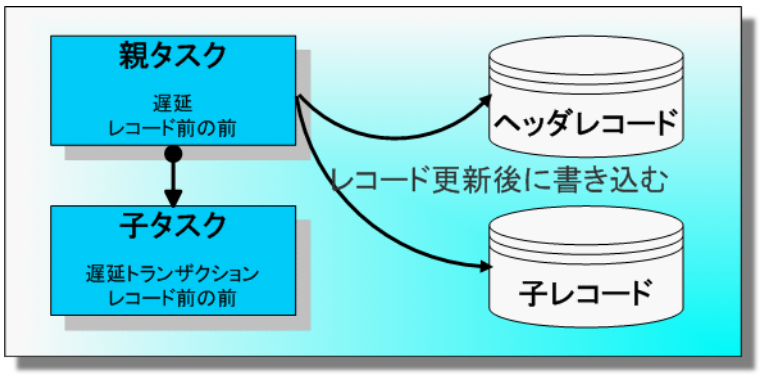
サブタスクが子レコードを表示し更新しながら、親タスクが 1 レコードを表示し更新するようなフォームを定義する場合があります。この場合、子タスクに制御が移っている時に、親レコードは実際にデータベーステーブルに書き込みを行っていない事があります。

このようなことを回避するために、子タスクに制御が移る前に親レコードをコミットする必要があります。以下の方法で、簡単にこのよな処理を実現させることができます。

- 親タスクの**トランザクションモード**特性を、*D= 遅延*または *W= 親と同一*に設定します。
- 子タスクの**トランザクションモード**特性を、*D= 遅延*または *W= 親と同一*に設定します。
- コール処理コマンド上で、**同期**特性を *Yes* に設定します。

コール処理コマンドではなくサブフォームを使用している場合、設定すべき**同期**特性はありません。しかし、親子タスクが、上記の説明のように設定された場合、**同期**特性が *Yes* と設定されたかのように、サブフォームタスクが自動的に呼び出されます。

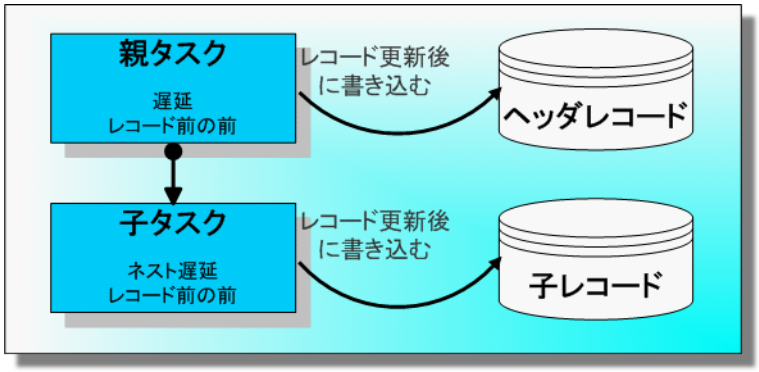
ネストされたトランザクションを実現するには



遅延トランザクションを使用している場合、あるタスクがトランザクションをオープンし、さらにそのタスクがトランザクションをオープンしているタスクを呼び出すような場合があるかもしれません。例えば、**トランザクションモード**特性が ***D= 遅延***で、**トランザクション開始**特性が ***P= レコード前の前***に設定された親タスクがあり、同じ設定の子タスクを起動したとします。

しかし、このような場合のコミット処理は、親タスクのレベルで同時に行われます。子タスクのレベルでロールバックをした場合、子タスクでの変更処理と同様に親タスクでの変更内容もロールバックされます。親子タスクは同じトランザクションを共有しています。

子タスクでの更新内容を親タスクとは独立して扱いたい場合は、以下のような***ネスト遅延***と呼ばれる異なる種類のトンザクションを使用する必要があります。



ここでは、親子タスクの両方で作成された更新内容をキャッシュします。子タスクのトランザクションは、子タスクが終了する前にコミットされます。ロールバックが子タスクで実行された場合、子タスクのトランザクションだけがロールバックされます。親タスクのトランザクションは、親タスク内で個別に処理されます。

ここでは、ネイティブな DBMS 内でのネストトランザクションを実行しているわけではありませんが、同じような結果になります。

注： トランザクションタイプ特性の ***W= 親と同一***は、物理または遅延トランザクションのどちらかを使用することで子タスクでも同じタイプを使用することができるようになりますが、ネストはされません。子タスクを ***W= 親と同一***に設定した場合、最初の例のように ***D= 遅延***に設定されたように動作します。

データベースに現行レコードを強制的に書き込むには

通常、オンラインタスクでは、次のレコードに移動したりタスクを終了することで、現在のレコードから抜ける際に現在のレコードをデータベースに書き込みます。しかし、ユーザがタスクを抜ける前にレコードを保存したい場合もあります。

例えば、現在のレコードの内容を印刷するために印刷ボタンを画面に追加するような場合を考えてみます。印刷処理の対象に現在のレコードを含める場合、現在のレコードが処理の実行前にコミットされている必要があります。これは、**レコード書込**の内部イベントを使用することで簡単に実現できます。

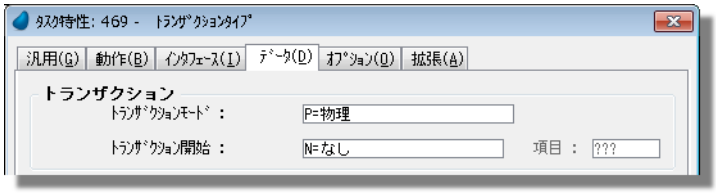
ここでは、印刷処理ハンドラをもとに説明します。これは、受注書印刷プログラムを呼び出す前に**レコード書込**イベントを発行する**ロジックユニット**です。これによって、受注書印刷プログラムが呼び出される前に、レコードが書き込まれることが保証されます。

レコード書込を使用する場合、タスクがアイドル状態（ユーザの操作が行われていない状態）になっていることを確認する必要があります。また、**ウェイト**特性が *No*（**イベント実行**処理コマンドのデフォルト）に設定することを推奨します。

***ヒント :** レコード書込を明示的に発行する代わりに、ユーザイベントの**強制終了**カラムを **P= レコード更新後**に設定することでも同じような結果になります。この設定によって、イベントが発行される前に現在のレコードが強制的に書き込まれます。*

トランザクションをオープンしないようにするには

デフォルトでは、ほとんどのタスクはトランザクションをオープンします。しかし、トランザクションをオープンしないようにしたい場合は、以下のように設定します。



オンラインタスクの場合

1. **タスク特性**（*Ctrl+P*）を開きます。
2. **トランザクションモード**特性を、***P= 物理***に設定します。
3. **トランザクション開始**特性は、***N= なし***に設定します。

ブラウザタスクの場合

1. **タスク特性**（*Ctrl+P*）を開きます。
2. **トランザクションモード**特性を、***N= なし***に設定します。

明示的にトランザクションをロールバックするには

トランザクションを持つ背景にある主な考えとして、ロールバックが可能であるということが挙げられます。つまり、何か発生した場合、データをコミットしないようにすることができます。データベースエラーが発生した場合など、何らかの状況下で xpa は自動的にトランザクションをロールバックします。また、**Rollback()** 関数を使用することでイベントやデータ条件にもとづいたトランザクションのロールバックを手動で行うこともできます。

関数の詳細は、『リファレンスヘルプ』に記載されていますが、基本的に以下のように動作します。

構文：

Rollback(*Message?, NestingLevel***)**

パラメータ：

- *Message?*：'TRUE'LOG を設定すると、ユーザ用の確認ダイアログが表示されます。
- *NestingLevel*：どのレベルまでロールバックするかを示す数値を指定します。
  - *1*…最も内側のネストトランザクションをロールバックします。
  - *2*…最も内側のすぐ上位のトランザクションをロールバックします。
  - *0*…最も外側のトランザクションをロールバックします。

データベースオプティマイザの動作を制御するには

通常、SQL データベースを使用して動作させる場合、データベースオプティマイザはステートメントの構文解析方法や、どのインデックスの使用するかを決定します。ほとんどの状況下で、オプティマイザはよい決定をし、可能な限り効率的に検索します。

しかし、オプティマイザの動作を変更するための機能がデータベースオプティマイザには組み込まれています。ヒントを SQL ステートメントに送ることで、これを可能にします。ヒントの正確なコード内容は、使用する SQL エンジンによって異なります。

Magic xpa では、３つの異なるレベルで SQL ヒントを設定することができます。

- データベース：データベースレベルでヒントを設定した場合、ヒントはそのデータベースに対応した全ての SELECT ステートメントに含まれます。
- データソース：データソースレベルでヒントを設定した場合、ヒントはそのテーブルの送られるすべての SELECT ステートメントに含まれます。
- インデックス：インデックスレベルでヒントを定義した場合、ヒントはそのテーブルとインデックスに送られるすべての SELECT ステートメントに含まれます。

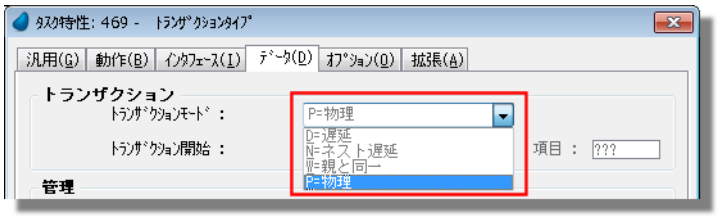
これらのレベルの各特性ダイアログ（*Alt+Enter*）の **SQL** タブ上に**ヒント**特性があります。さらに、SQL タスクでは、ヒントをコード化して指定することができます。

データベーストランザクションを初期化するには

Magic xpa には、トランザクション処理が組み込まれており、デフォルトで自動的に初期化されます。この機能を効果的に使用する方法について理解することが重要です。

ここでは、Magic xpa がどのようにデータベーストランザクションを設定するかについての基本的な説明を行っています。詳細は、『リファレンスヘルプ』を参照するようにしてください。また、プログラム内でトランザクション処理がどのように実行されているかが明確になっていない場合は、Magic xpa のログファイルやネイティブなデータベースのログファイルをチェックしてください。

トランザクションのタイプの定義



トランザクションモード特性は、各タスクの**タスク特性**（*Ctrl+P*）の**データ**タブで設定します。オンラインまたはバッチタスクの場合、以下の 4 つのオプションがあります。ブラウザタスクの場合は、***P= 物理***の代わりに ***N= なし***を選択することができます。

基本的には、３種類の異なるタイプのトランザクションがあります。

- ***P= 物理*** ：全てのトランザクション処理は、使用する DBMS 側に依存しています。
- ***D= 遅延*** ：Magic xpa 側で処理されます。Magic xpa はデータ操作のステートメントをキャッシュし、必要であればロールバックを行います。データがコミットされたら、Magic xpa はデータ操作ステートメントを DBMS サーバに送ります。
- ***N= なし*** ：トランザクションをオープンしません（詳細は、「トランザクションをオープンしないようにするには」（762 ページ）を参照してください）。

他の２種類のトランザクションのタイプは、物理と遅延の拡張です。

- ***N= ネスト遅延*** ：遅延トランザクションの特別なタイプです（詳細は、「ネストされたトランザクションを実現するには」（760 ページ）を参照してください）。
- ***W= 親と同一*** ：親タスクでトランザクションが使用されていても、物理または遅延トランザクションを開始します。

従って、Magic タスクを設定する場合に最初に決めることは、トランザクションモードを ***P= 物理*** か、***D= 遅延*** か、***N= なし***のどれにするかになります。遅延トランザクションは、多くの柔軟性があり、遅延トランザクションを使用する場合でのみ利用できるいくつかの機能（ネストトランザクションなど）があります。ブラウザタスクを使用する場合、物理トランザクションは選択できません。

トランザクションツリー

どのタイプのトランザクション処理を使用するか決める場合に、プログラムのツリー構造に留意する必要があります。トランザクションが子タスクでどのように動作するかは、親タスク側のトランザクション設定に依存します。

ここでは、1 つのヘッダレコードと３つの子レコードを表示する受注画面を考えてみます。ヘッダレコードを変更すると、すべての３つの子レコードが変更されます。

最後の子レコード上でパークしている時に、子レコードのレベルでロールバックをします。そして、親レベルでロールバックをする処理を繰り返します。ここに、各設定に対する結果が表示されています。（全てのトランザクションが***レコード前の前***に設定されています）。

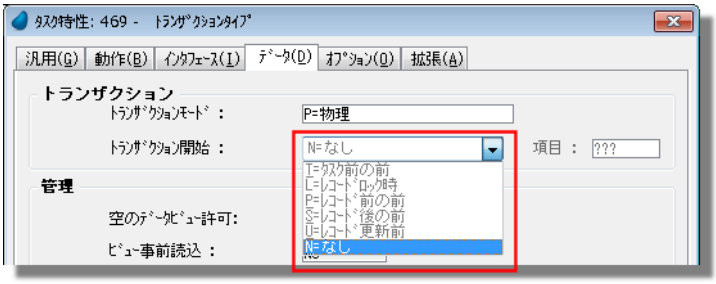
親タスク	子タスク	結果	親でロールバック	子でロールバック
遅延	遅延	子タスクの変更は、親タスクでの変更の一部と見なされます。これらはまとめてコミットされるかロールバックされます。	親タスクの変更と子タスクの全ての変更がロールバックされます。	親タスクの変更と子タスクの全ての変更がロールバックされます。
遅延	親と同一	遅延 - 遅延　の場合と同じです。		

親タスク	子タスク	結果	親でロールバック	子でロールバック
遅延	ネスト遅延	各子タスクの変更と各親タスクの変更は独立しているものと見なされます。ユーザがそのレコードから抜けた場合、各レコードはコミットされます。	親タスクの変更（まだレコード上にカーソルがあるためコミットされていません）がロールバックされます。	最後のコミットされていない子タスクの変更だけがロールバックされます。
遅延	物理	各子タスクの変更と各親タスクの変更は独立しているものと見なされます。ユーザがそのレコードから抜けた場合、各レコードはコミットされます。	親タスクの変更（まだレコード上にカーソルがあるためコミットされていません）がロールバックされます。	最後のコミットされていない子タスクの変更だけがロールバックされます。
物理	物理	子タスクの変更は、親タスクでの変更の一部と見なされます。これらはまとめてコミットされるかロールバックされます。	親タスクの変更と子タスクの全ての変更がロールバックされます。	親タスクの変更と子タスクの全ての変更がロールバックされます。
物理	親と同一	物理 - 物理と同じです。		
物理	遅延	エラー		

親タスクでトランザクションがオープンされた場合、子タスクは独自のトランザクションをオープンせず同じトランザクションを共有することは通常よく行われます。例外としては、遅延トランザクションが設定された親タスクが、物理トランザクションかネスト遅延に設定された子タスクを起動する場合です。

また、物理トランザクションとして設定された親タスクが、遅延トランザクションとして設定された子タスクを起動した場合、エラーが発生します。（タスクが複数の異なるプログラムから呼び出される可能性が考えられるため）親プログラムのトランザクションモードがどのように設定されているか確認できない場合、子タスクでは ***W= 親と同一***を設定することが最も安全な方法です。この場合、物理または遅延トランザクションのどちらかを使用する親タスクと同じトランザクションモードで実行されます。

トランザクションの開始と終了の場所の定義

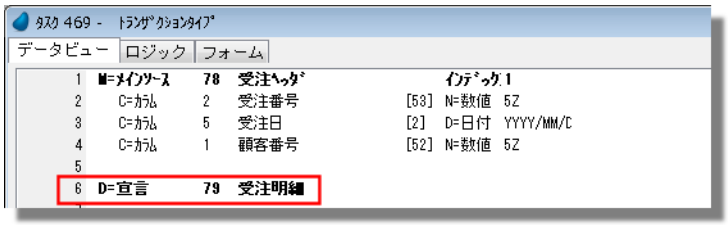


トランザクション開始特性は、トランザクションをいつオープンするかを指定します。物理トランザクションの場合、図のような６つのオプションがあります。遅延トランザクションの場合は、***T= タスク前の前***と ***P= レコード前の前***のみ選択できます。

トランザクションは、開始時と同じレベルで終了します。トランザクションがタスクレベルで開始された場合、タスクの終了時にそれは終了します。トランザクションがレコードレベルで開始された場合、ユーザがそのレコードを抜けた時点で終了します。

しかし、前のセクションで説明したように、現在のタスクが親タスクのトランザクションを共有している場合があります。この場合、トランザクション開始の設定に関係なく子タスクの実行中はオープンされ続けます。

関係するテーブルの定義



タスクツリー上の親タスクでトランザクションを開始している場合、トランザクションに関与する可能性のあるすべてのテーブルが親タスクで宣言されていることを確認する必要があります。親タスクでテーブルを宣言することで、上位レベルでテーブルがオープンされるため、トランザクションの設定を行うことでデータアクセスがより早くなります。

複数行を更新するとき、ロックされた行をスキップするには

行のロックエラーが発生した場合、バッチタスクの処理を継続させることができます。ロックされた行をスキップして、ロックされない行のみ処理するためには以下のような方法があります。

レコードレベルのトランザクションを使用する

エラーが発生すると、自動的にロールバックされます。トランザクションをレコードレベルに設定することで、ロックされたレコードのトランザクションのみロールバックさせることができます。

タスクの［エラー発生時］特性（[タスク特性]）を「復旧」に変更する

これは、行がロックされた後でのレコード処理を可能にするために必要です。「アボート」（バッチタスクのデフォルト）に設定されたタスクでは、エラーが発生するとタスクは停止します。

［エラー］ロジックユニットの［指示］特性を「無視」に設定する

エラーが発生しても、エラー処理は何も行われず、タスクの処理が継続されます。

第 36 章 : ブローカ

Magic xpa がリクエストを受信できるように Web サーバ（IIS）を設定するには

MRB（Magic　Request Broker）を使用して動作させる前に、Web サーバを適切に設定する必要があります。設定するには、以下のような作業を行います。

1. Magic xpa をインストールする前に、あらかじめ IIS Web サーバをインストールしておく必要があります。
2. Magic xpa のインターネットリクエスタを一緒にインストールします。
3. Magic xpa のインストールの後にインストール内容を確認し、Web サーバが動作していることを確認します。

これらの手順を順に説明します。

1.Microsoft IIS をインストールする

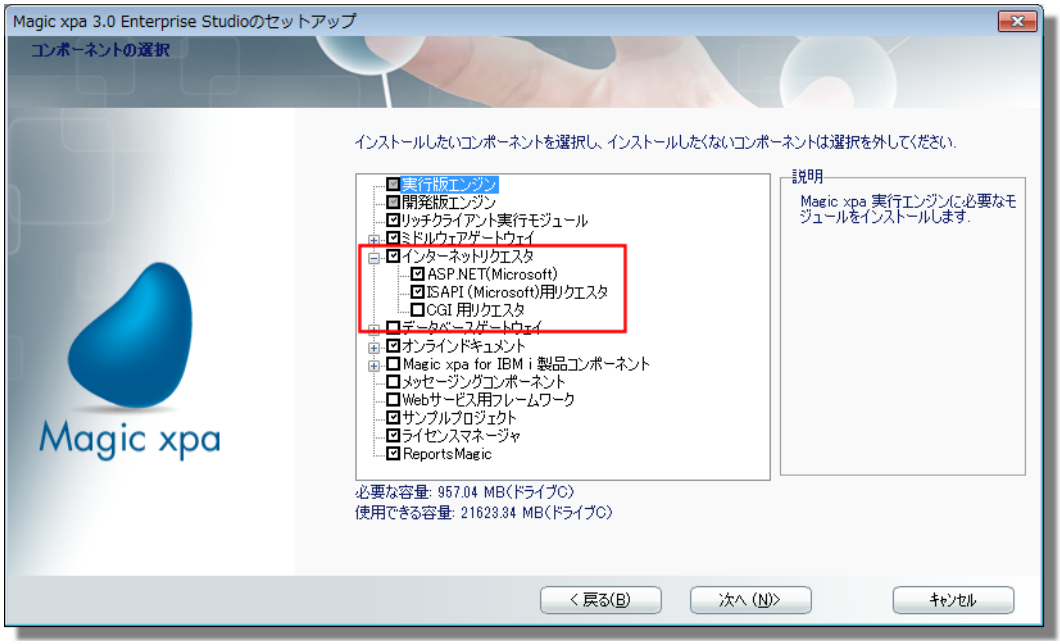
IIS（Internet Information Services）は、Web サーバとして動作する Windows OS のコンポーネントです。コンピュータの管理ウィンドウ（**スタートメニュー→設定→コントロールパネル→管理ツール**）を参照することで確認することができます。

IIS がインストールされていない場合、Windows コンポーネントウィザード（**スタートメニュー→設定→コントロールパネル→プログラムの追加と削除→ Windows コンポーネントの追加と削除**）を使用してインストールすることができます（その際、OS のインストール CD が必要になります）。

Windows 内に IIS が既にインストールされている場合は、Magic xpa のインストール時に自動的にインターネットリクエスタのセットアップを行うように選択されます。その際、リクエスタのコピー先のエイリアスも自動的に設定されます（カスタムインストールにて変更することもできます）。

ヒント : *IIS のインストール方法などは、使用する Windows のバージョンによって異なる場合があります。*

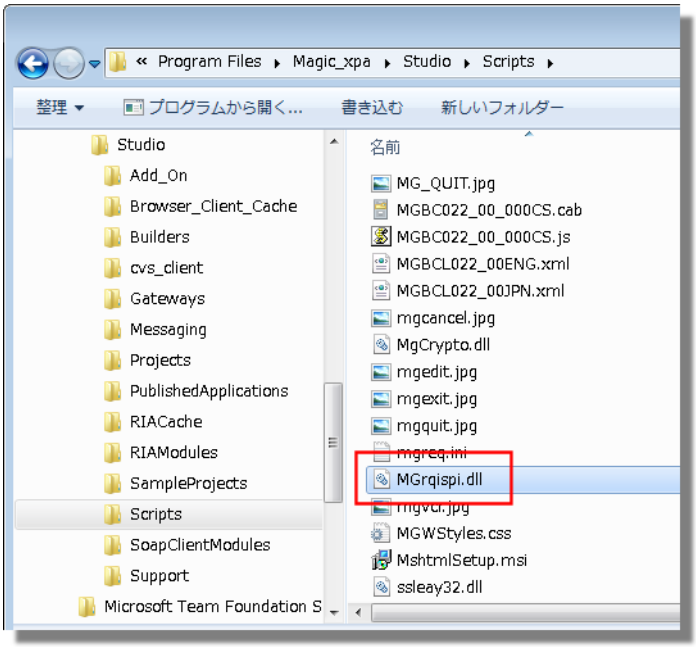
2.Magic xpa をインストールする



次に、Magic xpa をインストールし、インターネットリクエスタがセットアップされることを確認します。Microsoft IIS の場合は ISAPI リクエスタのチェックボックスをチェックし、Apache の場合は CGI リクエスタのチェックボックスをチェックします。

すでに Magic xpa がインストールされている状態で、リクエスタトタのみ追加したい場合は、以下のようにします。

1. **コントロールパネル→プログラムの追加と削除→ Magic xpa Enterprise Studio**（または、**Magic xpa Entereprise Server** や **Magic xpa RIA Server**）**→変更と削除**を**クリック**します。
2. 修正のインストールオプションを選択します。
3. 前述のように、ISAPI リクエスタのチェックボックスをチェックして追加します。
4. 他のオプションのチェックを外さないようにしてください。



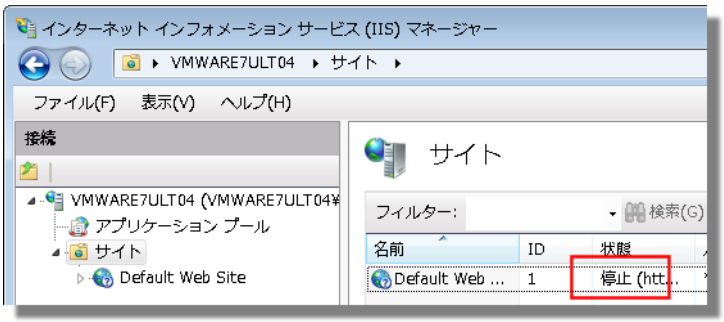
リクエスタが適切にインストールされると、Magic xpa の Scripts サブディレクトリ内に MGrqispi.dll というファイルがコピーされます。これは、IIS 環境下で実行されるインターネットリクエスタと呼ばれる DLL です。

3.MRB が実行していることを確認する

Magic xpa のインストール方法によって、MRB はサービスまたは実行形式のどちらかで実行されます。Windows のサービスとして実行する場合、OS の起動時に自動的に開始されます。実行形式でインストールされた場合は、手動で起動さ

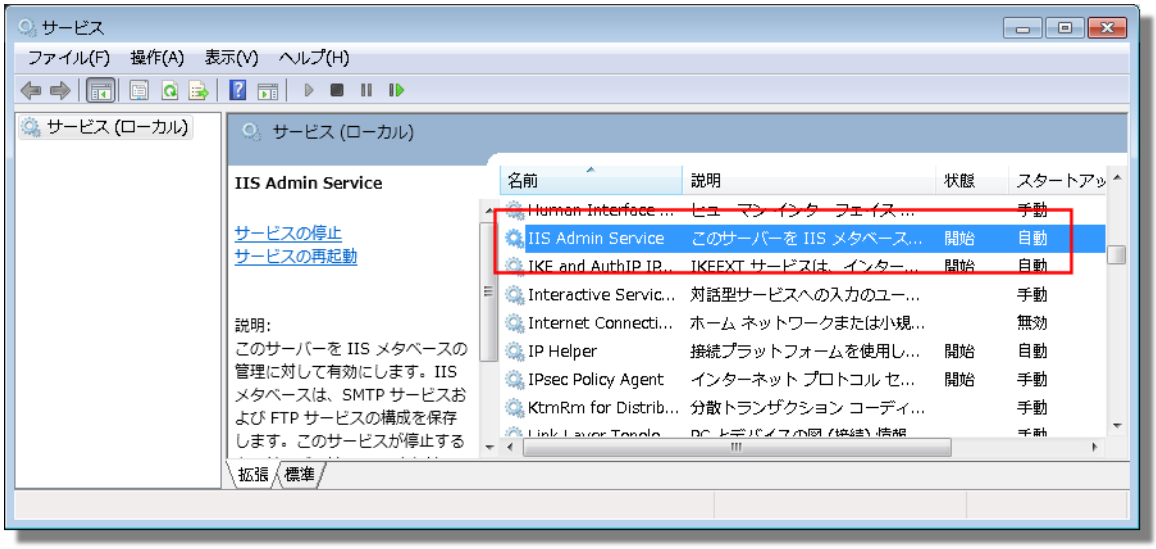
せる必要があります。MRB の起動／停止は、ブローカとリクエスタメニュー（**スタートメニュー→ Magic xpa Enterprise Studio**）上にあります。

Broker モニタを使用することで MRB の実行状態を確認することができます。モニタに関する詳細は、「MRB の動作を監視するには」（775 ページ）を参照してください。



Web サーバの実行状態は、IIS サービスの管理ウィンドウで確認することができます。サービスが停止している場合、起動する必要があります。コンテキストメニューから Web サイトの起動を選択することができます。

4.Web サーバが実行していることを確認する



OS のサービス設定によって、PC の起動時に Web サーバを自動的に起動させるかどうかを指定することができます。IIS の場合、サービスウィンドウ（**スタートメニュー→コントロールパネル→管理ツール→サービス**）で設定することができます。

Magic xpa がリクエストを受信できるように Apache Web サーバを設定するには

ここでは、Magic xpa 用に Apache サーバを設定するために必要な手順を説明しています。

1. Magic xpa をインストールする前に、あらかじめ Apache サーバをインストールしておく必要があります。
2. Magic xpa のインターネットリクエスタをインストールします。
3. Apache のディレクトリを設定します。
4. インストール後にインストール状態を確認し、サーバが動作していることを確認してください。

これらの手順を順に説明します。

注：　ここでは Apache2（Ver2.n.n）を前提に説明しています。Apache1（Ver1.n.n）の場合は、インストール方法や起動方法等が異なります。

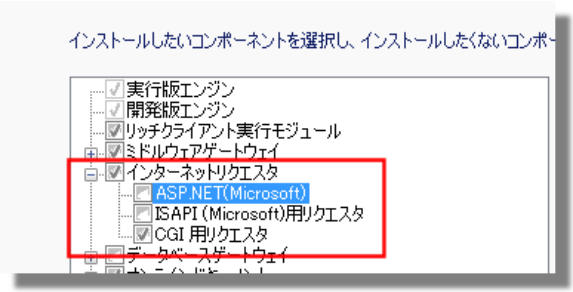
1.Apache サーバをインストールする

Apache サーバをダウンロードし、インストールします。http://www.apache.org から必要なバージョンをダウンロードすることができます。Apache Projects の左下の HTTP Server を選択してください。このサイトには、Windows セキュリティに関する問題をどのように扱うかなどを含めた、さまざまなインストール情報があります。

ダウンロードされたファイルでインストールします。表示されるメッセージに従って操作してください。Network Domain と Server Name の指定欄では、開発用 PC の場合、localhost とだけ入力してください。Apache がインストールされると、PC のシステムトレイに小さな Apahce アイコン（羽）が表示されます。このアイコンをクリックすることで、Apache サービスを管理したり監視することができます。

Apache のインストール内容を確認するには、Web ブラウザを開き、http:\\localhost という URL を入力してください。

2.Magic xpa のリクエスタがインストールされていることを確認する



次に、Magic xpa をインストールし、インターネットリクエスタがセットアップされることを確認します。Apache の場合は、CGI リクエスタのチェックボックスをチェックします。

すでに Magic xpa がインストールされている状態で、リクエスタトタのみ追加したい場合は、以下のようします。

1. **コントロールパネル→プログラムの追加と削除→ Magic xpa Enterprise Studio**（または、**Magic xpa Entereprise Server** や **Magic xpa RIA Server**）**→変更と削除をクリック**します。
2. 修正のインストールオプションを選択します。
3. 前述のように、CGI リクエスタのチェックボックスをチェックして追加します。
4. 他のオプションのチェックを外さないようにしてください。

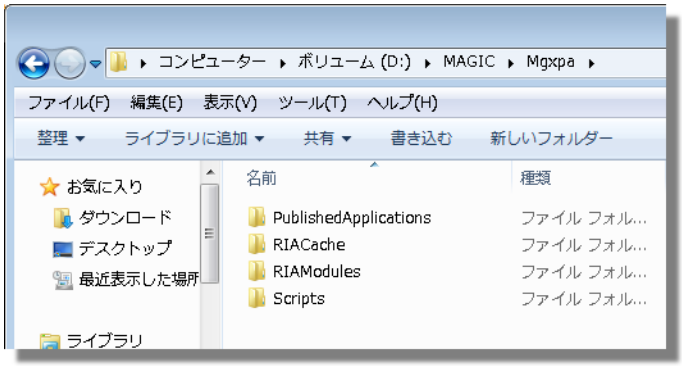
リクエスタが適切にインストールされると、Magic xpa の Scripts サブディレクトリ内に MGrqcgi.exe というファイルがコピーされます。

3.Apache のディレクトリを設定する

次に、Apache 用のディレクトリ環境を設定する必要があります。以下の手順で行います。

1. スクリプト用のディレクトリを作成します。
2. これらのディレクトリが認識できるように Apache のコンフィギュレーションファイルを修正します。

3. Magic.ini でこれらのディレクトリを定義します。



ディレクトリの作成

リッチクライアントアプリケーションを実行させる場合は、以下の４つのディレクトリを作成する必要があります。

- スクリプト用
- RIA モジュール用
- RIA キャッシュ用
- RIA アプリケーション公開用

この例では、C:¥xpa というディレクトリ内にこれらの４つを作成しています。以下の例ではこれらのディレクトリを使用しています。

スクリプト用ディレクトリ

例では、Scripts です。このディレクトリ内に以下のファイルをコピーします（「xxx」は、Magic xpa のバージョンによって異なります)。

- mgrqgnrc.dll
- mgrqhttp.dll
- mgrqcgi.exe
- MGREQ.INI

注： ファイル名は、Magic xpa のバージョンによって異なる場合があります。

RIA モジュール用ディレクトリ

例では、RIAModules です。このディレクトリ内に Magic xpa の Windows 用 RIA モジュール用ファイルをコピーします。.

- Mgcontrols.dll
- MgGui.dll
- MgNative.dll
- Mgutils.dll
- MgxpaRIA.exe
- MgxpaRIA.exe.config
- MgxpaRIA.exe.manifest

公開アプリケーション用ディレクトリ

例では、PublishedApplications です。このディレクトリ内に RIA として公開する html ファイルやマニフェストファイル等をコピーします。.

キャッシュ用ディレクトリ

例では、RIACache です。このディレクトリは、リッチクライアントの実行時にキャッシュファイルを作成するためのディレクトリです。

Apache のコンフィギュレーションファイルの変更

Apache のコンフィギュレーションファイルは httpd.conf という名前のテキストファイルで、Apache のインストール先の¥conf サブディレクトリにコピーされます。

ここに、前述で作成した２つ（スクリプト用、公開アプリケーション用）のディレクトリに対するエイリアスを定義する必要があります。以下の内容をコピーしディレクトリ名だけを実際のインストール環境に合わせて変更することで利用できます。

```
ScriptAlias /Magic3xScripts/ "C:/Mgxpa/Scripts/"
<Directory "C:/Mgxpa/Scripts">
Options None
AllowOverride None
Order allow,deny
Allow from all
Alias /Magic3xRIAApplications/ "C:/Mgxpa/PublishedApplications/"
<Directory "C:/Mgxpa/PublishedApplications">
Options None
AllowOverride None
Order allow,deny
Allow from all
</Directory>
```

Magic.ini の変更

Magic.ini を以下のように変更します。

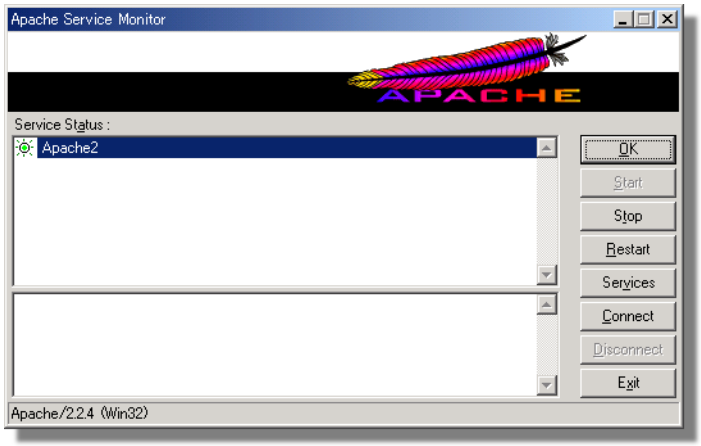
```
InternetDispatcherPath = /Mgxpa3xScripts/mgrqcgi.exe
WebDocumentPath = C:¥Mgxpa¥RIAModules
WebDocumentAlias = /Magic3xRIAApplications
CTLCacheFilesPath = C:¥Mgxpa¥RIACache
```

4.MRB が実行していることを確認する

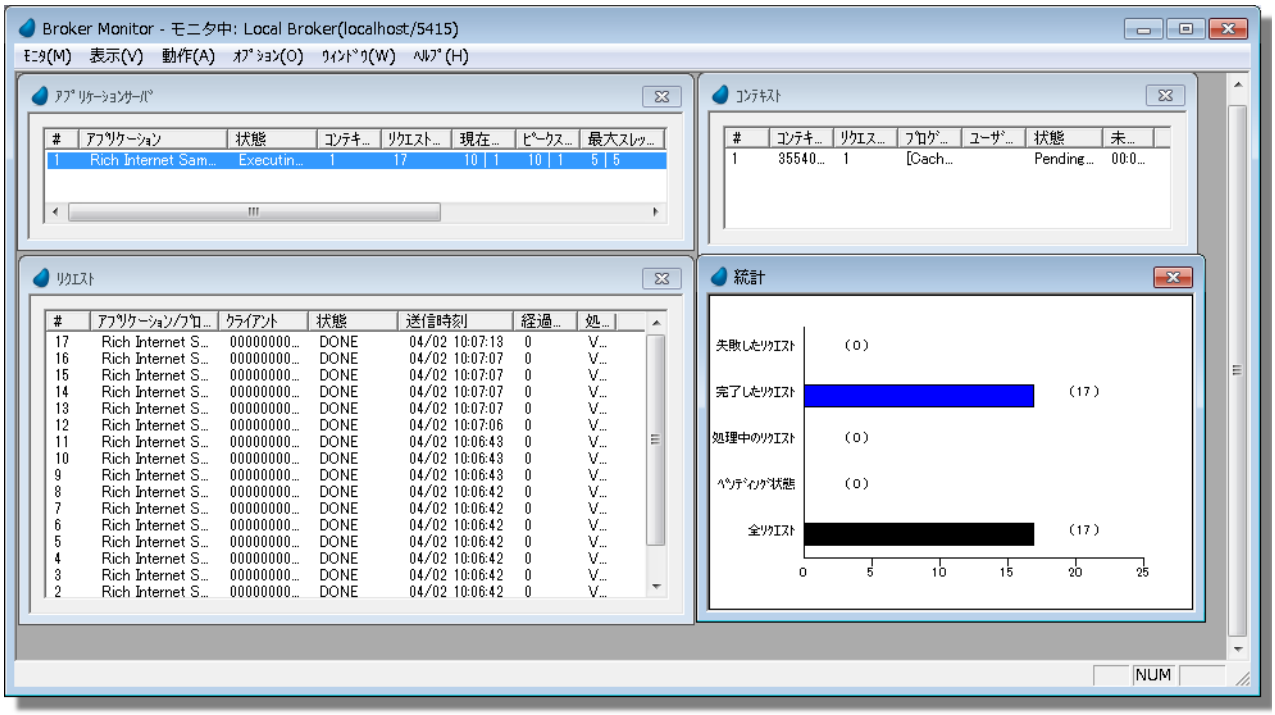
Magic xpa のインストール方法によって、MRB はサービスまたは実行形式のどちらかで実行されます。Windows のサービスとして実行する場合、OS の起動時に自動的に開始されます。実行形式でインストールされた場合は、手動で起動させる必要があります。MRB の起動／停止は、ブローカとリクエスタメニュー（**スタートメニュー→ Magic xpa Enterprise Studio**）上にあります。

ブローカモニタを使用することで MRB の実行状態を確認することができます。モニタに関する詳細は、「MRB の動作を監視するには」（775 ページ）を参照してください。

Apache サーバが実行していることを確認するには、Apache サービスモニタを使用するか、Windows のタスクバーにアイコンが表示されていることを確認します。

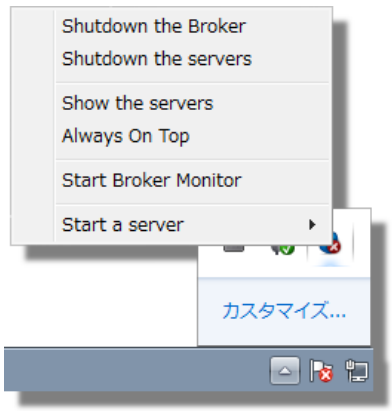


MRB の動作を監視するには



MRB を使用する場合、起動された MRB の統計情報を取得する必要があるかもしれません。Magic xpa には、MRB の実行状況を監視するためのいくつかのツールがあります。

上図のような Broker モニタは、MRB の操作状況をビジュアルでかつリアルタイムに表示することができます。



Windows のタスクバーの MRB アイコンでコンテキストメニューを開き、**Broker Monitor** をクリックすることで Broker モニタが起動されます。

Magic xpa のリクエスト関数

Broker モニタに表示される情報は、Magic xpa に組み込まれているリクエスト関数を使用することでも取得できます。

例えば、**RqLoad** 関数は、リクエストの総数や、各状態毎のリクエスト数（ペンディング中、処理中、処理が成功、処理が失敗）を返します。

さらにこれらの関数の中には、指定した MRB の処理を制御することができるものもあります。例えば、**RqRtBlock** 関数は、特定のサーバまたはサービスにリクエストが送られることを防止します。

これらの関数の多くは、Mgrb.ini ファイルに設定されたパスワード（**Supervisor Password** や **Query Password**）を指定する必要があります（「MRB のパスワードを定義するには」（779 ページ）を参照してください）。

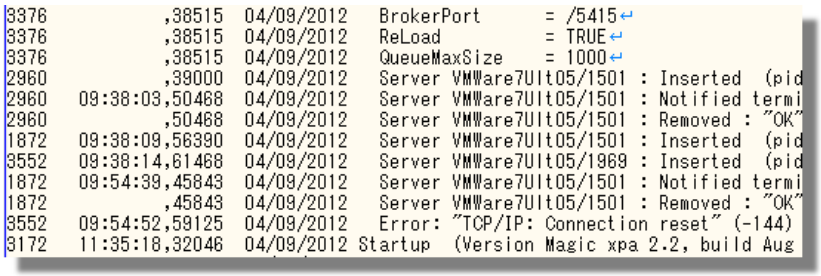
コマンドラインの情報

mgreqcmdl.exe を使用することで、Magic xpa プログラム外から MRB の情報を取得することができます。例えば、以下のように実行した場合、現在のアプリケーションサーバでサポートされている全てのアプリケーションが表示されます。

```
mgrqcmdl -query=app
```

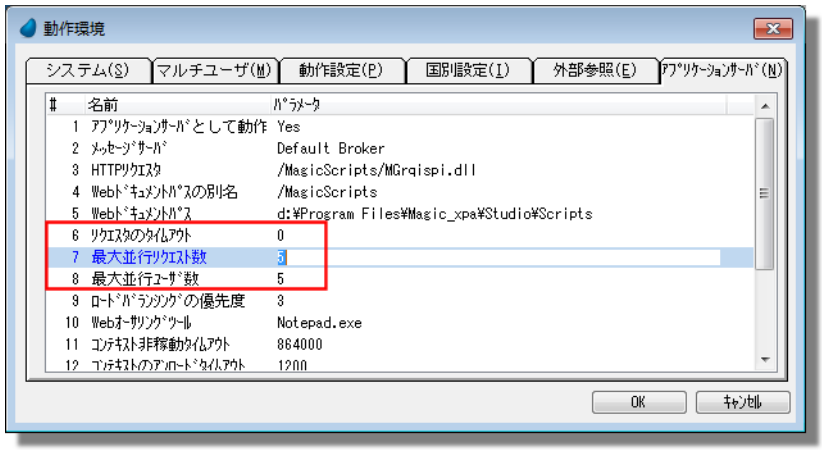
コマンドラインリクエストのパラメータ情報を確認する場合は、パラメータ無しでコマンドを実行してください。

MRB のメインログ



BrokerActivity.log から情報を取得することもできます。このログファイルは、デフォルトで Magic xpa のインストールディレクトリに作成されます。

Magic xpa が同時に処理するリクエスト数を制限するには



Magic xpa が使用するライセンスの内容にもとづいて、アプリケーションサーバが同時に処理することができるスレッド数が決定されます。しかし、**最大並行リクエスト数**（**オプション→設定→動作環境→アプリケーションサーバ**）の設定によってスレッド数の上限を制限することができます。エンジンが使用するスレッド数は、ライセンス数の上限内までの値を指定することができます。

0 が指定された場合、スレッド数はライセンス上の上限まで使用できます。

MRB が自動的にアプリケーションを起動するようにするには

```
[APPLICATIONS_LIST]
Online = MgxpaStudio.exe /DeploymentMode=T,C:¥Program Files¥Mgxpa¥Studio,,,0,0
MyApp1 = MgxpaRuntime.exe /DeploymentMode=R /StartApplication=C:¥Mgxpa¥Projects¥Masterring
xpa¥Mastering.ecf,C:¥Program Files¥Mgxpa¥Studio,,,0,0
```

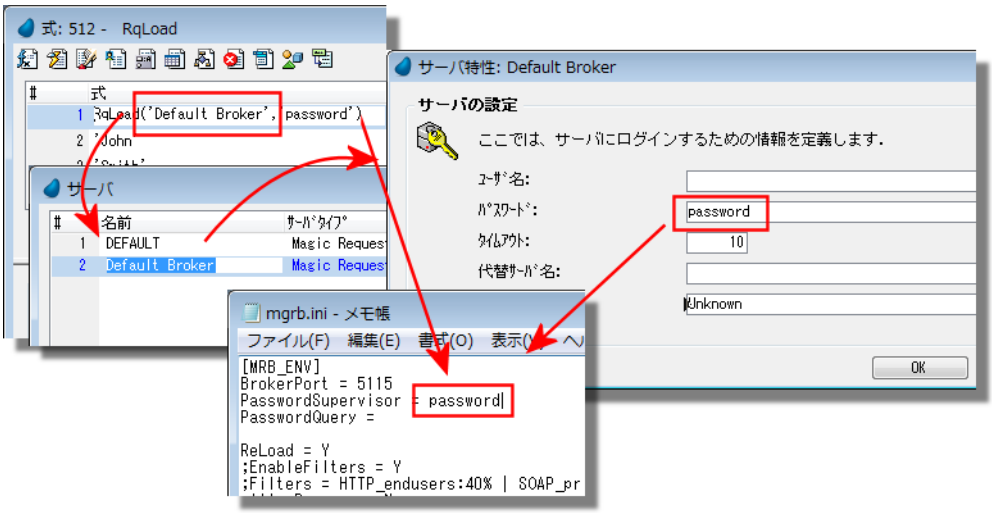
アプリケーションを自動的に起動するようにしたい場合は、mgrb.ini の [APPLICATIONS_LIST] セクションで指定します。上記の例は、**Magic xpa Enterprise Studio** と **Magic xpa Client** を起動するように指定したものです。

構文は以下の通りです：

エントリ名 =< コマンド >[< 作業フォルダ >], [< ユーザ名 >], [< パスワード >], [<MRB の初期化時に実行される回数 >], [< 起動されるエンジンの最大数 >]

<MRB の初期化時に実行される回数 > に 1 が設定された場合、指定されたアプリケーションを 1 インスタンスだけ起動するようになります。

MRB のパスワードを定義するには



MRB のパスワードは、mgrb.ini で定義します。ここには、２つのパスワード（**Supervisor Password** と **Query Password**）が設定されています。**Supervisor Password** は MRB を管理する場合、**Query password** は MRB に問い合わせをする場合に必要です。

Supervisor Password と同じパスワードを**サーバ特性**（**オプション→設定→サーバ→特性**）に指定する必要があります。

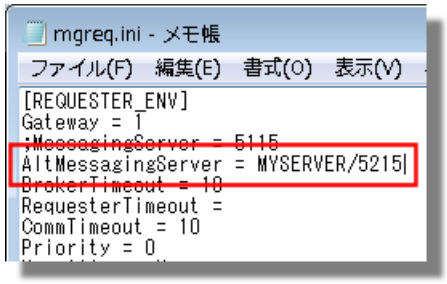
Supervisor Password は、Magic xpa のインストール時に設定されます。カスタムインストールを行うことでパスワードを任意に指定することができます。

Query Password はインストール処理では設定されません。これは、エンジンや実行アプリケーションの状況を確認するためだけに限定して使用されるためです。

代替えの MRB を定義するには

Magic xpa のリクエスタ機構は、メインで使用される MRB に加え、代替えの MRB を使用することができます。これによりリクエスタは、メインの MRB が動作していない場合、代替えの MRB を使用して処理を行うことができます。

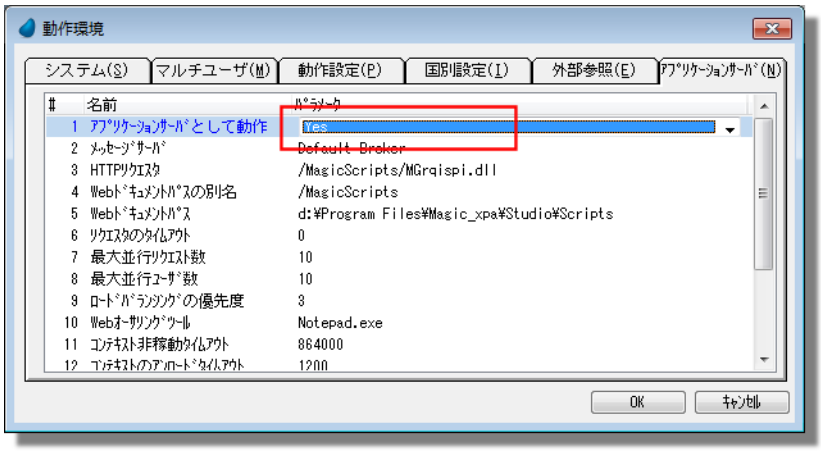
Mgreq.ini で MRB を定義する



リクエスタの動作環境は、**mgreq.ini** という名前のファイルで定義します。

代替えの MRB を追加するには、AltMessagingServer に使用する MRB のホスト名 / ポート番号を指定するだけです。

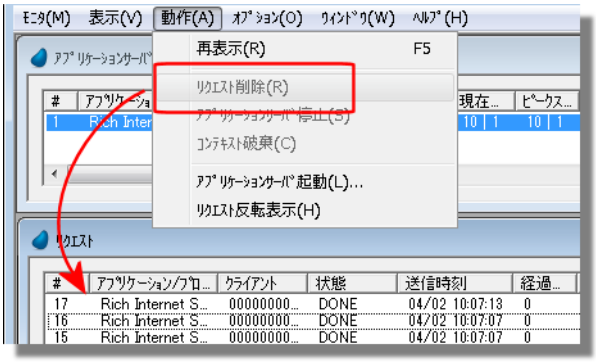
実行エンジンが MRB のリクエストを受けないようするするには



Magic xpa エンジンがリクエストを受け付けないようにするには、**アプリケーションサーバとして動作**（オプション→設定→動作環境→アプリケーションサーバ）を *No* に設定します。

ただし、一時的にリクエストを受けないようにする場合は、**RqRtBlock** 関数を使用します。再びリクエストを受信するようにする場合は、*RqRtResume* 関数を使用します。

キュー内で待ち状態のリクエストを削除するには



リクエストがまだリクエストキュー内でペンディング状態の場合、Broker モニタからリクエストを削除することができます。

- リクエストパネルでリクエストを選択します。
- **動作→リクエスト削除**を選択します。

これで、リクエストは削除されます。

ロードバランシングを実装するには

簡単なロードバランシング構成を実現するには、少なくとも２つのアプリケーションサーバが同じアプリケーションを実行し、同じ MRB に接続する環境が必要です。MRB は自動的に Magic サーバ間でリクエスト処理を平準化するように動作します。

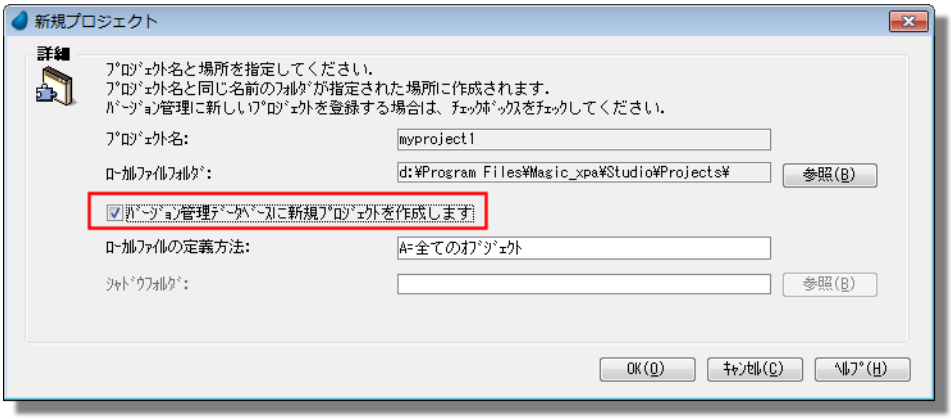
複数の異なる PC 上で複数の異なる Magic サーバを実行させることで、より複雑なロードバランシングを実現することができます。さらに、**ロードバランシングの優先度**（**オプション→設定→動作環境→アプリケーションサーバ**）を設定することで、各サーバに対し異なる優先順位を指定することができます。

［このページは意図的に空白にしています。］

第 37 章 : ソース管理

注 : ここでは、Team Foundation Server の使用を前提に説明しています。

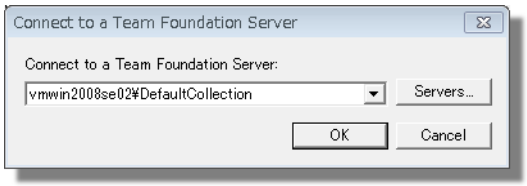
バージョン管理ソフトによって管理されるプロジェクトを作成するには



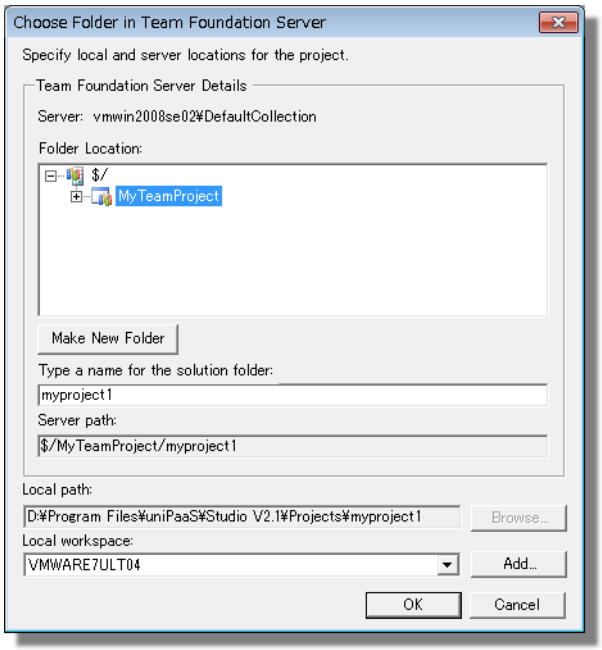
Magic xpa で新しいプロジェクトを作成する際に、最初からバージョン管理で管理するように指定することができます。ここではその方法について説明します。

必要条件 : あらかじめバージョン管理データベースを設定しておく必要があります。詳細は、「バージョン管理プロバイダを決めるには」（797 ページ）を参照してください。

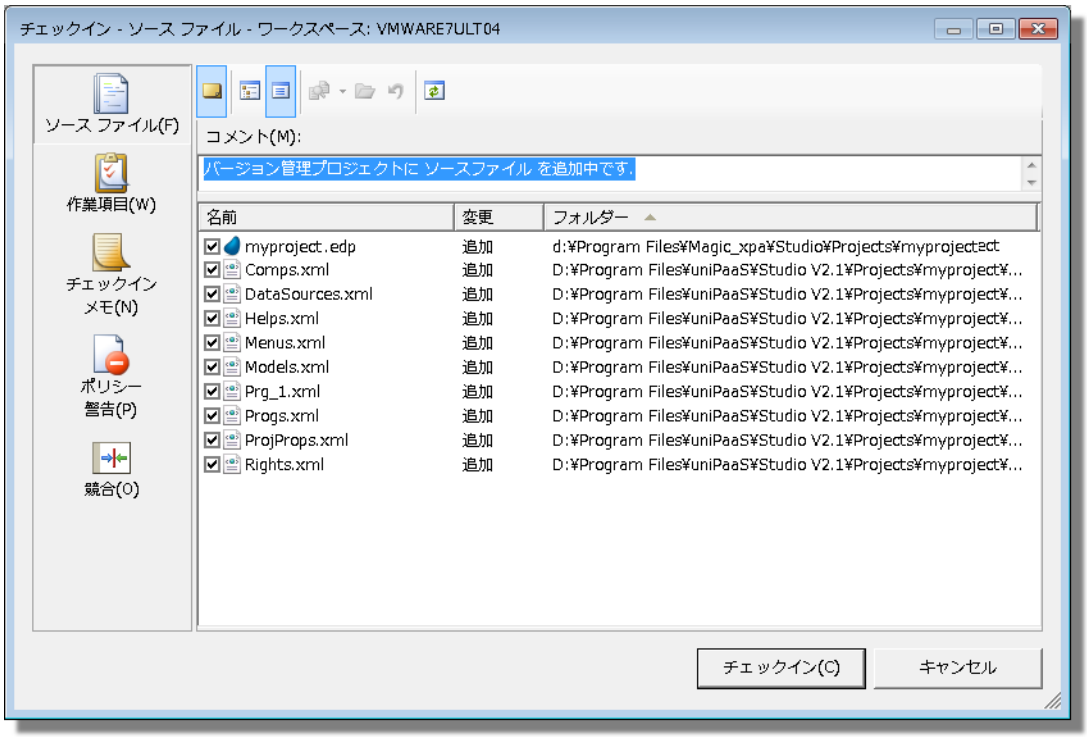
1. **ファイル→新規作成**を選択するか、**ウェルカムスクリーン**上の**新規作成**ボタンを**クリック**することで、手動で新規プロジェクトを作成します。**新規プロジェクト**ダイアログが表示されます。
2. 通常と同じように**プロジェクト名**と**位置**を入力します。そして、**バージョン管理データベースに新規プロジェクトを作成します**チェックボックスをチェックします。
3. **ローカルファイルの定義方法**は***全てのオブジェクト***のままにします。
4. OK をクリックすると、バージョン管理プロジェクトの作成ダイアログが表示されます。ここには、コメントを入力することができます。
5. OK をクリックすると、Team Foudation Server への接続ダイアログが表示されます。ここで接続するサーバを選択し、OK をクリックします。Team Foudation Server へのログインダイアログが表示されるので、ユーザ ID とパスワードを入力して OK をクリックします。



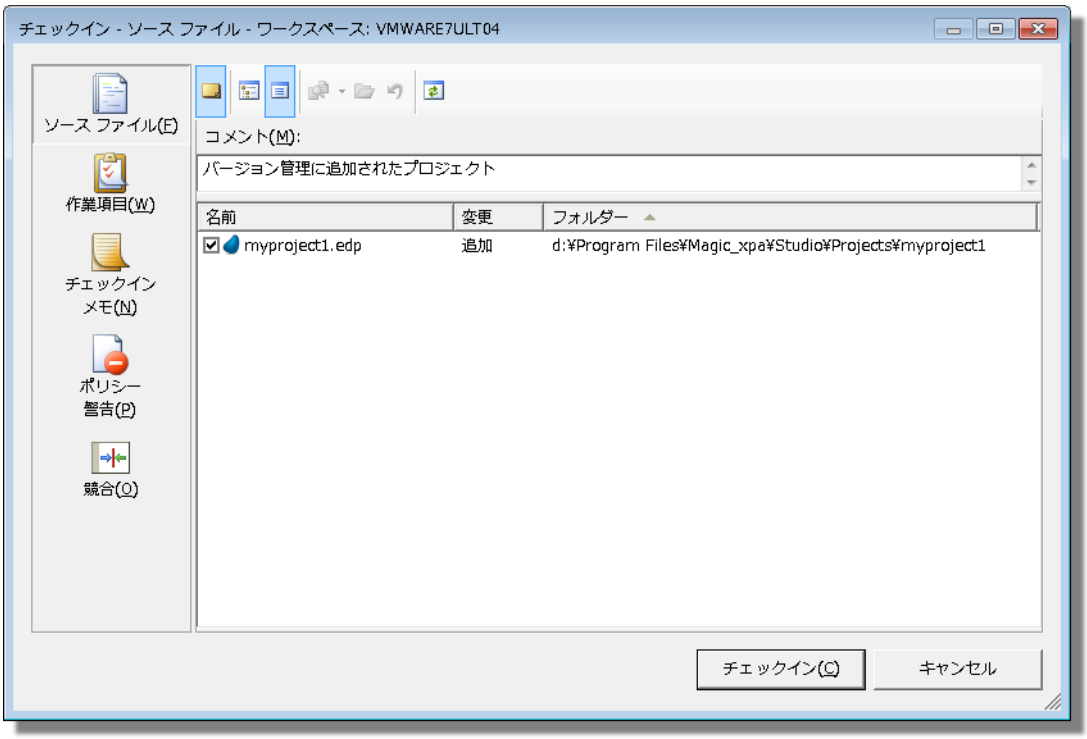
6. フォルダの選択ダイアログが表示されます。Team Explore で事前に定義されたフォルダ（例では、”Sample”）を選択し、OK をクリックします。



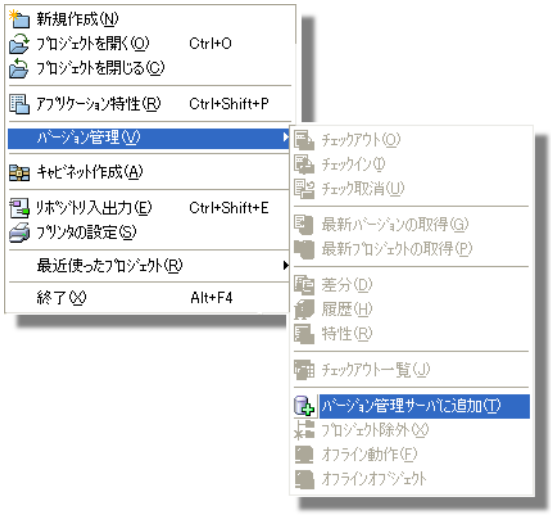
7. Team Foudation Server のチェックインダイアログが表示されます。チェックインをクリックすることで表示されているオブジェクトがまとめてチェックインされます。



8. 次にプロジェクトファイルに対するチェックインダイアログが表示されます。チェックインをクリックすることでプロジェクトがチェックインされます。



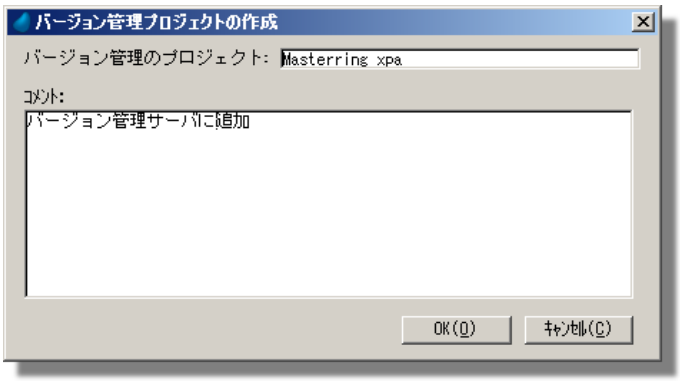
既存のプロジェクトをバージョン管理データベースに追加するには



すでに開発中のプロジェクトが存在している場合、必要であれば VC サーバに登録することができます。

必要条件： あらかじめバージョン管理データベースを設定しておく必要があります。詳細は、「バージョン管理プロバイダを決定するには」（795 ページ）を参照してください。

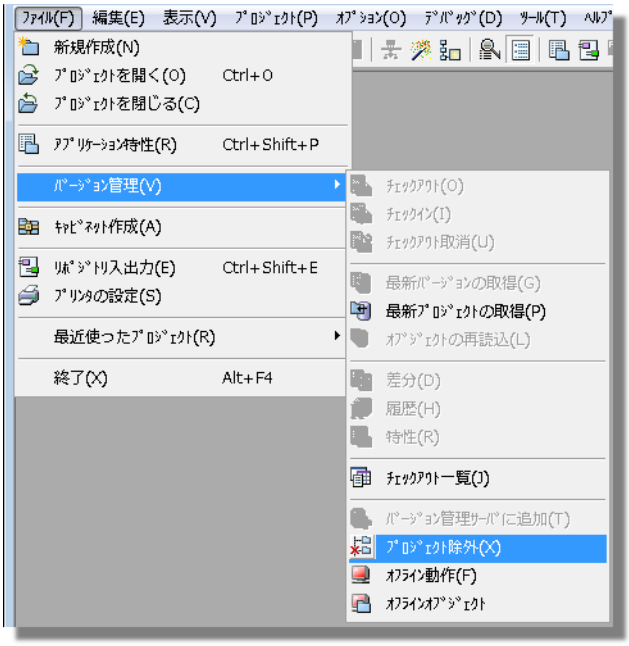
1. バージョン管理を行いたいプロジェクトを開きます。
2. **バージョン管理プロジェクトの作成**ダイアログが表示されます。上記のように、使用しているバージョン管理製品に応じて表示されるウィンドウが異なります。必要なデータを入力し OK をクリックします。



Team Foundation Server を使用している場合は、このようなウィンドウが表示されます。プロジェクトを作成する時にコメントを追加することができます。

OK をクリックした後、第 37 章：「バージョン管理ソフトによって管理されるプロジェクトを作成するには」（785 ページ）と同じように操作します。

プロジェクトをバージョン管理から削除するには



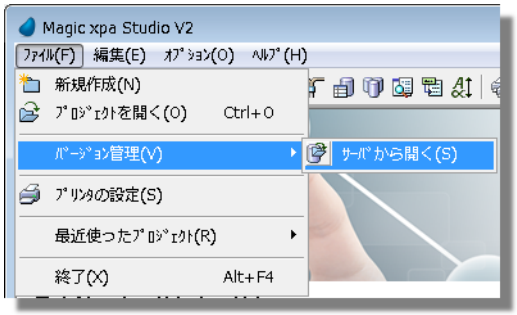
プロジェクトを VC サーバで管理しなくなった場合、**ファイル→バージョン管理→プロジェクト除外**を選択することで、VC サーバの管理から除外することができます。

プロジェクトを除外しても、プロジェクトファイルのコピーはまだ VC サーバ上に残っています。再びバージョン管理を使用し始める場合は、VC サーバ上のコピーかクライアント側のコピーのどちらかを削除する必要があります。

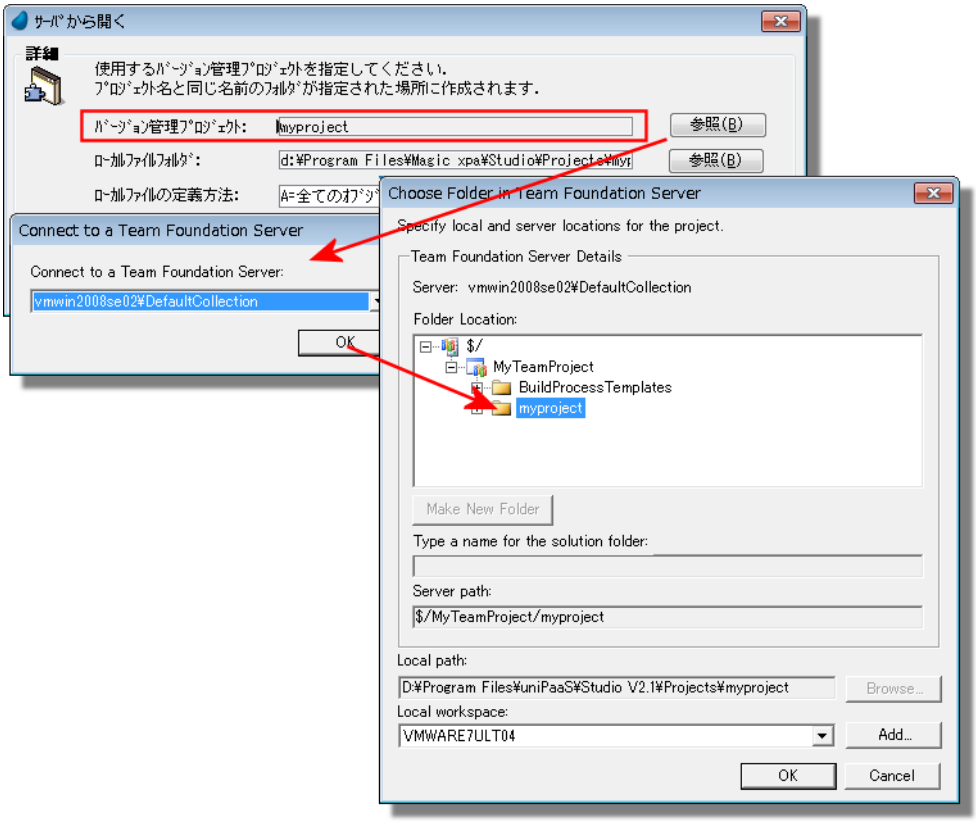
バージョン管理プロジェクトを新たにオープンするには

バージョン管理されたプロジェクトを開発者が初めてオープンする場合、一旦、開発者は作業用としてプロジェクトのローカルなコピーを作成します。

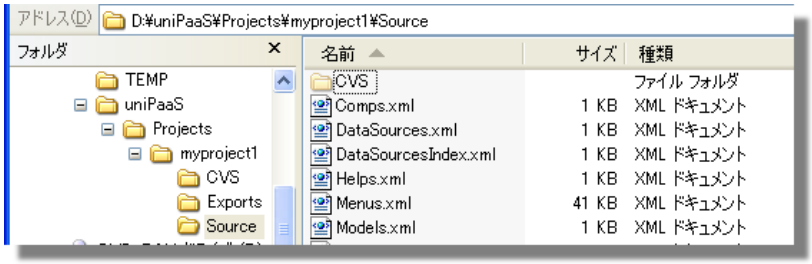
1. 現在のプロジェクトを閉じます。



2. ファイル→バージョン管理→サーバから開くを選択します。



3. サーバからオープンするダイアログが表示されます。
4. 参照をクリックし、オープンするプロジェクトを選択します。この例では myproject を選択しています。
5. プロジェクトのコピーは**位置**で指定されたパスに作成されます。

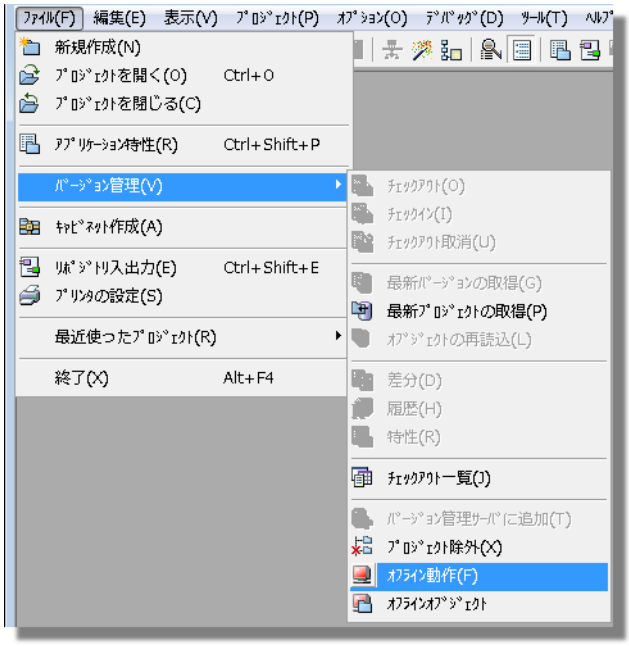


注： 1台の PC 上で複数のユーザ ID を使用してプロジェクトをオープンするような場合は、一覧に指定するコピー先のパスをユーザ毎にユニークなパスにする必要があります。

バージョン管理サーバが無効な場合に、バージョン管理プロジェクトを開発するには

バージョン管理を使用している場合、VC データベースが何らかの理由でオフラインになる可能性があります。このような場合に開発を止めないようにしたいのであれば、オフラインの状態で VC 環境のまま開発を継続させることができます。

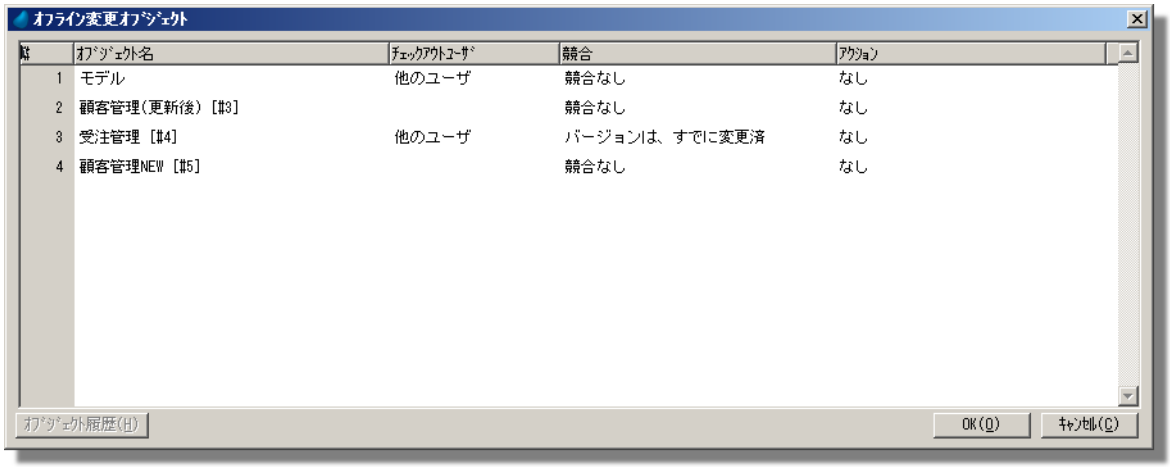
オフライン状態で開発する場合もオブジェクトを変更することができます。Magic xpa は変更されたオブジェクトを追跡し、VC サーバに再接続したときこれらの変更を反映するようにします。



プロジェクトは、一度オフラインモードになると VC プロジェクトで管理されていないかのように動作します。チェックアウトの操作を行うことなくリポジトリやリポジトリ内のオブジェクトを修正することができます。

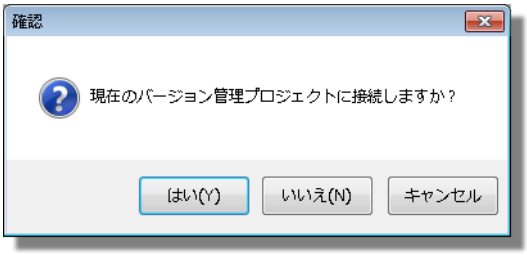
オフラインの場合に変更される項目

ファイル→バージョン管理→オフラインオブジェクトを選択することで、オフラインで作業するように選択してから変更された全てのオブジェクト一覧が表示されます。

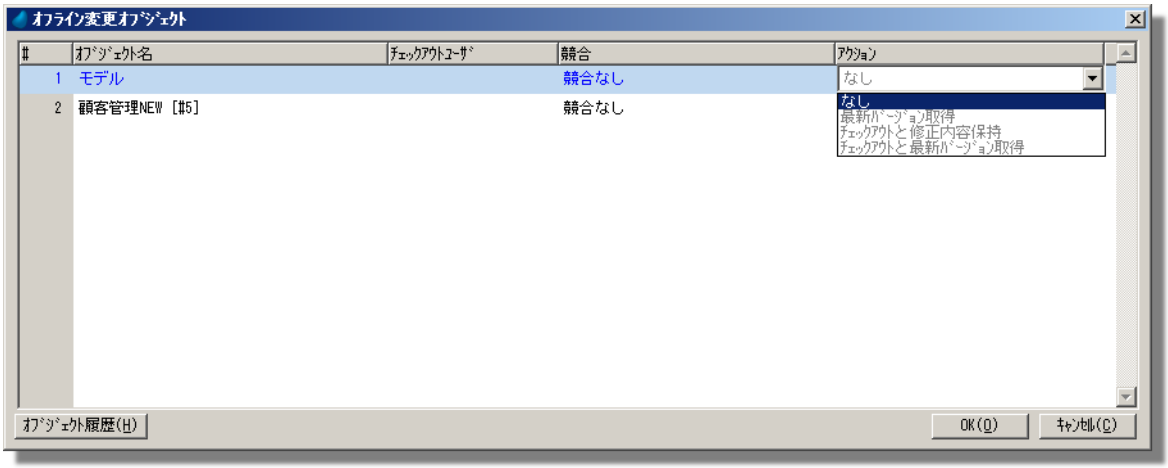


このモードでは、変更されたオブジェクトを見るために VC サーバをオンラインにする必要がありません。しかし、取得できる情報がないためオブジェクト名だけが表示されます。

VC への再接続



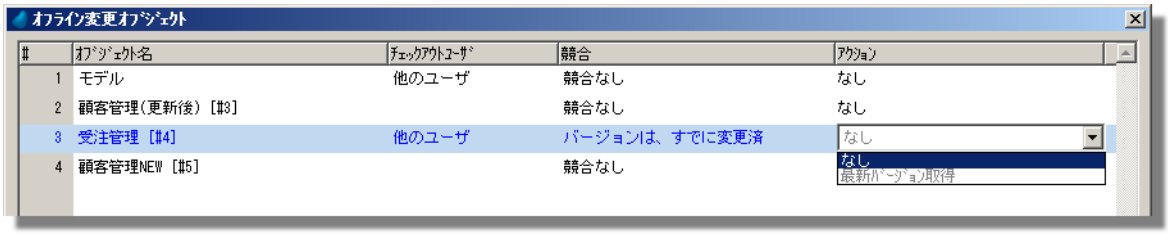
オフラインモードで作業している場合、新しい **Magic xpa Studio** のセッションを開始するたびに、VC サーバに接続するかどうかを確認するダイアログが表示されます。



その際はい（Y）を**クリック**すると、変更されたオブジェクトの一覧が表示されます。ここで変更された各オブジェクトに対してどのような対応を行うかを指定することができます。

1. **オブジェクト名**カラムでは、Magic xpa 上の名前が表示されます。この名前には、バーレン（丸括弧）内にプログラム番号が表示されます。
2. **チェックアウトユーザ**カラムには、チェックアウトしたユーザ ID が表示されます。
3. **競合**カラムには、競合しているかどうかが表示されます。２人の異なるユーザが同じオブジェクトに異なる変更を加えた場合、競合していることになります。
4. **アクション**カラムでは、競合している場合の対応方法を選択することができます。

すでに他のユーザによってチェックアウトされているオブジェクトがオフラインモードで修正されている場合、以下の２つのオプションのみ表示されます。



- なし …… 何もしません（変更内容を有効にします）。
- 最新バージョン取得 …… 最新のバージョンによって変更内容を上書きします。

更新内容を追跡するには

開発者がプログラムをチェックアウトしたりチェックインするたびに、VC ツールは変更内容を追跡します。開発者は、変更理由について簡単なコメントを書き込むことができます（これは必ず行ってください）。

変更内容を確認したい場合は、ファイル→バージョン管理→履歴を選択することで参照できます。どのような内容が表示されるかは、使用している VC ソフトに依存します。

モデルとデータソースをチェックアウトする

モデルとデータソースは個別にチェックアウトすることができません。これらのリポジトリ内でファイル→バージョン管理→チェックアウトを選択すると、リポジトリ全体がチェックアウトされます。

プログラムのチェックイン / チェックアウトを行う

リポジトリ内のオブジェクトを使用して開発する場合、ファイル→バージョン管理→チェックアウトを選択します。その際、コメントを入力するダイアログが表示され、その後に編集可能な状態で開きます。

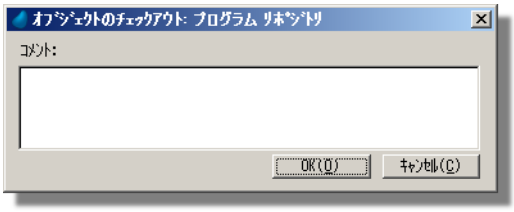
修正が終了したら、オブジェクト内でファイル→バージョン管理→チェックインを選択します。その際もコメントを入力するダイアログが表示されます。

これらのコメントは、オブジェクトの変更履歴として表示されます。

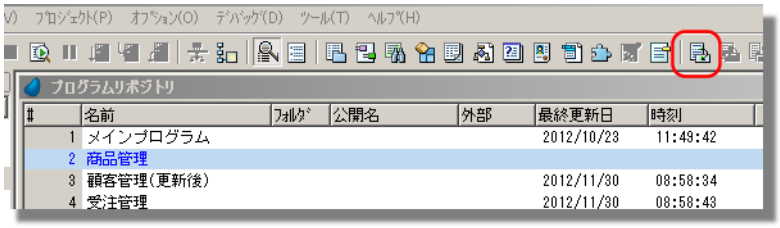
自動的にプログラムリポジトリをチェックアウトする

オブジェクトの追加や削除などの変更を行う場合、リポジトリ全体をチェックアウトする必要があります。

開発者がこのような処理を行おうとした場合、Magic xpa は自動的にリポジトリをチェックアウトしようとします。その際、コメントを入力するダイアログが表示され、リポジトリはチェックアウトされます。



手動でプログラムリポジトリをチェックアウトする



手動でプログラムリポジトリをチェックアウトするには以下のようにします。

1. ヘッダ行にカーソルを置きます。
2. リポジトリのチェックアウトアイコンをクリックします。

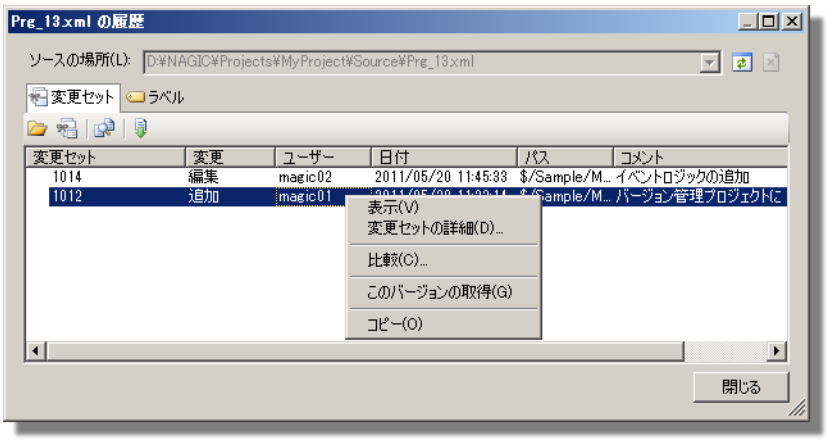
ヘッダ行（カラム名が表示されている行）に位置付けた状態でチェックアウトアイコンをクリックすることで、手動でリポジトリのチェックインやチェックアウトを行うことができます。

２つのファイルの比較結果が並べた状態で表示されます。変更、挿入、削除の行が色分けされて表示されます。

注： ソースファイルは、UTF-8 でエンコードされているため、日本語はエンコードされた状態で表示されます。

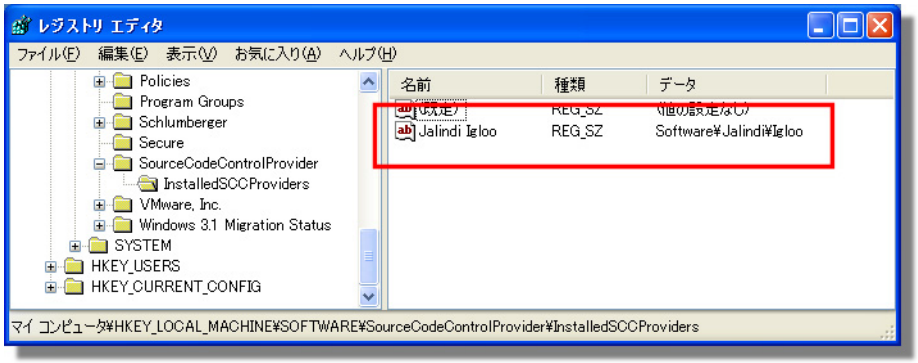
履歴を表示する

ファイル→バージョン管理→履歴を選択することで、パークしているオブジェクトの変更履歴が表示されます。カーソルがヘッダ行のある場合は、リポジトリのチェックイン／チェックアウト履歴が表示されます。



コンテキストメニューから比較をクリックすると差分ツールが起動されます。

バージョン管理プロバイダを決定するには



現在使用されているバージョン管理プロバイダは、OS のレジストリに登録されています。登録内容は以下のようにして確認できます。

1. スタートメニュー→ファイル名を指定して実行を選択します。
2. **Regedit** を入力し、OK をクリックします。レジストリエディタが起動されます。
3. 次のパスを選択します。

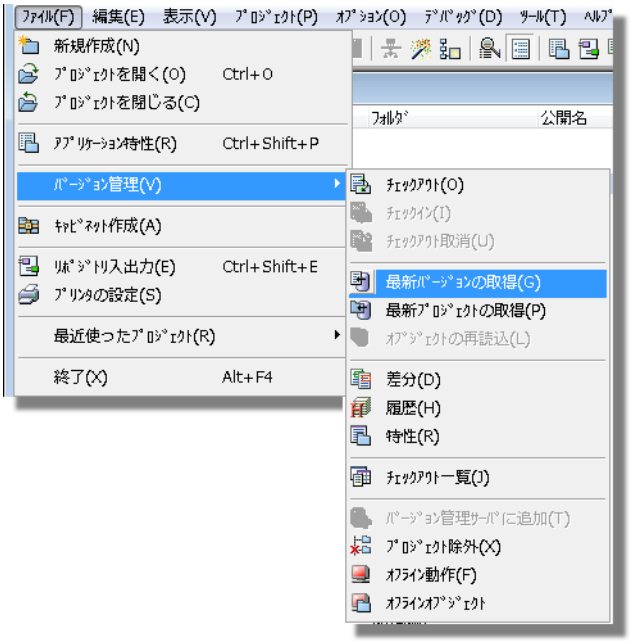
   HKEY_LOCAL_MACHINE¥SOFTWARE¥SourceCodeControlProvider¥ProviderRegKey

ここに Magic xpa が使用するソースコード管理プロバイダが定義されています。他のバージョ管理システムがインストールされている場合もあります。しかし、使用できるものはここに定義されているものだけです。

以下のキー内にインストールされているすべてのプロバイダのリストが定義されています。

HKEY_LOCAL_MACHINE¥SOFTWARE¥SourceCodeControlProvider¥InstalledSCCProviders

オブジェクトを前の状態にロールバックするには



この説明におけるロールバックとは、オブジェクトまたはプロジェクト全体を前のバージョンに戻すことを意味しています。

1 つのオブジェクトの最新バージョンを取得する

1 つのオブジェクトの最新バージョンを取得するには以下のようにします。

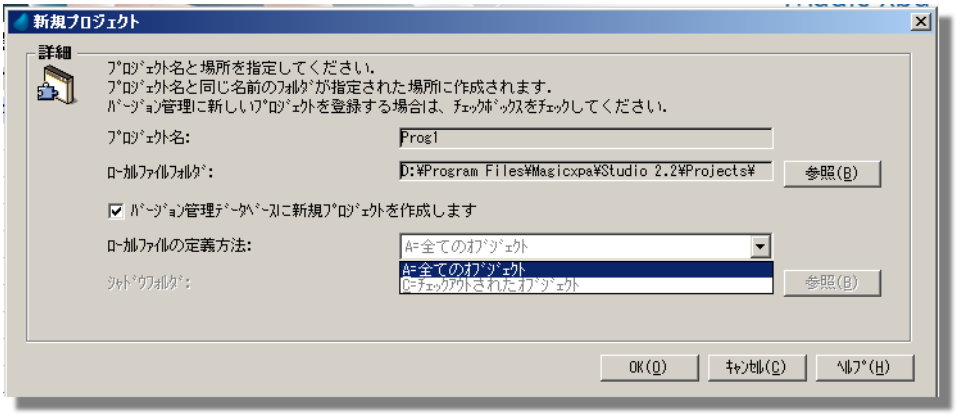
1. ロールバックしたいオブジェクト上にカーソルを置き、**ファイル→バージョン管理→最新バージョンの取得**を選択します。
2. 確認ダイアログが表示されるので、OK を押下します。
3. これでそのオブジェクトの以前にチェックインされたバージョンが取得されます。

プロジェクト全体をロールバックする

プロジェクト全体をロールバックするには、以下のようにします。

1. **ファイル→バージョン管理→最新プロジェクトの取得**を選択します。
2. 差し変わる各オブジェクトに対する確認ダイアログが表示されます。
3. 処理が終了すると、差し変えられたすべてのオブジェクトのリストが表示されます。

ソースファイルをローカルにコピーしないで、バージョン管理を行うには



バージョン管理ツールを使用している場合、2 つのどちらかもモードで管理することができます。

- ***A= 全てのオブジェクト***……ローカル PC 内にすべてのオブジェクトを保存します。これは次のことを意味しています。
  - ローカル PC 上にすべてのプログラム、モデル、データソースなどの各リポジトリのソースファイルをコピーをします。
  - これらをリフレッシュするまで、オブジェクトは変更されません。
  - プログラムのテストを行ったり、問題が発生した場合、ローカル側での変更によるものであることが確認できます。
- ***C= チェックアウトされたオブジェクト***……現在作業しているオブジェクトのみをローカル PC 内に保存します。これは次のことを意味しています。
  - 他のオブジェクトは、（バージョン管理ツールによって定義された）シャドウフォルダ内に格納されています。
  - ここには、他の開発者によって行われた変更内容が反映されています。

言い換えると、**チェックアウトされたオブジェクト**を使用して、簡易「統合テスト」を行うことができます。これを行うかどうかは開発者次第です。通常は、***全てのオブジェクト***を使用することを推薦します。この場合、他のユーザがオブジェクトを変更しても、その変更の影響を受けません。

しかし、他の開発者の変更内容を参照したい場合、バージョン管理システムがシャドウフォルダをサポートするのであれば、以下のように設定することで可能になります。

1. **ローカルファイルの定義方法**で***チェックアウトされたオブジェクト***を選択します。
2. **VC 用シャドウフォルダ**を指定します。ここから、ローカル PC にコピーされます。

これでプログラムをテストするとき、Magic xpa はソースファイルのグローバルコピーを使用します。

［このページは意図的に空白にしています。］

第 38 章 : マルチ言語サポート

アプリケーションを別の言語用に変換するには

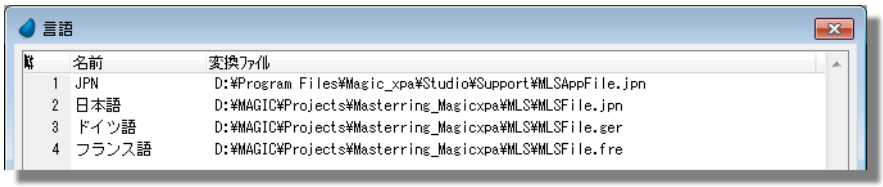
今日のグローバル経済において、1 つの言語用だけのアプリケーションを作成するのでは十分でありません。アプリケーションを構築する際、必ずしもエンドユーザがどのような言語を使用しているを分かっているとは限りません。Magic xpa には、複数の言語をサポートする機能が組み込まれています。このため、アプリケーションの作成後に、対応する言語をサポートすることができます。

これは、MLS（Magic Language Service）を使用することで実現できます。MLS は、ユーザに表示される内容を変換することができます。アプリケーションのソースファイルは、変更する必要がありません。この章では、MLS 機能をアプリケーションに反映させる方法について説明しています。

1. 変換ファイルの作成

アプリケーションを翻訳したい言語に対応した変換ファイルが必要となります。作成方法は、「変換ファイルを定義するには」（801 ページ）を参照してください。

2. 言語ファイルの設定



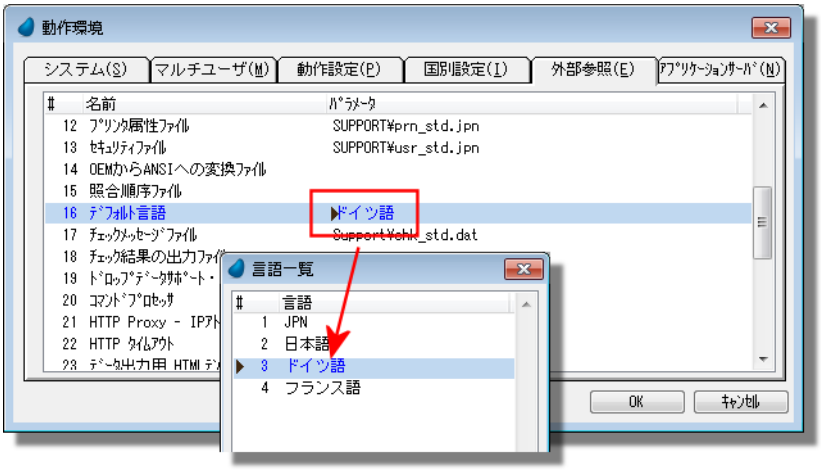
次に使用する言語ファイルを Magic xpa で定義する必要があります。

1. オプション→設定→言語に移動します。
2. 言語の名前（例えば、" ドイツ語 "）を入力します。これは、使用する言語を選択するときに、使用する名前です。
3. 変換ファイルの名前を入力します。これは、MLS_BLD ユーティリティによって作成されるファイルの名前です。

これによって Magic.ini ファイルの［MAGIC_LANGUAGE］セクションで、以下のように設定されます。

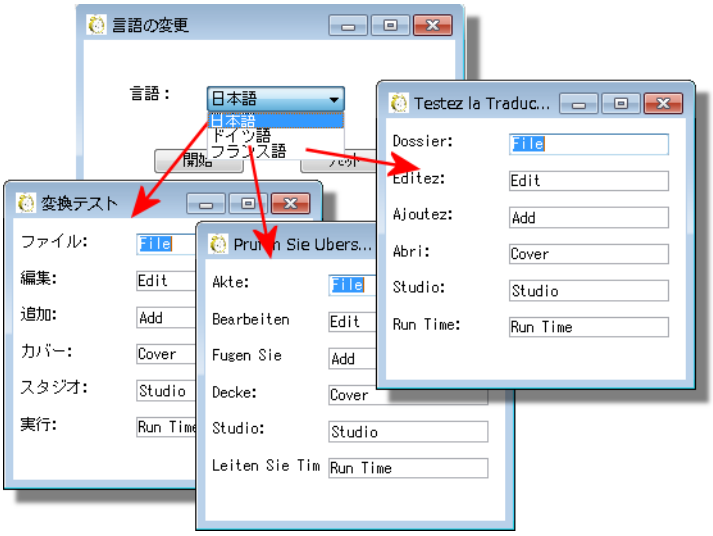
```
[MAGIC_LANGUAGE]
JPN = D:¥Program Files¥xpa¥Studio¥Support¥MLSAppFile.jpn
日本語 = D:¥MAGIC¥Projects¥Masterring_xpa¥MLS¥MLSFile.jpn
ドイツ語 = D:¥MAGIC¥Projects¥Masterring_xpa¥MLS¥MLSFile.ger
フランス語 = D:¥MAGIC¥Projects¥Masterring_xpa¥MLS¥MLSFile.fre
```

3. 実行時の言語を選択する



これで、プログラムは実行時に変換することができるようになります。これは、Magic.ini ファイルの **StartingLanguage** パラメータを指定することでも可能です。この例では、アプリケーションはドイツ表示で開始されます。

この他に、**SetLang()** 関数を使用することで実行時に言語を変更することもできます。



この例では、ユーザが言語を設定することで、異なる変換ファイルを使用して同じプログラムを呼び出しています。ユーザの選択内容に従って、プログラムのテキストは、日本語やドイツ語、フランス語で表示されます。

4. MLS でのその他の問題

指定した語句と一致したものが見つからない場合、その語句は翻訳されません。また、テキストは語句と変換ファイルで同一でなければなりません、同じでない場合、見つからないものとして扱われます。これは、"Studio" と "Studio：" は、変換ファイル内では別の項目でなければならないことを意味しています。

式は実行時に評価されるため、式の内容は異なる処理を行う必要があります。これについては、「式で使用している文字列を別の言語に翻訳するには」（803 ページ）を参照してください。

MLS によってすべてが変換されるわけではありません。日付フィールドでは、Magic xpa は異なる日付表示形式を内部でサポートしています。この設定は、**オプション→設定動作環境→国別設定**の**日付モード**で行います。金額のフィールドは、実行時に変更することができる書式を使用して処理することができます。

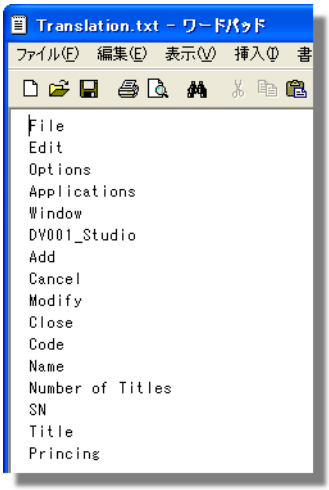
Magic xpa の実行時の表示は、日本語版の場合、日本語表示がデフォルトになります。言語に依存する大部分のテキストは、MGCONST.xxx ファイルで定義されています。.xxx は言語を示しています。

一旦これらの手順に従った場合、二つ以上の異なる言語環境で動作するアプリケーションを持つことになります。

変換ファイルを定義するには

A 変換ファイルには、アプリケーション内で使用されている語句とその対訳語のリストが含まれています。

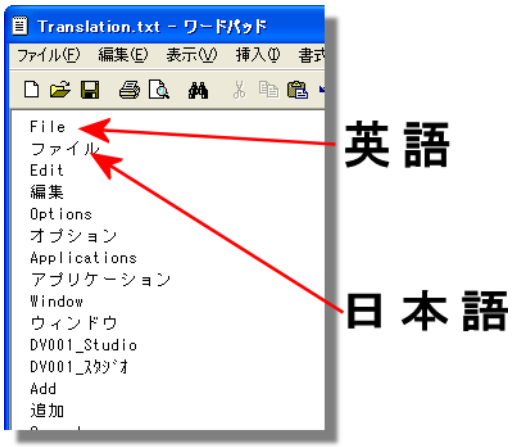
1. 変換リストを作成する



最初に、変換する文字列のリストを作成する必要があります。このファイルには、ネイティブ言語に含まれている文字列の全てを含めます。

変換する必要のある文字列を入力することで、手動でファイルを作成することができます。

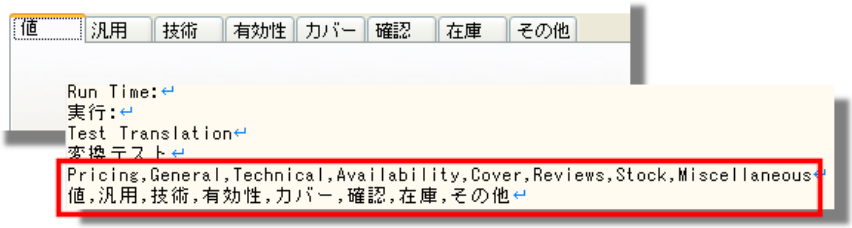
Magic xpa に添付されている文字列抽出ユーティリティを使用することもできます。このユーティリティは、アプリケーション内で使用されている文字列を抽出し、XML ファイルとして出力することができます。このユーティリティは、ソースコードも含まれているため、変換元や変換リストの出力処理をカスタマイズすることもできます。



次に、ネイティブ言語の語句の下に 1 行挿入して、対訳語句を入力します。この例では、英語の各文字列行の下に日本語の対訳行が挿入されています。

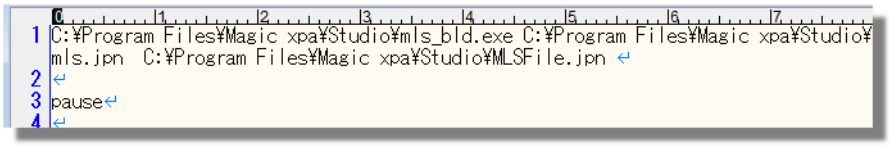
この手順を実行するには、両方の言語をよく知っている人のサポートが必要な場合があります。

また、テキストファイルで Unicode を使用するため、各言語で使用される特殊文字を入力することもできます。



チョイスコントロール用に変換文字列を作成する場合、文字列全てを翻訳対象とする必要があります。それぞれの文字列の項目数が一致していなければなりません。

2. 言語ファイルの作成



変換リストが準備できたら、Magic 用の言語ファイルを作成することができます。これは、**MLS_BLD** コマンドを実行することで実現できます。

MLS_BLD コマンドの構文は、以下の通りです：

MLS_BLD *< ソースファイル >< 出力ファイル >< コードページ >*

パラメータ：

- *ソースファイル*…… 変換ファイルのパスとファイル名
- *出力ファイル*…… （MLS_BLD で作成される）Magic の言語ファイルのパスとファイル名
- *コードページ*…… 変換ファイルのコードページ。変換ファイルが Unicode で作成されている場合は不要です。

コマンドプロンプトから実行したり、.bat ファイルを作成したり、Magic プログラムから実行させたりすることができます。

これで終了です。MLS_BLD コマンドを実行したら、第 38 章：「アプリケーションを別の言語用に変換するには」（799 ページ）での説明に従うことで、言語ファイルを使用することができます。

式で使用している文字列を別の言語に翻訳するには

MLS 機能は、実行時にフォーム上の文字列を変換します。しかし、プログラムの実行中に、式が評価され、その内容が翻訳される必要があります。**MlsTrans()** 関数を使用すること、このような処理を実現できます。

通常のテキスト文字列については、以下のように **MlsTrans()** 関数のパラメータとして文字列を指定するだけで変換されます。

```
MlsTrans('Hello world!')
```

これで、テキストは現在選択された言語で表示されます。

ヒント: *例えば、以下のような式を翻訳する場合：*

```
'User ' & trim(A) & ' logged off at ' & tstr(time()) & ' on ' dstr(date(), '##/##/####')
```

MlsTrans() 関数のパラメータとしてテキストの各々の部分を指定した場合、以下のように定義することになります。

```
MlsTrans('User') & Trim(A) &  MlsTrans('logged off at ') & Tstr(Time(),'HH:MM:DD') &
MlsTrans(' on ') &Dstr(Date(), 'YYYY/MM/DD')
```

このように指定した場合、変換先の言語の文法に合わなくなる場合があります。このような場合の解決策として、**StrBuild()** 関数を使用して、以下のように定義することができます。

```
StrBuild (MlsTrans('User @1@ logged off at @2@ on @3@'),A,Tstr(Time(),'HH:MM:DD'),
Dstr(Date(), 'YYYY/MM/DD'))
```

これによって、以下の文字列を変換することができます。

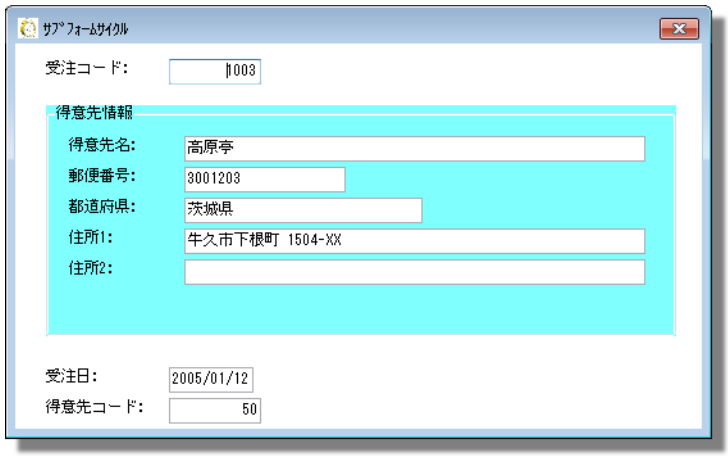
```
'User @1@ logged off at @2@ on @3@'
```

完全な文にするために、必要に応じて語順を変更してください。

［このページは意図的に空白にしています。］

第 39 章 : リッチクライアント補足

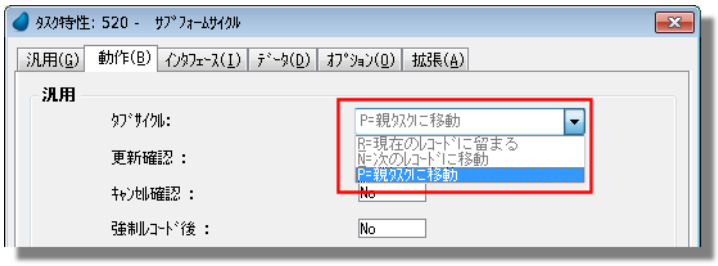
サブフォーム上の最後のコントロールから親タスクに移動するには



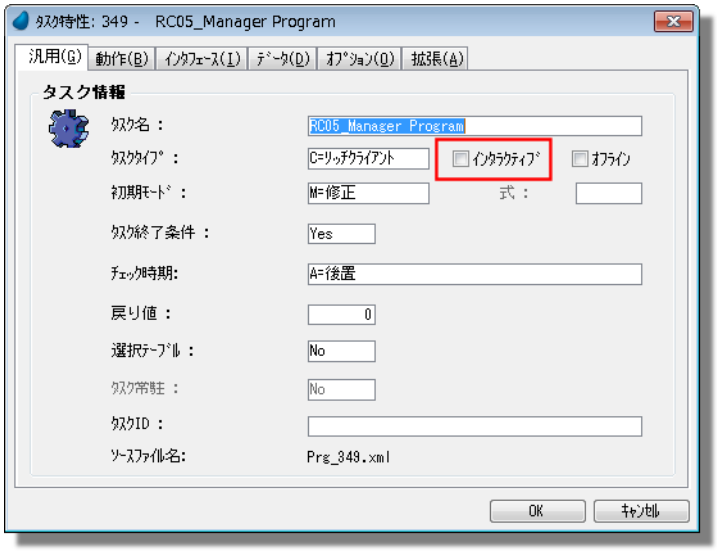
デフォルトでは、サブフォームは独立したウインドウのように表示されます。つまり、カーソルがサブフォームに移動したり、マウスでクリックすると、カーソルはサブフォーム内で循環し、ユーザが Esc キーを押下するかサブフォーム外をクリックするまでサブフォームから抜けません。

しかし、サブフォームが親タスクの一部であるかのように動作させたい場合があるかもしれません。これは、カーソルがサブフォーム内を移動し、フィールドからフィールドにタブ移動するようにサブフォームから抜けることができるようにすることです。オンラインタスクでこのようにするには、イベントを使用する必要があり、第 8 章 :「サブフォーム上のコントロールから親のコントロールに自動的に戻るには」（191 ページ）で説明されています。

しかし、リッチクライアントタスクでは、簡単に実現することができます。サブフォームタスクまたはプログラムの**タスク特性**で、**タブサイクル**特性を***親タスクに移動***に設定します。これだけで意図した動作になります。



リッチクライアントで管理プログラムを作成するには



プログラムの中には、ユーザの操作が全く必要がなく、ウィンドウを表示する必要のないものがあります。このようなプログラムは、バックグラウンドで実行され、必要に応じてユーザ用のウィンドウを表示させたりします。オンライン環境では、これらのプログラムはバッチプログラムになります。

しかし、リッチクライアント環境では、バッチプログラムはサーバでのみ実行されます。クライアントで実行するバックグラウンドタスクを作成するには、**タスク特性**の、**インタラクティブ**特性のチェックボックスのチェックを外してください。これにより、プログラムはユーザ操作を必要とせず、バックグラウンドで実行します。これらのタスクは、クライアントで実行されるため、インタラクティブなリッチクライアントタスクを呼び出すことができます。

ブラウザコントロールで実行している Web ページを操作するには

リッチクライアントのブラウザコントロールで Web ページを表示している時、その Web ページの情報を受け取ったり、Web ページに情報を送ることができます。

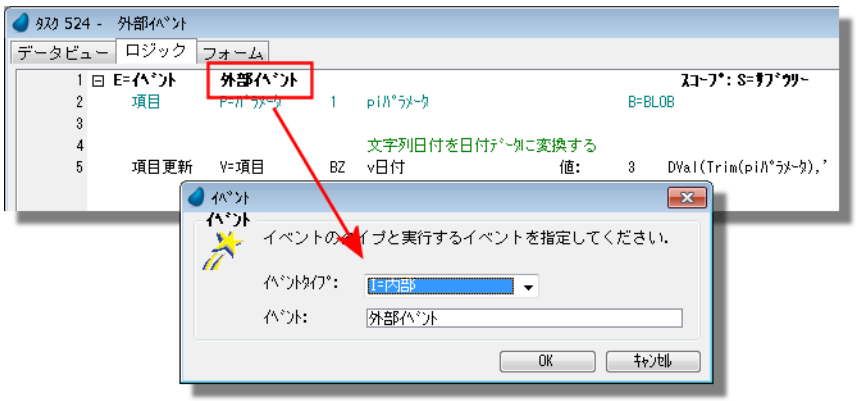
情報の送信

MGExternalEvent と呼ばれる特殊なイベントを使用することで、Web ページはリッチクライアントプログラムに情報を送信することができます。このイベントは、Web ページのコードに記述します。

たとえば、日付を引数として Magic xpa に送り返す VBScript は、以下のように記述します。

```
<SCRIPT LANGUAGE='VBScript'>

// This VBScript segment handles the click event of the Calendar object.
// Upon a click on the object the MGExternalEvent function is called with
//   the expected arguments.
// This allows external modules on the page to interact with the xpa
//   engine.

Sub Calendar1 click()
   call window.external.MGExternalEvent(document.Calendar1.Year & "/" & document.Calendar1.Month & "/" &
document.Calendar1.Day)
end Sub

</SCRIPT>
```

これは、**MGExternalEvent** を発行します。そして、'/' によって区切られた 1 つの文字列で 3 つの日付の値を送信します。1 つのパラメータ（BLOB）だけが返されるため、この設定は重要です。



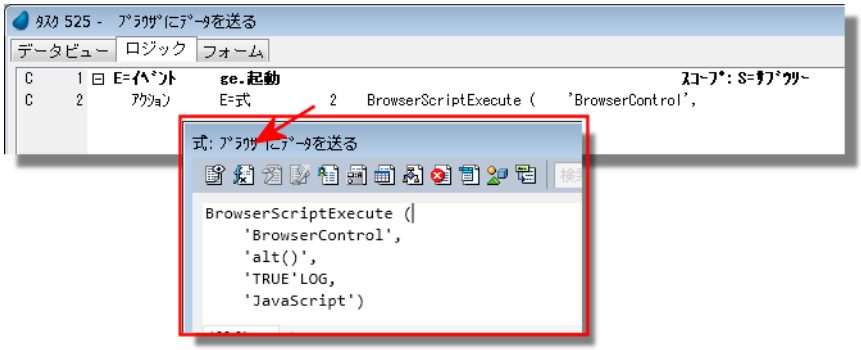
次に、リッチクライアントプログラムでは、そのイベントを処理する必要があります。これは、**外部イベント**と呼ばれる内部イベントに対応した**ロジックユニット**を作成することで実現できます。

ロジックユニット内では、送られた BLOB データを解析する処理を定義します。この例では、BLOB データはテキストのため、BLOB 項目の**内容**特性を ***Ansi*** として定義しています。これによって、BLOB 上で直接 **DVal()** 関数を使用することができます。Magic xpa で使用できる様々な文字列解析機能（**StrToken()** や色々な 変換関数）を、この種の分析に利用することができます。

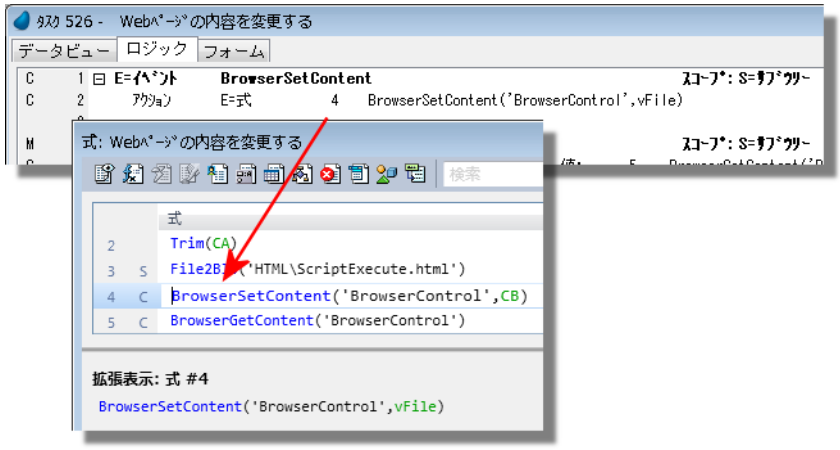
注： **MGExternalEvent** は、ブラウザクライアントで使用されるものと定義方法が異なりますので注意してください。

Web ブラウザにデータを送信する

対話形式で Web ブラウザにデータやコマンドを送ることもできます。



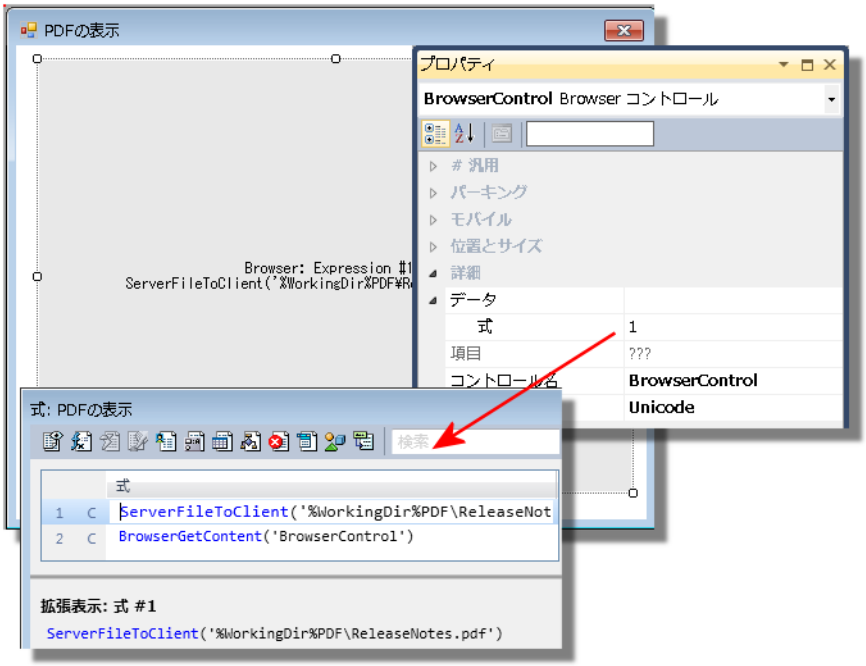
これを実現する 1 つの方法は、**BrowserScriptExecute()** 関数を使用することです。この関数は、Web ブラウザにスクリプトを送って実行することができます。デフォルトでは JavaScript が使用されますが、他の言語を指定することもできます。



BrowserSetContent() 関数を使用することにより、Web ページの全ての内容を変更することもできます。これは、ブラウザコントロールが表示しているすべての html の内容を置き換えます。また、**BrowserGetContent()** 関数を使用すると、Web ページの全てのコンテンツを取得することができます。

注： 別の方法として .NET のブラウザコントロール（**System.Windows.Forms.WebBrowser**）を使用することができます。これは、すべての .NET コントロールと同じように、より低レベルで制御しますが、より高い学習曲線を持っています。

リッチクライアントフォーム内に PDF を表示させるには



帳票出力やテキストデータをユーザに表示する 1 つの方法が PDF に変換することです。リッチクライアントでは、PDF を簡単に表示させることができます。

1. ブラウザコントロールをフォームに配置します。
2. コントロールの**データ**特性に PDF のファイル名を設定してください。例で示したように、ハードコードで指定することもできますが、変数に格納することで実行時に変更することができます。ファイル名は、URL で指定する方法とサーバのパス名で指定する方法があります。
   - URL で指定……PDF ファイルにアクセスするための URL を指定します。この場合、PDF ファイルを、Web サーバ上のエイリアスが定義されている場所に置く必要があります。
   - サーバのパス名で指定……ブラウザコントロールの**データ（式）**に **ServerFileToClient（）**関数を定義して、パラメータとしてサーバ上の PDF ファイルの位置をフルパスで指定します。

ブラウザコントロールを使用したプログラムの作成方法については、「ブラウザコントロールで実行している Web ページを操作するには」（807 ページ）を参照してください。

サーバとクライアントの間でファイル転送を行うには

リッチクライアントプログラミングにおいて、プログラマは（サーバ側とクライアント側の）2 つの OS 環境上でプログラムが動作していることを意識しておく必要があります。1 つの場所にファイルを置き、必要に応じて転生するようにした方がより効率的です。

Magic xpa は、2 台の PC 上のファイルを処理するために複数のリソースを提供しています。ここで、クライアントかサーバにあるファイルの位置を指定することができます。

次の 2 つの関数は、明示的にファイルの転送処理を行います。

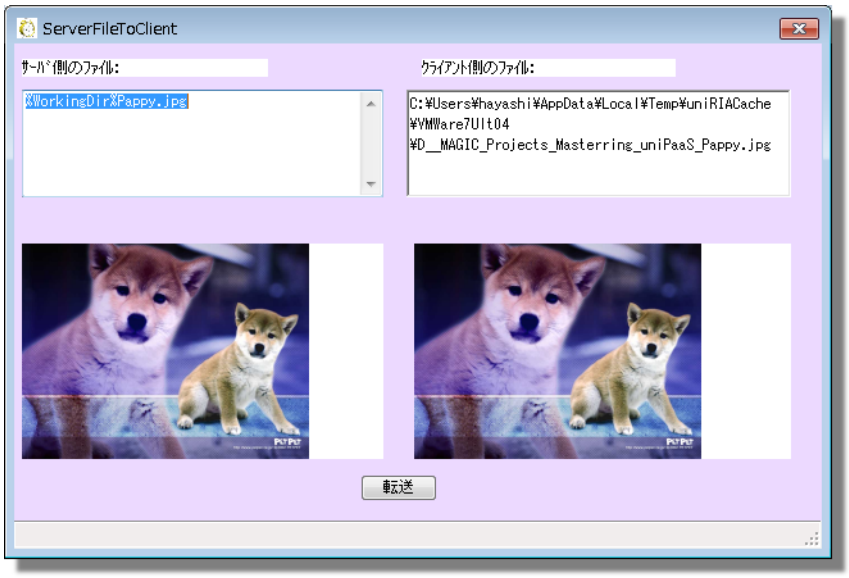
- **ServerFileToClient()**
- **ClientFileToServer()**

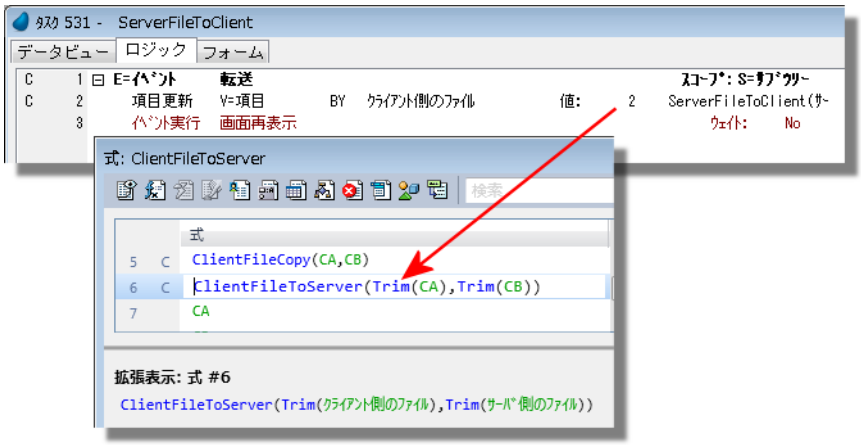
ServerFileToClient()

この例では、ファイルがサーバからクライアントに転送されます。

左側は、. サーバ上のイメージを表示しています。右側では、クライアント側のイメージが表示されています。

プログラマとして、サーバファイルが配置されている場所を管理するようにします。しかし、クライアント PC 上は管理しません。



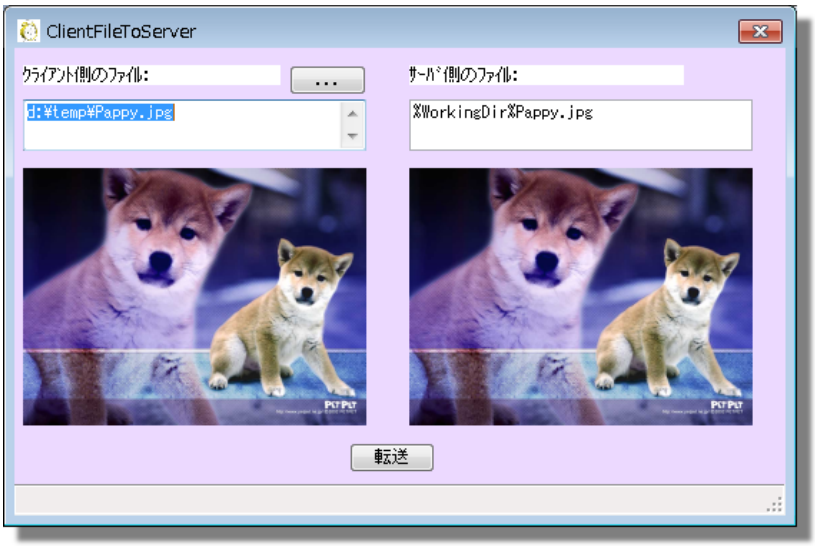


転送ボタンが押下されると、**ServerFileToClient()** 関数が実行されます。これは、Magic xpa の管理下となっている場所から、クライアントにファイルを転送し、プログラムに（パス名を含んだ）ファイル名を返します。

ClientFileToServer()

エンドユーザがサーバにファイルを転送したい場合、ファイル名はクライアント側で知っていることが前提になります。エンドユーザは、どのファイルを送るべきか、選択する処理が発生することがよくあります。

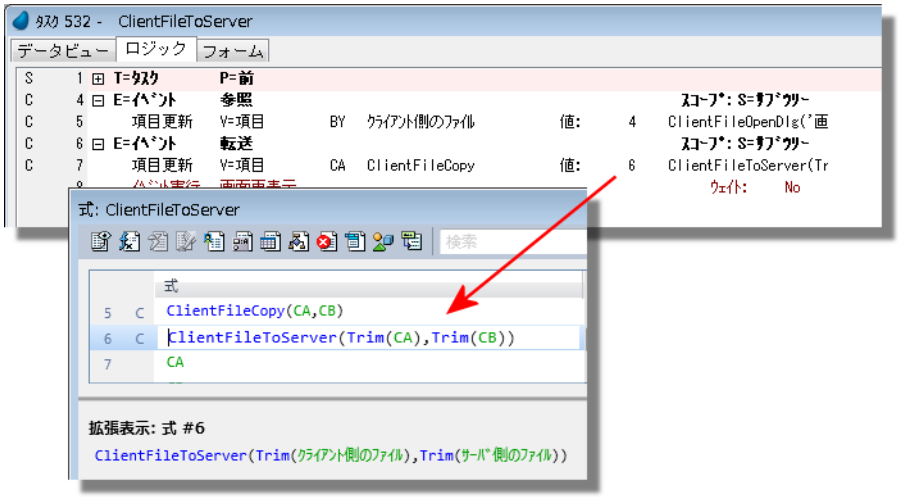
この例では、**ClientFileOpenDlg ()** 関数を使用してファイルを選択するズームボタンが提供されています。



次に、**Flip()** と **StrToken()** 関数を利用して、ファイル名が解析され、ディレクトリパスが外されます。

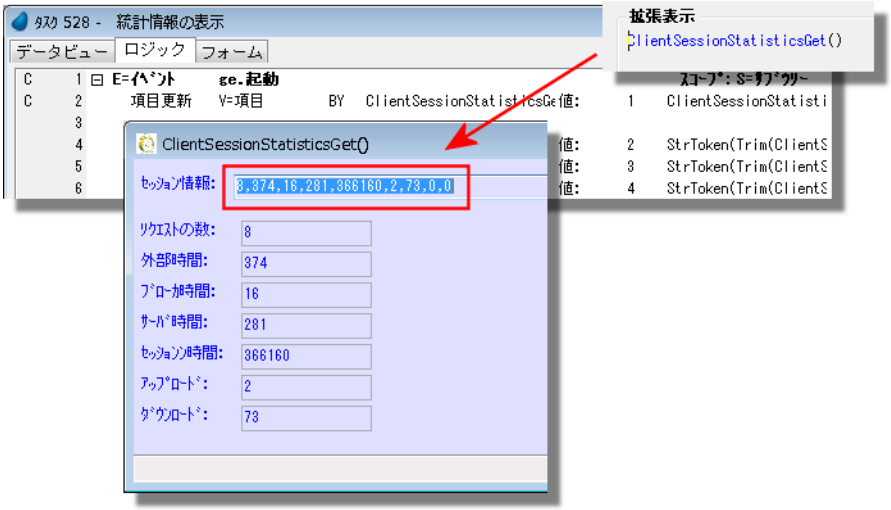
```
Flip(StrToken(Flip(Trim(A)),1,'¥'))
```

この場合、文字列「Pappy.jpg」の結果は、サーバ側の位置を表す論理名に連結されます。この場合、転送先はハードコードされていません。



最後に、**ClientFileToServer()** 関数が使用され、ファイルが転送されます。転送処理が正常に動作しなかった場合、戻り値のコードを使用してエンドユーザにエラーメッセージを表示させることができます。

リッチクライアントアプリケーションの微調整を行うには



リッチクライアントアプリケーションの実行中に、実行中の統計情報を収集することができる、いくつかの方法があります。

最初に、ブローカモニタを使用する方法です。これは、実行しているすべてのアプリケーションをモニタすることができます。

各セッションに対して、**ClientSessionStatisticsGet()** 関数を使用することもできます。この関数は、カンマ区切りの文字列を返します。この文字列は、**StrToken()** 関数を使用することで、上記のような様々な情報に解析することができます。

注： プログラムを実行する前後にこの関数を使用することで、統計情報がどのように変化したか確認することができます。

```
<body onload="initialize()">
   <xml id="rcExecProps">
      <properties>
         <property key="protocol" val="http"/>
         <property key="server" val="ServerName"/>
         <property key="requester" val="/Magic3xScripts/Mgrqispi.dll"/>
         <property key="appname" val="Rich Internet Samples"/>
         <property key="prgname" val="Menu"/>
         <property key="arguments" val=""/>
         <property key="envvars" val=""/>
         <property key="UseWindowsXPThemes" val="N"/>
         <property key="HTTPCompressionLevel" val="Normal"/>
         <property key="DisplayStatisticInformation" val="Y"/>
         <property key="InternalLogLevel" val=""/>
         <property key="InternalLogFile" val=""/>
         <property key="InternalLogSync" val="Session"/>
         <property key="LogClientSequenceForActivityMonitor" val="N"/>
      </properties>
   </xml>
```

Magic xpa Studio では、MAGIC.INI ファイルの［MAGIC_RIA］セクションを以下のように設定するとステータスバーの右の領域上にマウスカーソルを置くことで、実行中のプログラムの統計情報が表示されます。

```
[MAGIC_RIA]
DisplayStatisticInformation=Y
```

この機能は、実行時に利用できるように設定できます。**Publish.html** ファイルの **DisplayStatisticInformation** 設定を ***Y*** に設定することで、エンドユーザはステータスバーの右領域上にマウスカーソルを置くことで統計情報が参照できます。

この場合は、各セッションの終了後にこれらの統計情報を示すダイアログが開きます。

リッチクライアントのログオンダイアログの言語を変更するには

Magic xpa が実行中の場合、MLS 機能を利用することで画面上の対話表示を変更することができます。例えば、あるユーザが英語で表示している時に別のユーザが日本語で表示させることができます。しかし、ユーザがリッチクライアントアプリケーションにログインする際、最初に表示されるダイアログは、HTML ダイアログ（**publish.html**）になります。この時点では、ダイアログを表示しているのは Web ブラウザになり、MLS は機能しません。

このため、ログオンダイアログの言語を変更にするには、単に **publish.html** ファイル内のテキストを変える必要があります。リッチクライアント・アプリケーションを公開する場合、**publish.html** ファイルが作成されます。しかし、これは単純な HTML です。必要に応じてそれを修正することができます。リッチクライアントアプリケーションがどのように実行されるかについての詳細は、「リッチクライアントプログラムにパラメータを渡すには」（814 ページ）を参照してください。

```
<body onload="initialize()">
   <xml id="rcExecProps">
      <properties>
         <property key="protocol" val="http"/>
         <property key="server" val="ServerName"/>
         <property key="requester" val="/Magic3xScripts/Mgrqispi.dll"/>
         <property key="appname" val="Rich Internet Samples"/>
         ...
         <property key="LogonWindowTitle" val="ログオン"/>
         <property key="LogonGroupTitle" val="ログオンパラメータ"/>
         <property key="LogonMessageCaption" val="ユーザ ID とパスワードを入力してください."/>
         <property key="LogonUserIDCaption" val="ﾕｰｻﾞ ID"/>
         <property key="LogonPasswordCaption" val="ﾊﾟｽﾜｰﾄﾞ"/>
         <property key="LogonOKCaption" val="OK"/>
         <property key="LogonCancelCaption" val="ｷｬﾝｾﾙ"/>
      </properties>
   </xml>
```

Magic.ini ファイルの **InputPassword** パラメータを ***Yes*** に設定することで、**ウエルカム**ダイアログを表示した後に、Magic xpa のログインダイアログを表示させることができます。ログインダイアログも MLS が有効になる前に表示されます。しかし、HTML のウェルカムダイアログが閉じた後のため、テキストを使用して直接制御することもできません。その代わりに、Magic xpa は、ログオンダイアログのテキストに必要な値を設定することができる特別な環境変数を提供しています。

これらの環境変数は、リファレンスヘルプの公開される Web ページのトピック内に列挙されています。これらの環境変数を有効にするには、上記のように publish.html ファイルに変数を定義するだけです。

リッチクライアントプログラムにパラメータを渡すには

グローバル変数を使用することで、**Publish.html** ファイルから Magic プログラムにパラメータを渡すことができます。この機能を使用することで、実行時にコマンドラインで HTML や他のクライアントの環境変数に送られたデータを受け取ることができます。

これは実現するには、以下のようにします。

```
/*
 * launch the application
*/
function launchApplication()
    {
     var sRequiredCookies = collectRequiredCookies();
     var sAppQuery = "";
     if (sRequiredCookies.length > 0)
        sAppQuery = "?cookies=" + sRequiredCookies;
        var sHREF = "http://ServerName/Magic3xRIAApplications/ApplicationName/
ApplicationName.application" + sAppQuery;
        //document.write("Application query: " + sHREF + "<BR/>");
        window.location.href = sHREF;
```

1. **Publish.html** は、アプリケーションの公開時に作成されるマニフェストファイルを使用して Magic xpa を呼び出します。デフォルトの **Publish.html** 内に **launchApplication()** 関数があります。
2. この呼び出し処理にパラメータを追加してください。例えば、userLoc と userZip と呼ばれる 2 つのパラメータを渡す場合、以下のように定義します。

```
window.location.href = sHREF + "?userLoc=Atlanta&UserZip=91122";
```

3. Magic タスク内では、これらの変数の値を受け取るために **GetParam()** 関数を使用します。この例では、GetParam ('userLoc') と GetParam ('userZip') を実行することで渡された情報を受け取ることができます。
4. 値が可変のパラメータを渡したい場合は、以下のようにカンマ区切りの文字列を arguments パラメータに指定することで実現できます。そして、**GetParam()** 関数で MGARG1、MGARG2 などを使用することで値を取得することができます。

```
<properties>
     <property key="protocol" val="http"/>
     <property key="server" val="ServerName/>
     <property key="requester" val="/Magic3xScripts/Mgrqispi.dll"/>
     <property key="appname" val="Rich Internet Samples"/>
     <property key="prgname" val="Menu"/>
     <property key="arguments" val="Param1,Param2,Param3,Param4"/>
     <property key="envvars" val=""/>
     <property key="UseWindowsXPThemes" val="N"/>
     <property key="HTTPCompressionLevel" val="Normal"/>
```

参照 : この関数の使い方については、リファレンスヘルプの **GetParam()** のトピックを参照してください。

リッチクライアントプログラムに Cookie を送るには

リッチクライアントアプリケーションの起動時にクライアントの Cookie を送ることができます。

```
/*
 * collect selected cookies that will be sent by the RC on every request to the server
*/
function collectRequiredCookies()
{
  ...
   var aRequiredCookies = new Array();
   aRequiredCookies[0] = "Name1";
   aRequiredCookies[1] = "Name2";
```

1. publish.html（アプリケーション公開用 HTML ファイル）を開き、collectRequiredCookies（）という名前の Jscript 関数を探します。
2. collectRequiredCookies（）関数の中で、aRequiredCookies という名前の配列にアプリケーションに渡したい Cookie の名前を定義します。上記の例では、”Name1” と ”Name2” という名前の Cookie が渡されることを意味しています。

publish.html の launchApplication() 関数によって collectRequiredCookies() が呼び出され、次に RIA アプリケーションが以下の書式で起動されます。

```
http://myserver/Magic3xRIAApplications/myapp/my.application?cookies=Name1=val1,Name2=val2.
```

リッチクライアントアプリケーション側では、GetParam（）関数を使用することで Cookie の値を取得することができます。

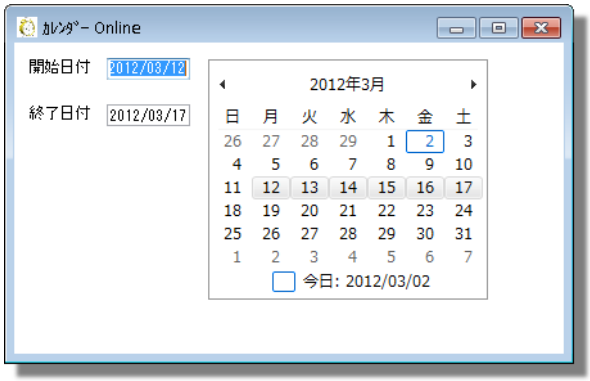
注： クライアント PC 側に ”Name1” と ”Name2” の Cookie が書き込まれている場合、その値（”val1” 及び ”val2”）が送られます。Cookie が書き込まれていない場合は、何も送られません。

リッチクライアントアプリケーションが一旦インストールされると、起動用のショートカットメニューがスタートメニューに作成されますが、ここから起動した場合は、Cookie は送られません。

［このページは意図的に空白にしています。］

第 40 章 : .NET 統合

カレンダコントロールを追加するには



.NET で動作する最初のプロジェクトは、カレンダコントロールが最適です。.NET のカレンダコントロールには、通常の Windows のカレンダコントロールより多くの特徴があります。そして、オンラインおよびリッチクライアントのプログラミングを行う上でより有益であることがわかります。このプロジェクトは、この章の他の節で記述されている処理でも利用できます。また、.NET での様々な機能を利用した実証例として使用することもできます。

.NET コントロールを使用するには、最初にモデルや変数項目にオブジェクトタイプを定義する必要があります。

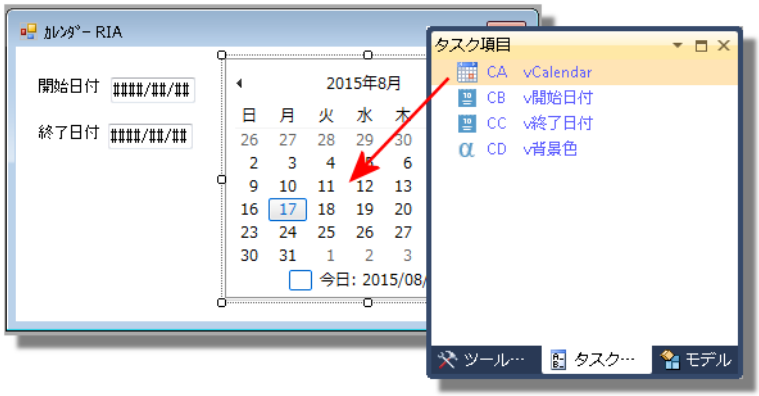
.NET モデルを使用する

1. .NET のコントロールモデルを作成し、**オブジェクトタイプ**特性には、以下のように入力します。
   System.Windows.Forms.MonthCalendar
2. .NET モデルに対応したマッピングルールを定義します（「.NET コントロールにデータをリンクさせるには」（819 ページ）を参照）。
3. リッチクライアント（またはオンライン）タスクを作成します。
4. フォームにカレンダコントロールを配置します（「.NET コントロールを定義するには」（818 ページ）を参照）。
5. **.NET** コントロールに .NET のコントロールモデルを割り当てます。
6. 日付型の変数項目を追加し、**.NET** コントロールのデータ特性に設定します。この変数項目の値を変更すると、.NET コントロールの表示に反映されます。そして、.NET コントロールを変更することでデータ項目も変更されます。

.NET 項目を使用する

1. リッチクライアント（またはオンライン）タスクを作成します。
2. System.Windows.Forms.MonthCalendar に対応した .NET 型の変数項目を作成します（「.NET オブジェクトを定義するには」（825 ページ）を参照）。
3. フォームにカレンダコントロールを配置します（「.NET コントロールを定義するには」（818 ページ）を参照）。
4. ユーザが選択したカレンダ上の値を返すイベント処理を作成します（「.NET オブジェクトのプロパティを変更するには」（823 ページ）を参照）。
5. 必要に応じてカレンダコントロールのプロパティをカスタマイズします（「.NET オブジェクトのプロパティの値を取得するには」（826 ページ）を参照）。
6. オプションで、インタラクティブに特性を設定することで、ユーザに対しカレンダをカスタマイズさせることができます（「.NET の 列挙型を使用するには」（830 ページ）を参照）。

.NET コントロールを定義するには



.NET オブジェクトによっては、表示しないで動作するものもあります。しかし、Form コントロールでは、ユーザ用にウィンドウに表示します。Magic xpa は、.NET のインタラクティブなコントロールを簡単に処理することができます。これには、以下のようにしてください。

.NET コントロールを使用するには、最初にモデルかデータ項目でオブジェクトタイプを定義する必要があります。

表示可能なコントロールの場合は、オブジェクト・タイプが System.Windows.FormsControl でなければなりません。

.NET モデルを使用する

1. .NET のコントロールモデルを作成し、**オブジェクトタイプ**特性を定義します。
2. フォームに **.NET** コントロールを配置し、.NET コントロールモデルを割り当てます。

.NET 項目を使用する

1. .NET 型のデータ項目を作成して、オブジェクトタイプ特性を定義します。
2. 次に、項目パレットからデータ項目を選択してフォームに配置します。

.NET コントロールを配置した後、さまざまな特性を設定することができます。例えば、カレンダーの場合、位置特性を (0,100,0,100) と設定すると、右または下に拡張することでより多くの月を表示させることができます。

参照 : 「.NET コントロールのプロパティの値を変更するには」（827 ページ）

.NET コントロールにデータをリンクさせるには

Magic xpa のコントロールと同じように、コントロールのデータ特性を使用することで、簡単にデータとリンクさせることができます。

.NET コントロールには、多数のプロパティとイベントが利用できるため、データ項目をデータ特性に割り当てる前に、の .NET のプロパティがデータ項目の値を取得するか、また、どの .NET イベントがデータ項目に .NET プロパティの値を設定するかを定義する必要があります。

この定義は、モデルレベルで **Value プロパティ名**や **Value Changed イベント名**特性で行います。

次に、.NET コントロールのモデルに .NET バインディング宣言を追加します。

この定義は、データをリンクさせたいコントロールタイプ毎に行うことができます。

1. ***GUI 表示***または***リッチクライアント表示***クラスのモデルを作成し、その型を .NET に設定します。
2. **モデル特性**を開き、**オブジェクトタイプ**特性を定義（例えば System.Windows.Forms.MonthCalendar）します。

注： Dot と入力することでオートコンプリート機能を使用することができます。利用可能なオプション一覧が（Magic の内部関数を選択する時のように）表示されます。

3. **Value プロパティ名**特性にデータ項目にリンクさせるプロパティ（例えば SelectionStart）を設定します。
4. **Value Changed イベント名**特性にデータ項目を更新するためのトリガにリンクするイベント（例えば DateChanged）に設定します。

これで、モデルをプログラムで使用することができます。

次にプログラムで .NET コントロールのモデルを使用します。

5. 新しいプログラムを作成します。
6. このプログラム内でデータ項目を作成します。このデータ項目は、コントロールのデータを取得したり設定するために使用されます。
7. フォームにズームして、.NET コントロールをフォームに配置します。
8. .NET コントロールに作成した .NET モデルを割り当てます。

注： オブジェクトタイプが System.Windows.Forms.Control から継承している場合のみ、フォーム上に配置することができます。

9. **データ**特性に作成したデータ項目を割り当てます。

次に、データ項目の値を変更すると .NET コントロールの表示に反映されます。そして、コントロールを変更するとデータ項目も更新されます。

.NET コントロールにデータビューをリンクさせるには

タスクのデータビューを **.NET** コントロール（例えば、Grid コントロール）にリンクさせることができます。これを Magic の**テーブル**コントロールの代わりに利用することができます。

フォーム上に複数の **.NET** コントロールを配置することができるため、どの **.NET** コントロールがデータビューを表示して管理するかを指定する必要があります。

この定義は、**.NET** コントロールの**データビューコントロール**特性（フォーム上の**コントロール**特性か、モデルレベル）で行います。

データソースプロパティ名特性に値を入力し、**DisplayMember プロパティ名**特性と **ValueMember プロパティ名**特性に値を設定していない場合、この特性が有効になります。

この特性の値が ***Yes*** に設定される場合、**データビューコントロールのフィールド**特性で*ズーム*することによって［.NET］コントロールで表示されない値を定義することができます。

実行時に、**.NET System.Data.DataTable** オブジェクトがタスクのデータビューから自動的に作成されます、そして、このオブジェクトはモデルレベルの**データソースプロパティ名**特性に定義された **.NET コントロール特性**に割り当てられます。

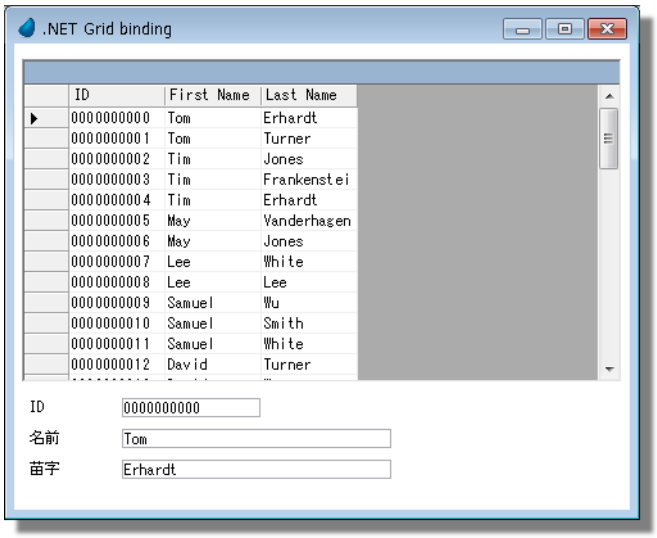
Magic xpa のデータ項目に対する変更内容が **.NET** コントロール上に反映されます。逆もまた同じです。

注： **この機能は、オンラインタスクのみ有効です。**

データソースの内容を表示させるために、.NET の Dataview コントロールを使用するプログラムを作成します：

1. ***GUI 表示***クラスのモデルを作成して、その型を ***.NET*** に設定します。
2. **.NET** コントロールモデルの**特性**シートを開きます。
3. **オブジェクトタイプ**特性に DataSource オブジェクトをサポートする .NET コントロール（例：***System.Windows.Forms.DataGrid***）を設定します。
4. **DataSource プロパティ名**特性にデータを取得するプロパティ（例：***DataSource***）を設定します。
5. **データビューコントロール**特性を ***Yes*** に設定します。（これによって、フォーム上の**コントロール特性**の代わり使用することができます。）
6. 新しいプログラムを作成します。
7. **タスク特性**を開き、**ビュー事前読込**特性を ***Yes*** に設定します。
8. プログラムのデータビューを定義（データソースといくつかのカラムを追加）します。
9. プログラムの**フォーム**エディタを開き、**.NET** コントロールをフォームに配置します。
10. コントロールの**モデル**特性に新規作成したモデルを割り当てるか、コントロールの **.NET オブジェクト**特性に .NET 型の変数項目を割り当てます。
11. コントロールの**特性**シートを開きます。
12. **データビューコントロールのフィールド**特性で*ズーム*して、**.NET** コントロールに割り当てる DataTable オブジェクトを追加します。「0」が設定された場合、データ項目が **.NET** コントロールで利用できないことを意味します。

プログラムを実行すると、データビューが DataGrid コントロールにリンクされていることを確認できます。行間または、.NET コントロール内のデータを更新することで、Magicxpa のデータ項目に反映されたり、その逆の操作を行うことができます。


.NET Grid binding
ID
First Name
Last Name
0000000000 Tom Erhardt
0000000001 Tom Turner
0000000002 Tim Jones
0000000003 Tim Frankenstei
0000000004 Tim Erhardt
0000000005 May Vanderhagen
0000000006 May Jones
0000000007 Lee White
0000000008 Lee Lee
0000000009 Samuel Wu
0000000010 Samuel Smith
0000000011 Samuel White
0000000012 David Turner
ID
0000000000
名前
Tom
苗字
Erhardt

.NET のチョイスコントロールを使用するには

Magic xpa のコントロールを使用する場合と同じように、データをコントロールの**データ**特性を使用している **.NET** コントロールに関連付けることができます。（参照：「「.NET コントロールにデータビューをリンクさせるには」（820 ページ）」）

しかし、Magic xpa のチョイスコントロールのように、.NET のチョイスコントロールも、項目リスト（データソース・オブジェクトとして知られています）を取得することができます。

.NET オブジェクト参照項目を通して .NET コマンドを使用したり、Magic xpa のチョイスコントロールとして動作するように Magic xpa の特性を設定することができます。

ここでは、**.NET** コントロールのデータソースを定義するために Magic xpa の特性を使用することについて説明しています。

.NET コントロールには多くの特性があるため、どの特性がデータソースを取得し、どの特性が値と表示のメンバー名を保持するかを定義する必要があります。

この定義は、モデルの **DataSource プロパティ名**、**DisplayMember プロパティ名**、**Value Changed プロパティ名**、**ValueMember プロパティ名**の各特性で行われます。

次に .NET のデータソース宣言を .NET コントロールモデルに追加します。

この定義は、設定したいデータのタイプに対応する各コントロールタイプ毎に、一度だけ実行されます。

1. ***GUI 表示***または***リッチクライアント表示***の各クラスのモデルを作成し、型を ***.NET*** に設定します。
2. モデルの**特性シート**を開き、**オブジェクトタイプ**特性を定義（例：***System.Windows.Forms.ComboBox***）します。
   オートコンプリート機能が実行され、ドット（.）を入力すると、利用できるオプションが（Magic xpa の**関数一覧**と同じように）リスト表示されます。
3. **Value プロパティ名**特性にデータ項目に連結させるプロパティ名（例：***SelectedValue***）を設定します。
4. **Value Changed イベント名**にデータ項目に値を返すためのトリガとなるイベント（例：***SelectedValueChanged***）を設定します。
5. **Datasource プロパティ名**特性に項目リストを受け取るプロパティ名を（例：***Datasource***）を設定します。
6. **DisplayMember プロパティ名**特性には、表示されるカラムの名前を定義するプロパティ名（例：***DisplayMember***）を設定します。
7. **ValueMember プロパティ名**特性には、データが格納されているカラムの名前を定義するプロパティ（例：***ValueMember***）を設定します。

これでモデルをプログラムで使用することができます。

次に、プログラムで **.NET** コントロールモデルを使用します。

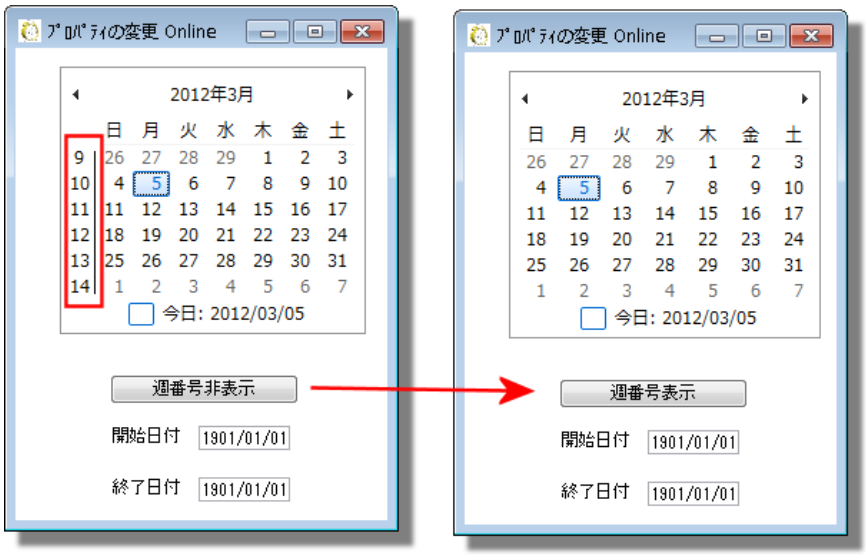
1. 新しいプログラムを作成します。
2. このプログラムに変数項目を作成します。この変数は、コントロールのデータの取得 / 設定を行うために使用されます。
3. フォームを開き、**.NET** コントロールをフォームに配置します。
4. **.NET** コントロールに上記で定義したモデルを割り当てます。

注：　オブジェクトタイプ特性が ***System.Windows.Forms.Control*** **に設定されているモデルを継承している場合のみ、.NET コントロールとしてフォームに配置できます。**

5. **データ**特性に変数項目を割り当てます。
6. コントロールの選択用の特性（例：**選択項目リスト**や**選択表示リスト**）に通常のチョイスコントロールと同じように設定します。

プログラムを実行すると、**.NET** コントロール上に項目リストが表示され、データ項目の変更内容がコントロール上に反映されます。また、コントロールで値を変更するとデータ項目に反映されます。

.NET オブジェクトのプロパティを変更するには

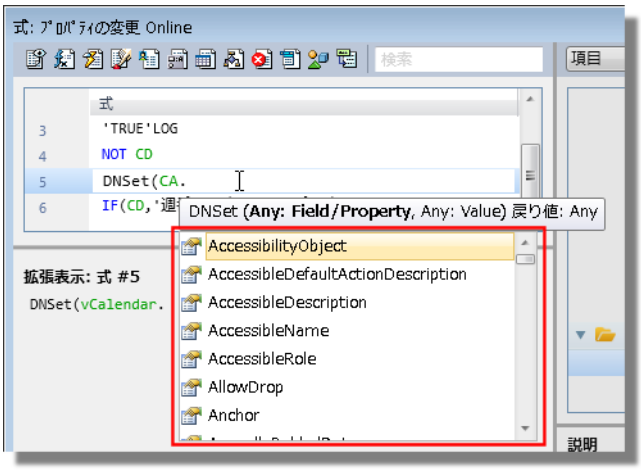


一旦 .NET オブジェクトを定義した場合、そのプロパティを変更するには 2 つの方法があります。

1. オブジェクトがフォーム上に表示される場合、フォームエディタ上でプロパティを編集したり、式を使用して設定することができます。これはコード化が簡単で、即時に実行されます。詳細は、「.NET コントロールのプロパティの値を変更するには」（827 ページ）を参照してください。
2. **DNSet()** 関数を使用することもできます。これによって、指定した時間でプロパティを設定したり、イベントロジックユニットで使用することができます。これについては、このセクションで説明しています。

DNSet() 関数は、.NET オブジェクトのプロパティを設定するために、式で使用されます。以下のように関数を入力します。

1. DNSET(を入力します。
2. .NET オブジェクトの項目 ID（この例では、A) を入力するか、項目一覧から選択します。



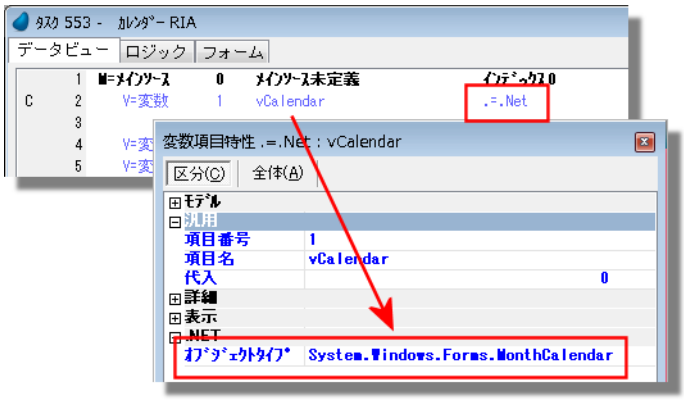
3. ドット（.）を入力してください。この時点で、オブジェクトのプロパティのリストが表示されます。**ShowWeekNumbers** を選択します。

4. カンマ（,）を入力します。
5. プロパティを更新する値を入力します。この例では、論理型変数（W）になります。
6. 括弧を閉じます。

プロパティのリストが表示されるため便利です。この機能は、Magic で関数を定義する場合のオートコンプリートと同じように動作します。***Ctrl+Space*** を押下すると、関数の定義と同じようにもう一度表示されます。しかし、以下のように全ての式を手動で入力することもできます。DNSet(A.ShowWeekNumbers,B)

注： **この例では、プロパティを更新するために使用される値は、単純な論理型の値になっています。しかし、プロパティによっては、列挙型の値を使用したり、これらを提供するために .NET メソッドを使用する必要があります。詳細は、「.NET の 列挙型を使用するには」（830 ページ）を参照してください。**

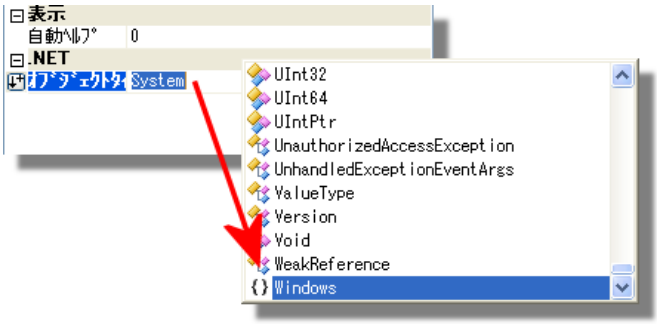
.NET オブジェクトを定義するには



NET オブジェクトを使用するには、最初に変数を定義する必要があります。

1. **データビュー**タブを開き、変数項目を定義します。
2. ***.NET*** のデータ型を選択してください。
3. .NET 項目の**項目特性**で、**オブジェクトタイプ**を入力します。

ここでは、任意でいくつかのことが可能です。他のプログラミング言語で作成された .NET コードを使用する場合、テキスト名を入力したり、コピー / ペーストを使用することができます。大文字小文字の違いやスペースの数に気をつけて、正確に名前を入力してください。



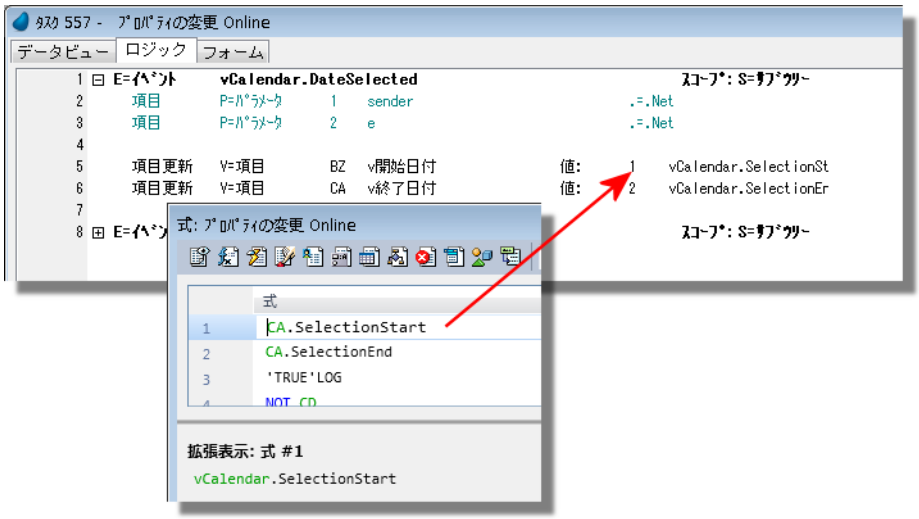
また、入力内容がはっきりしなかったり、キー入力したくない場合、いくつかの選択肢から選びたい場合は、Magic xpa の選択リストを使用することができます。ドット（.）を入力すると、オプションリストが表示されます。ここから 1 つを選択し、***Enter*** を押下してください。次のドット（.）でも同じように操作します。

頻繁にアクセスするライブラリに対して、エイリアスを使用することもできます。これにより、キー入力操作を省くことができます。

これで、.NET オブジェクトを作成することができました。今後、このオブジェクトを使用することができます。次は、これをフォームに配置したり、イベントで利用することができます。

注：　モデルを使用することで、上記の作業のほとんどを簡素化することができます。使用したい .NET オブジェクトを簡単に見つけることができるため、モデルを使用することを推奨します。

.NET オブジェクトのプロパティの値を取得するには



.NET オブジェクトを使用する場合、そのプロパティにアクセスする必要があります。例えば、このカレンダプログラムでは、ユーザによって選択された日付や日付範囲を保存します。

1. 適切な時間に実行されるロジックユニットを作成します。この例では、ユーザが日付または日付範囲を選択したときに、ロジックユニットが実行されるように、.NET イベントが使用されています。
2. 戻り値を保持する変数項目を作成します。この場合、開始日と終了日が保存されます。
3. .NET のプロパティにアクセスするための式を作成します。これらは、カレンダ用の ,NET 項目 (T) を選択するか、キー入力し、ドット（.）を入力します。ドットが入力されると、プロパティのリストが表示されます。選択範囲を取得するため、**T.SelectionStart** と **T.SelectionEnd** が必要となります。
4. .NET 式で Magic の変数項目を更新するため、**項目更新**処理コマンドを使用します。

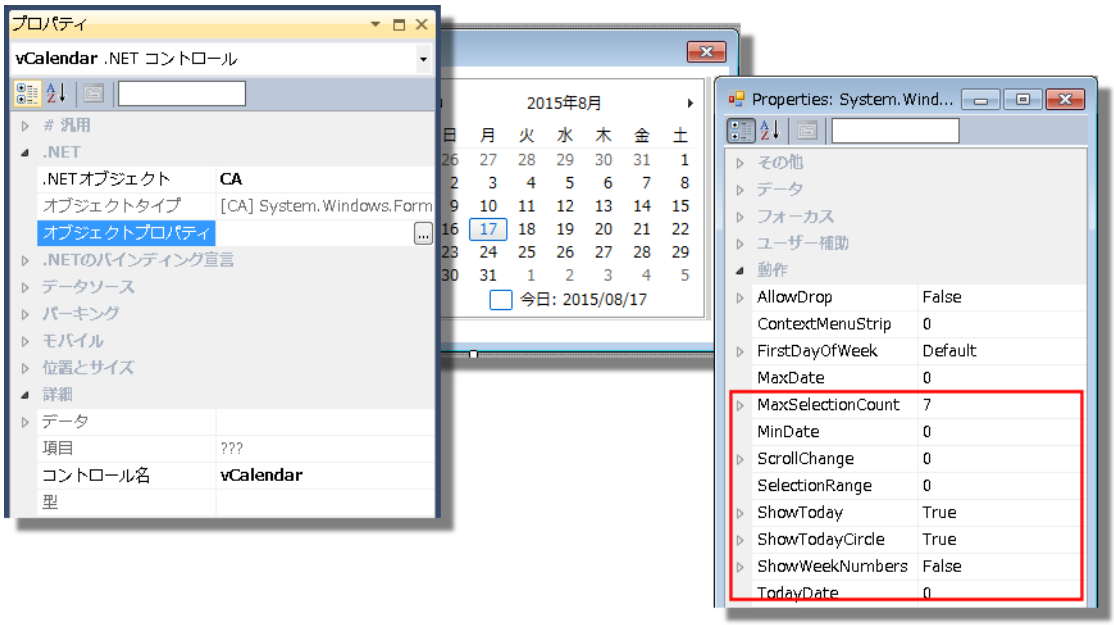
プロパティのリストが表示されるため便利です。この機能は、Magic で関数を定義する場合のオートコンプリートと同じように動作します。***Ctrl+Space*** を押下すると、関数の定義と同じようにもう一度表示されます。しかし、以下のように全ての式を手動で入力することもできます。

.NET イベントを使用したロジックユニットを作成すると、通常 2 つのパラメータ（sender と e）が作成されます。

これらは標準的な .NET パラメータです、そして、これらのパラメータを表示可能な値に変更するために **DNCast** と **DotNet** メソッドを使用して、Magic プログラム内で利用することができます。sender は、イベントを発行した **System.Object** です。e はイベントによって渡された **System.EventArgs** 引数です。

注： この例では、単純な .NET の DateTime データがプロパティから返されます、そして、Magic xpa は必要に応じてそれを変換します。しかし、プロパティによっては、列挙型の値を使用したり、これらを提供するために .NET メソッドを使用する必要があります。詳細は、「.NET の 列挙型を使用するには」（830 ページ）を参照してください。

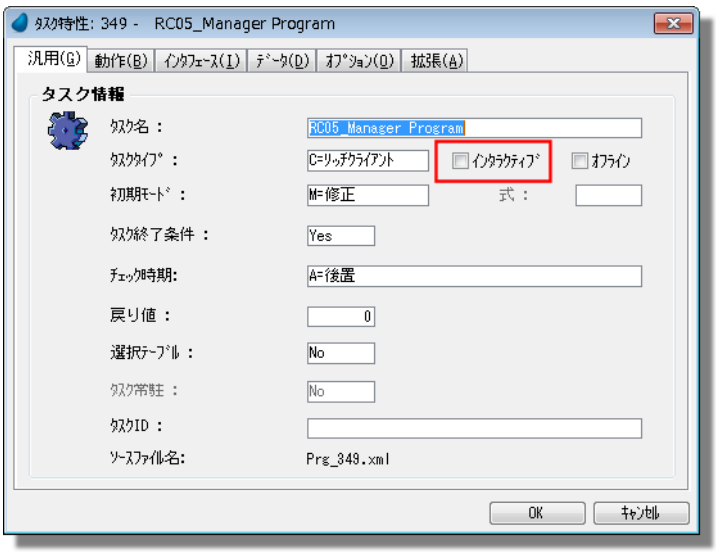
.NET コントロールのプロパティの値を変更するには



.NET コントロールには、標準的な Magic の**特性シート**があります。ここには、**位置**特性や**パーキング可**特性などがあり、他の Magic コントロールと同じように設定することができます。たとえば、この例では、ユーザがウインドウを拡大するとカレンダ表示も拡大するように、**位置**特性が設定されています。その際、複数月が表示されるようになります。

しかし、**特性シート**の一番下には特別な特性（**オブジェクトプロパティ**特性）があります。この特性から**ズーム**すると、このオブジェクトのための内部の .NET プロパティにアクセスすることができます。

このプロパティシートは、各 .NET オブジェクト毎に異なります。ここでは、実行時の表示内容を設定することができます。この例では、**MaxSelectionCount** が ***60*** に設定されています。これは、ユーザが最高 60 日にわたる日付範囲を選択できるようにすることを意味しています。

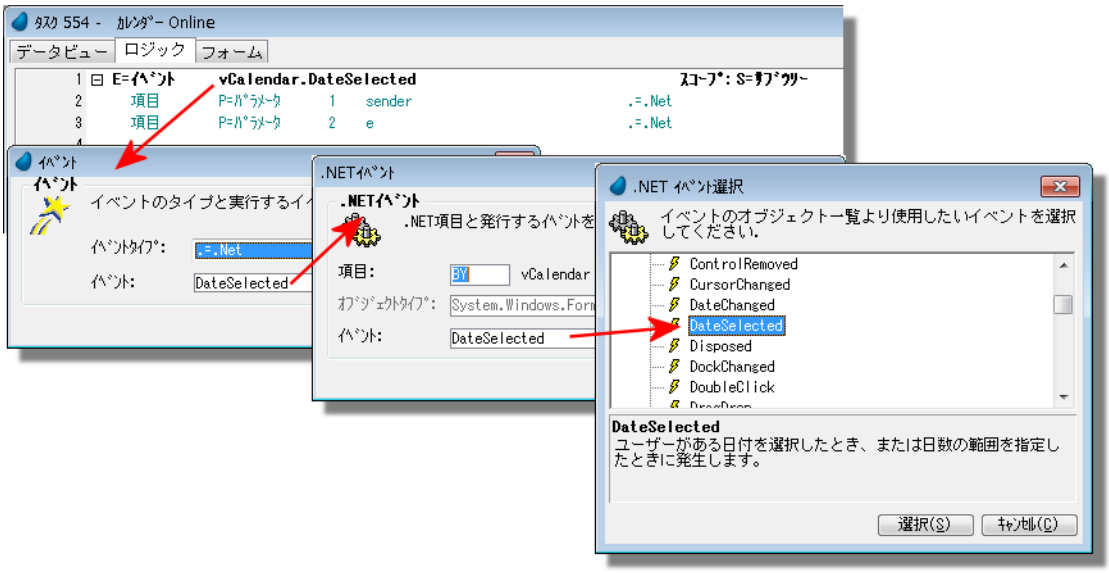


式を使用することにより、動的にプロパティを変更することもできます。右側のカラムで ***0*** が表示されている場所からズームすることで、通常の Magic 特性と同じように式を設定することができます。イベントや処理コマンドを別途定義する必要がなく、フォームに即時に反映されるため、オブジェクトのプロパティを変更する方法として便利です。

カーソルを .NET 特性の上に置くと、特性シートの下部にヘルプが表示されます。この場合、System.Drawing.Color のヘルプが表示されています。これは、式を使用して背景色を設定するために、オブジェクトタイプが System.Drawing.Color の .NET オブジェクトを使用しなければならないことを示しています。列挙型を追加することもできます。「.NET の 列挙型を使用するには」（830 ページ）も参照してください。

参照： 「.NET オブジェクトのプロパティを変更するには」（823 ページ）.

.NET オブジェクトのイベントを処理するには



.NET オブジェクトで動作しているとき、オブジェクトから発行されるイベントに基づいたロジックを実行させたい場合があります。これは、Magic xpa のイベントロジックユニットを使用して実現できます。.NET イベントの処理は、Magic xpa で他のイベントタイプを処理する場合と同じように行うことができます。

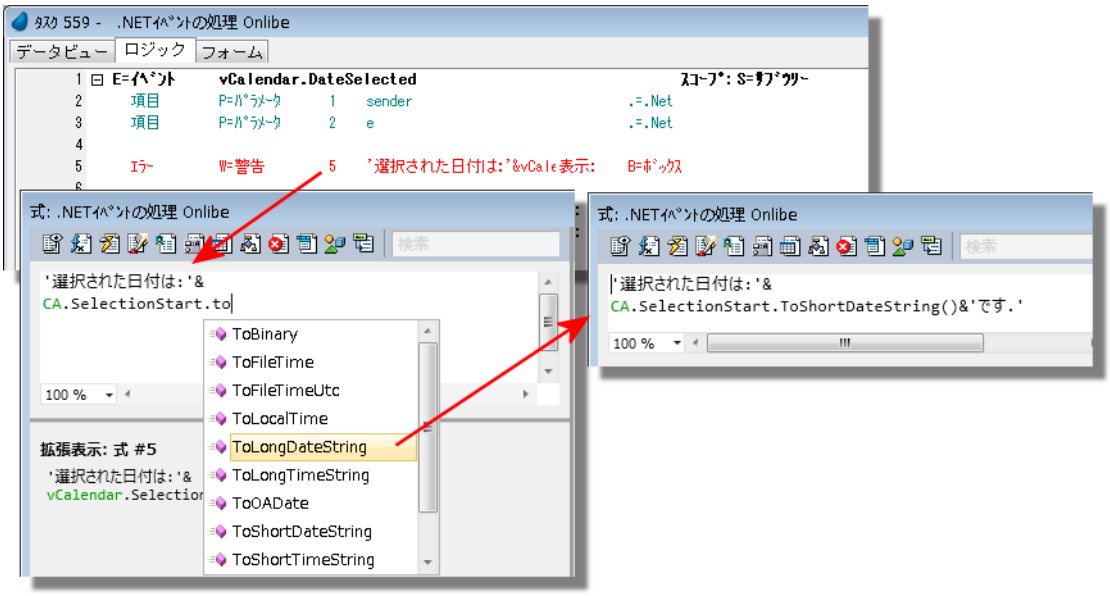
1. ロジックタブを開き、*Ctrl+H* を押下して新しいロジックユニットを作成します。
2. *E* を入力してイベントを選択します。イベントダイアログが表示されます。
3. イベントタイプで *.NET* を選択します。
4. イベント欄に移動し、ズームします。.Net イベントダイアログが表示されます。
5. 項目欄でズームして、.NET 項目を選択します。
6. イベント欄からズームして、一覧から .NET イベントを選択します。
7. ロジックユニットに戻るまで *Esc* を押下します。

ヒント :*オブジェクト名を知っている場合は、選択しないで、直接入力することもできます。*

これでこのイベントは、同じタイプのすべての .NET オブジェクトに対して、イベントロジックユニットによって処理されることになります。

注 : .NET イベントを使用する場合、伝播の問題は Magic イベントより複雑です。追加情報は、Magic xpa のリファレンスヘルプを参照してください。

.NET メソッドを実行するには



.NET メソッドは、Magic xpa の関数と似ています。しかし、Magic 関数とは異なり、明示的な呼び出し定義を行いません。それは単に .NET 定義の文字列に続けて指定するだけで、その文字列が評価される時に暗黙のうちに実行されます。

.NET メソッドは、オブジェクトによって使用されるデータ型とプログラムで使用されるデータ型の間でデータ変換を行うために必要になる場合があります。この例では、カレンダコントロールは、**DateTime** オブジェクトを返し、メッセージボックス内で使用されます。このため、戻り値を文字データに変換する必要があります。

Magic の **DStr()** 関数を使用した場合は、以下のように式を定義します。

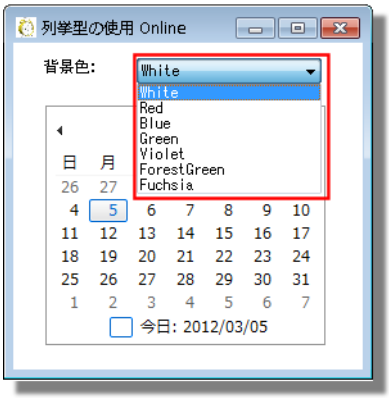
```
DStr(T.SelectionStart,'YYYY/MM/DD')
```

また、.NET の **ToShortDateString()** メソッドを使用する場合は、以下のように定義します。

```
T.SelectionStart.ToShortDateString()
```

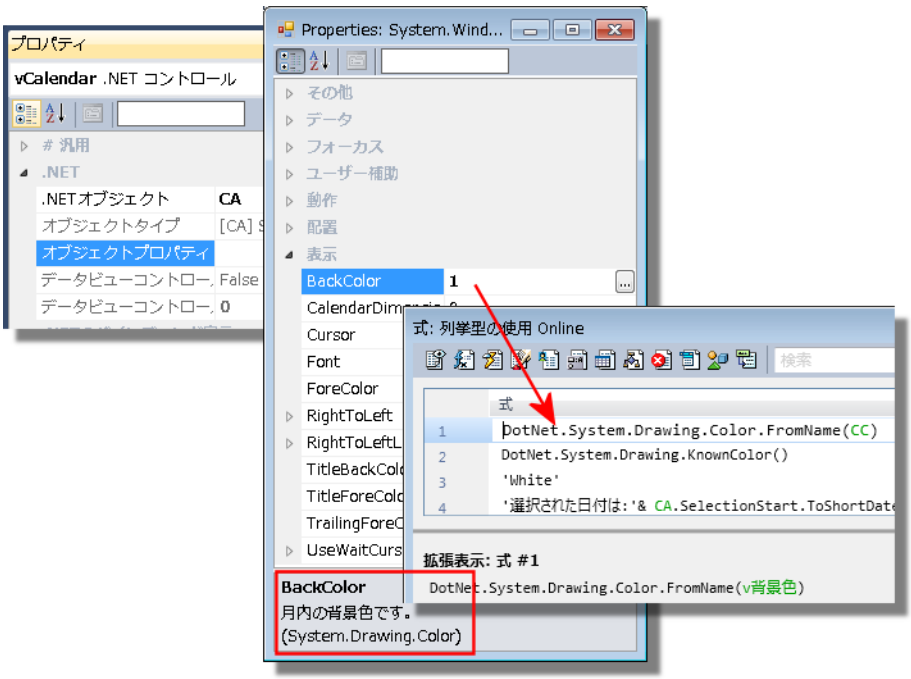
Magic の式エディタで .NET オブジェクトを選択しているとき、その隣に小さなピンクのボックスでメソッドが表示されます。メソッドを選択すると、このメソッドの引数として何を使用したらいいかをトールティップで表示されます。

.NET の 列挙型を使用するには

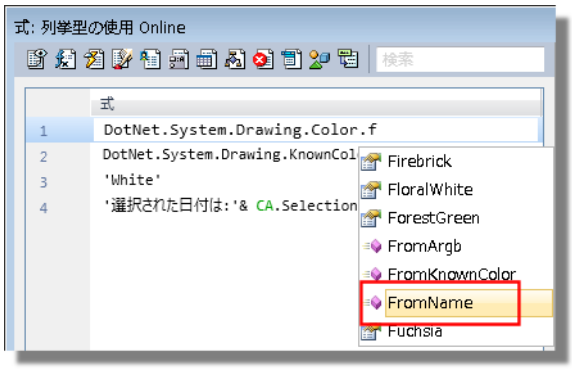


注： このプログラムの動作を確認する場合は、”Windows クラシック ” のテーマを使用するか、**Magic xpa の動作環境→動作設定**の **Windows XP のテーマを使用の設定**を ***No*** に変更する必要があります。

.NET オブジェクトを扱うとき、Magic xpa は大部分のデータ変換を自動的に行います。たとえば、通常、数値が Long/Short の Int 型かどうか、あるいは、文字列を渡すためにどのようなデータ型を使用したらいいかを心配する必要がありません。しかし、.NET オブジェクトの値のいくつかは、列挙型と呼ばれているデータ型です。Magic xpa と .NET の間でデータを渡す唯一の方法は、列挙型を提供したり、それを変換する、.NET メソッドを使用することです。



この例では、**背景色（BackColor ）**のプロパティは、**System.Drawing.Color** クラスの出力が必要になります。Magic xpa では、**特性**シートの下辺にヒントが表示されます。



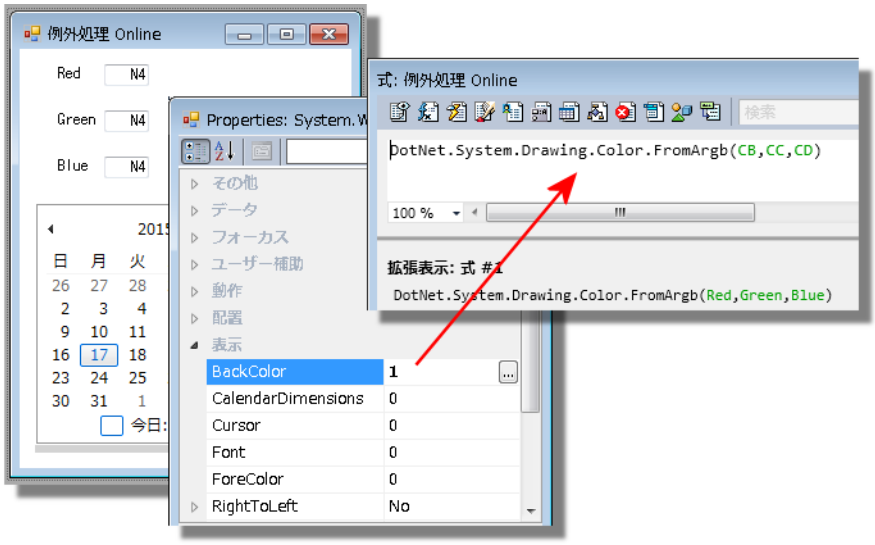
DotNet.System.Drawing.Color を入力すると、ドット（.）を入力するたびに、Magic xpa は有効なオプションのリストを作成します。これらの多くは、値がハードコードされています（例えば、'ForestGreen' または 'Fuchsia'）。しかし、他にデータを変換する関数もあります。このデータ変換は、キャストと呼ばれています。**FromName** は、色の名前を意味する文字列（例えば、'Red または 'ForestGreen'）を **BackColor** プロパティで処理できるように変換するメソッドです。

Magic xpa の式では、以下のように入力できます。

```
DotNet.System.Drawing.Color.FromName('Red')
```

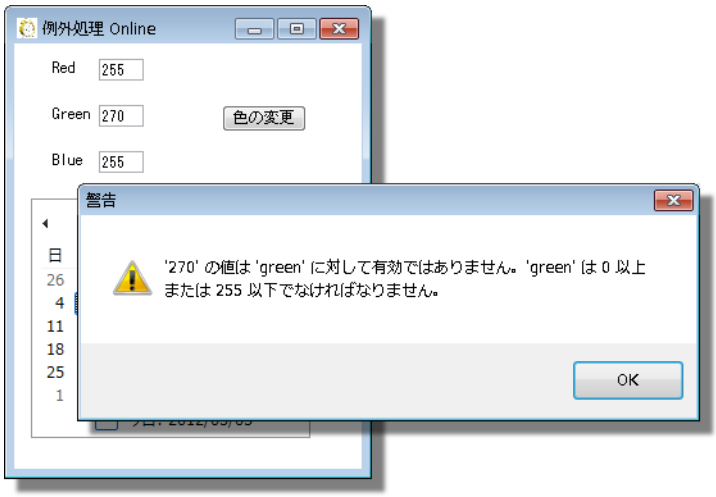
この式が返す値が、**BackColor** プロパティで利用することができます。

通常は、変数項目を使用して色の名前を指定する場合が多いと思います。この場合、文字列 'Red' をユーザの選択内容を格納する文字型項目を定義し、それを使用することになります。単純なコンボボックスで有効な .NET の色リストを表示させることにより、あらかじめ選択された値から選ぶことができます。



FromArgb メソッドを使用して色を変更することもできます。この場合は、一組の RGB 値をパラメータとして指定します。

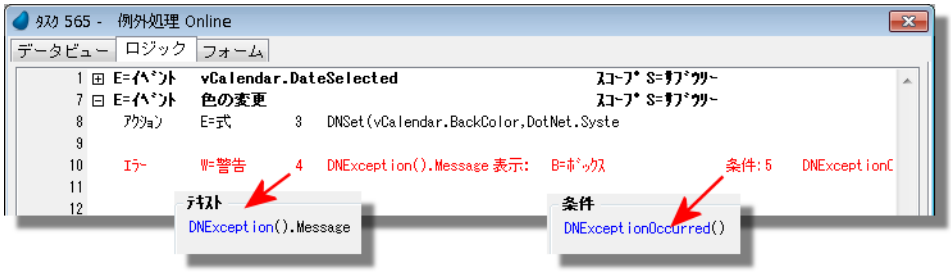
.NET の例外処理を行うには



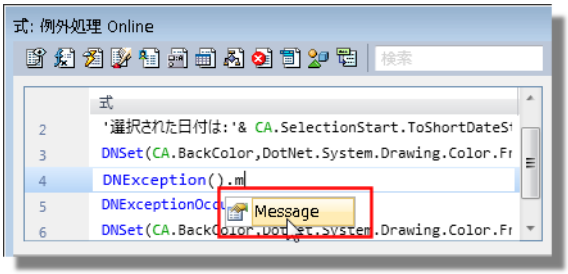
.NET オブジェクトの利用時に、エラーが発生する場合があります。オブジェクトは、これらのエラーを処理せず、どのような対応を取るかを決めてもらうために Magic xpa 側に例外情報を渡します。Magic xpa は、.NET の例外を受け取り、例外処理を実行します。オンラインプログラムでは、ユーザ向けのメッセージを表示するためにエラー処理コマンドを使用することになります。バッチプログラムでは、エラーログにメッセージを記録したり、電子メールを送信することになります。

この単純な例では、ユーザがカレンダの背景として RGB 値を設定できるようになっています。RGB 値は、0 から 256 までの値が指定できます。しかし、このプログラムでは、変数に入力範囲が設定されていません。このため、範囲外の値を入力刷ることができます。

無効な値が .NET に渡されたると、エラー条件は発生します。エラーとエラーメッセージの両方を受け取るには、次のように対応します。

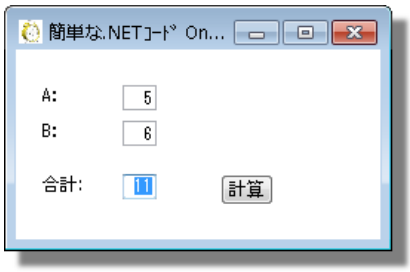


1. 最初に、エラーが発生したかどうかを判定するために、**DNExceptionOccurred()** 関数を使用します。この関数は、直近で実行された .NET の呼び出しでエラーが発生した場合のみ評価されます。

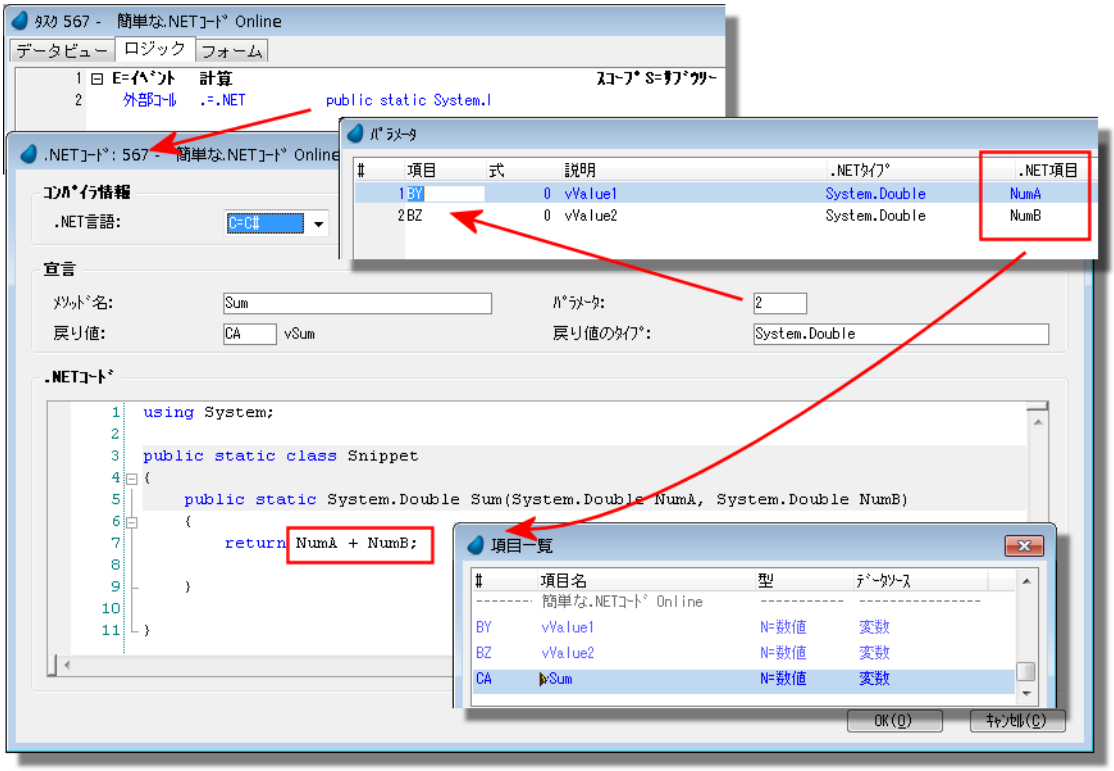


2. 次に、**DNException()** 関数を使用して、エラー処理を実行します。この関数は、.NET の例外オブジェクトを返すため、直接 .NET オブジェクトを使用するように処理を定義することができます。これには、いくつかのオプションがあります。**DNException().Message** を使用すると、ユーザにメッセージを表示させることができます。

簡単な .NET コードを実行するには



.NET オブジェクトを使用するときは、Magic xpa は常にバックグラウンドで .NET コードを実行します。しかし、.NET コードを直接させることもできます。この動作を制御することもできます。



1. 最初に、必要な変数項目を作成します。この例では、3 つ（**vValue1**、**vValue2**、**vSum**）を作成します。全て数値型です。
2. コードを実行する**ロジックユニット**を作成します。この例の場合、計算ボタンをクリックすると**イベント**ロジックユニットが実行されます。イベントは、**強制終了**が***編集***に設定されています。このため、ボタンがクリックされてイベントが発生する前に、フィールドの値は保存されます。
3. **ロジックユニット**内に、**コール .NET** 処理コマンドを定義します。**.NET** 表示の次のカラムから**ズーム**します。
4. ここで、**.NET コード**ウインドウが開きます。**メソッド名**に（半角のアルファベットで）任意の名前を入力してください。
5. **パラメータ**フィールドから**ズーム**します。これで、**パラメータ**ウインドウが開きます。2 つの入力用パラメータを以下のように作成します。
   - ***F4*** を押下して 1 行追加します。
   - **項目**カラムから**ズーム**して項目を選択するか、**式**カラムから**式**テーブルを開き、式を定義します。
   - Magic xpa で推奨するデータ型が **.NET タイプ**に表示されます。このデータ型を使用するか、必要に応じて変更してください。
   - **.NET 項目**に変数名を定義します。ここでは、***NumA*** と ***NumB*** が定義されています。
6. **戻り値**フィールドで**ズーム**して、結果を保持する変数項目を選択します。**戻り値のタイプ**には、推奨するデータ型が表示されます。このデータ型を使用するか、必要に応じて変更してください。
7. 次に、**.NET コード**領域に、任意のコードを入力してください。例えば：

```
return NumA + NumB;
```

は、加算処理です。Magic xpa は、自動的に **NumA** と **NumB** を引数として渡します。その結果は、変数項目 C に**戻り値**として格納されます。

これは、.NET コードの単純な例です。しかし、必要に応じてより複雑なコードを使用することができます。

第 41 章 : モバイル開発

モバイルデバイスのカメラを使用したりギャラリからイメージを選択するには

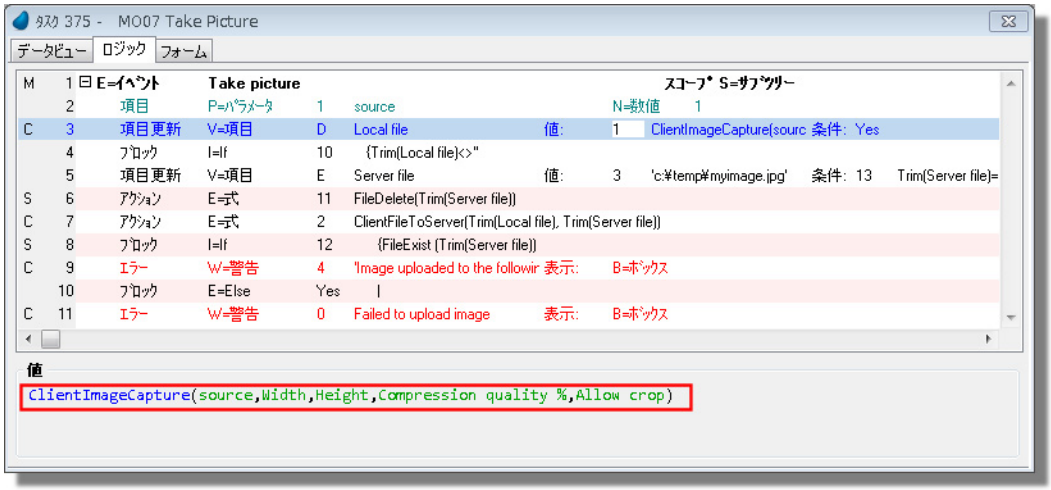
Android や iOS デバイスからカメラを起動したりギャラリーからイメージを選択しすることができます。

カメラやギャラリにアクセスするには :

1. ロジックタブにイベントロジックユニットを定義します。例えば、以下の図のようにズームイベントを割り当てます。
2. 項目更新処理コマンドを定義します。
3. 更新値として **ClientImageCapture()** 関数を定義し、*ソース*パラメータ（1 番目のパラメータ）には、以下の値を指定します。
   - 0 …… **カメラ機能を起動します。**
   - 1 …… **イメージギャラリを開きます。**

例えば、カメラ機能を起動する場合は、以下の様な式を定義します。

```
ClientFileOpenDlg (Source,Width',Height,Compression quality',Allow crop)
```



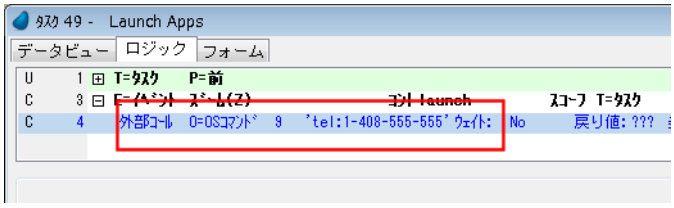
関数からは、イメージのファイル名がフルパスで返されます。イメージが取得できない場合は、空白が返ります。

モバイルアプリから直接電話を掛けるには

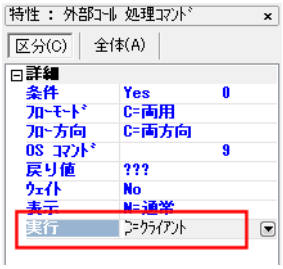
モバイルアプリから電話を掛けることが簡単にできます。

電話を掛けるには：

1. コール OS コマンド処理コマンドを定義し、式エディタを開いて tel と電話番号をコロン（:）で接続した式を設定します。
   例：'tel:1-408-555-555'



2. 外部コール処理コマンドの実行特性を *クライアント* に設定することが重要です。



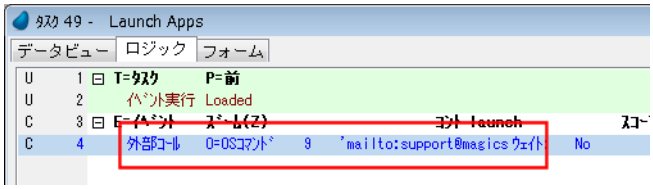
参照： http://developer.apple.com/library/safari/#featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007891-SW1

モバイルアプリから e メールを送るには

モバイルアプリから e メールを送ることが簡単にできます。

e メールを送るには：

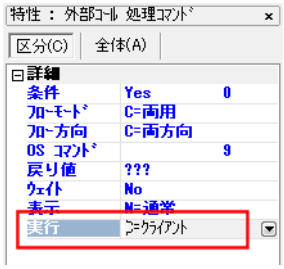
1. コール OS コマンド処理コマンドを定義し、式エディタを開いて mailto とメールアドレスをコロン（:）で接続した式を設定します。
   例：'mailto:support@magicsoftware'



mailto の値に以下のように指定することで、件名や本文を指定することもできます。

'mailto:?to=support@magicsoftware.com&subject=this is a test&body= test mail.'

2. 外部コール処理コマンドの実行特性をクライアントに設定することが重要です。



注： メールを送るとき、件名と本文として送ることができない文字がいくつかあります。

例えば、iOS では、アンパサンド（&) 文字は %26 に置き換えられます。Android の場合は、以下の文字は削除する必要があります。

#,%,&, :, and =.

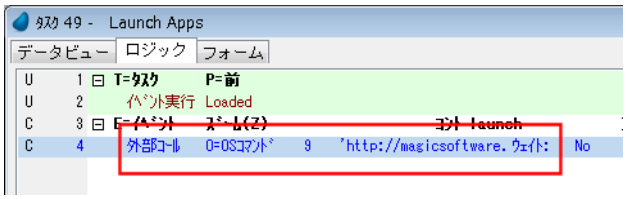
参照： http://developer.apple.com/library/safari/#featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007891-SW1

モバイルアプリから直接ブラウザを開くには

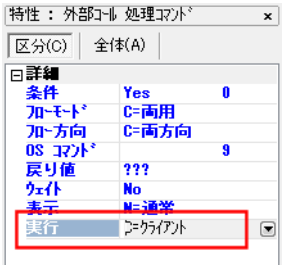
モバイルアプリからブラウザを開くことが簡単にできます。

ブラウザを開くには：

1. **コール OS コマンド**処理コマンドを定義し、**式**エディタを開いて開きたい Web サイトの URL を式として設定します。
   例：'http://magicsoftware.com'



2. **外部コール**処理コマンドの**実行**特性を***クライアント***に設定することが重要です。



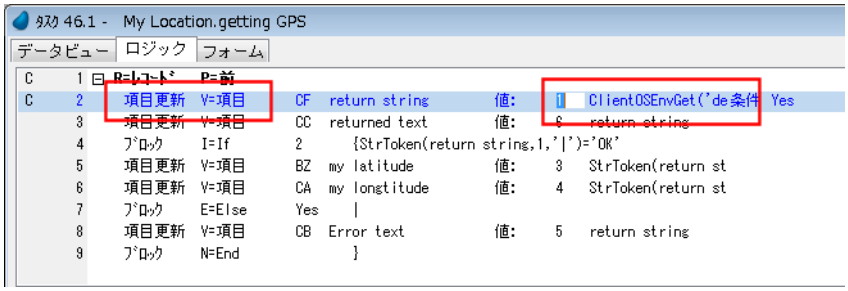
参照： http://developer.apple.com/library/safari/#featuredarticles/iPhoneURLScheme_Reference/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007891-SW1

モバイルデバイスの GPS 機能を使用してデバイスの位置情報を取得するには

ClientOSEnvGet()関数を使用することでGPS機能を使ったモバイルアプリケーションの内でデバイスの位置を取得することができます。内部機能や GPS 機能に接続することで可能になります。

位置情報を取得するには：

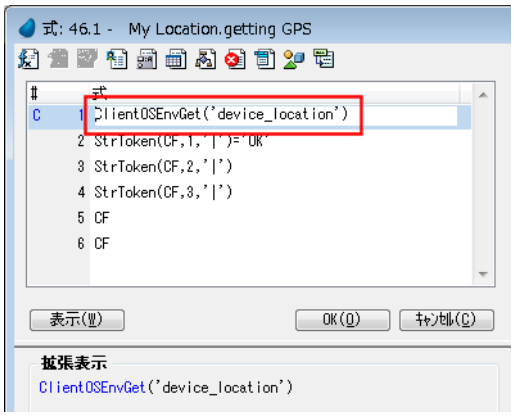
1. **項目更新**処理コマンドを定義します。



2. **項目更新**処理コマンドを定義します。更新値として以下の式を定義します。

   ClientOSEnvGet (*'device_location'*)

   この関数は、、利用可能な機能（GPS、ネットワークなど）を使用して現在のデバイスの位置情報を取得します。



結果は、以下の書式で返ります。

OK|Latitude|Longitude

- OK…… **結果が返った場合の固定値**
- Latitude …… **現在の位置の緯度**
- Longitude…… **現在の位置の経度**

位置情報が取得できなかった場合は、いかなる理由であれ、エラーメッセージが返されます。

Android デバイスでの GPS タイムアウトは 20 秒です。GPS から位置情報を取得する 10 秒と、それが失敗した場合にネットワークから位置情報を取得する 10 秒です。

iOS では、GPS のタイムアウトは 10 秒です。これらのデバイスは、内部の規定に従って GPS またはネットワークを使用します。

タイムアウトを、GPS とネットワークの両方からの位置情報に対して設定することもできます。

以下の構文を使用して行うことができます。

ClientOSEnvGet (*'device_location*|*xxx*|*yyy'*)

- xxx …… **GPS での待ち時間（秒数）**
- yyy …… **ネットワークでの待ち時間（秒数）**

サポートバージョン :3.2

iOS では、GPS のタイムアウト値だけが使用されます。これらのデバイスは内部規定に従って GPS またはネットワークを使用します。

例 : ClientOSEnvGet ('device_location|5|0')

GPS からの応答を 5 秒まで待ち、ネットワークからの位置情報は要求しないことになります。

注： GPS デバイスは衛星を捜すため、位置情報の問合せに時間がかかる場合があります。この間、クライアントの処理が停止し、応答を待ちます。"GPS の検索中 " などのようなメッセージを表示することを推奨します。

Android6 以降からは、アプリが GPS にアクセスする必要がある場合、デバイスが位置情報にアクセスする許可を要求します。（サポートバージョン：3.2）

位置情報の問合わせには、20 秒の組込タイムアウトがあります。

GPS の位置情報を取得した後、例えば、以下のような式を**実行**特性が***クライアント***に設定された**コール OS** 処理コマンドで呼び出すことにより、サードパーティアプリケーションを使用して、この位置の地図を表示させることができます。

```
'https://maps.google.com/?q='&Trim (GPS Location)
```

Google マップアプリがデバイスにインストールされている場合、位置をどのアプリで表示するかを問い合わせるメッセージが表示されます。

```
'geo:'&Trim (GPS Location).
```

これは、地理上のアドレスを表示することができる各アプリが表示されます。

```
'waze://?q=London' ?
```

直接アプリ名を指定して起動することができます。Waze GPS アプリを開き、ロンドンを表示します。

参照： 「他のアプリから Magic xpa のモバイルアプリを起動するには」（842 ページ）

モバイルアプリケーションからサードパーティのアプリケーションにアクセスするには

関連する URL を使用したコール OS コマンド処理コマンドを定義することで、モバイルアプリ内からサードパーティ製のナビゲーションアプリにアクセスすることができます。

例：

- 'waze://?q=New York' …… Waze GPS アプリを開き、New York を表示します。
- 'geo:59.915494,30.409456’ 例えば Google Maps または Waze のようなナビゲーションアプリを開き、緯度：59.915494 経度：30.409456 を表示します。

他のアプリから Magic xpa のモバイルアプリを起動するには

別のアプリや、リンクを使用してた e メールから Magic xpa のモバイルアプリを起動することができます。リンクをクリックしたりタップすると、Magic xpa アプリが開きます。

Magic xpa 3.1 より、以下の URL 書式でアプリが起動されるようにビルド環境が設定されています。

```
<アプリ名>://?URL=<URL アドレス>
例：magicxpa3x://?URL=http://192.168.137.1/MagicRIAApplications/DevProps.txt"> Start My App
```

＜アプリ名＞は、以下の値が使用されます。

- Android …… settings.properties ファイルの client.title の値
- iOS ……PRODUCT_NAME の値

＜URL アドレス＞は、サーバ側のアプリケーションに接続するための実行ファイルの URL です。

この設定を独自に変更する場合は、以下の説明を参考にしてください。

Android デバイスの場合：

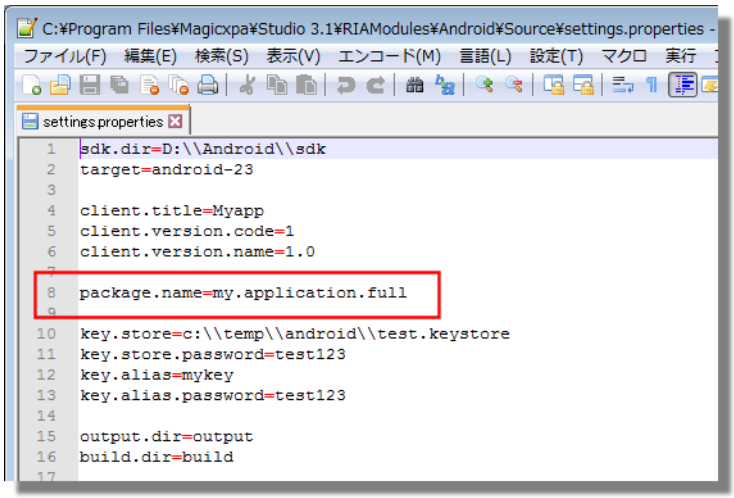
以下のリンクを使用します：

```
'http://com.mycompany.myapp?parameters'
```

com.mycompany.myapp は、アプリケーションをビルドする際に指定するパッケージ名です。

パッケージ名は、リッチクライアントインタフェースビルダで設定したり、RIAModules¥Android¥Source フォルダ内にある settings.properties ファイルの package.name で設定します。

以下の図で表示される情報を使用した場合、Magic xpa アプリを起動するリンクは、http://my.application.full になります。



AndroidManifest.xml ファイルの <intent-filter> セクションの <data android:scheme で始まるタグを以下の赤枠のように修正しすることで変更することができます。

注： **ClientOSEnvGet()** 関数で *device_udf\getargs* を指定することでアプリケーションに渡される照会パラメータ（URL の ? 以降）を取得することができます。

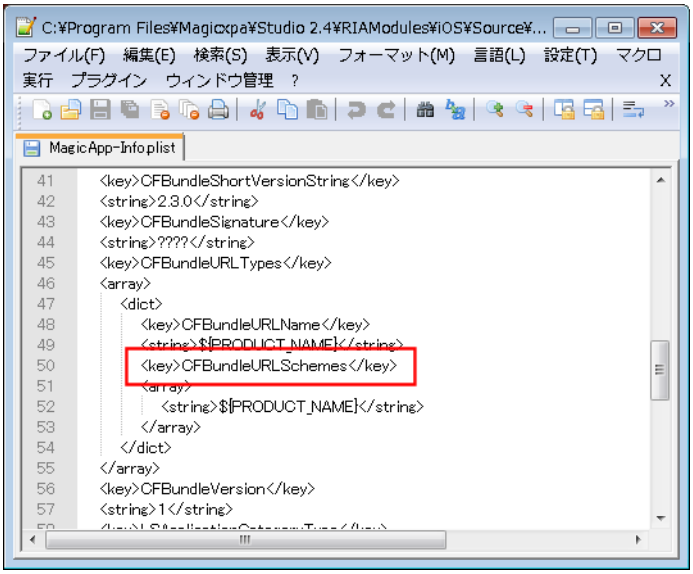
iOS デバイスの場合：

以下のリンクを使用します：

```
'myapp://?parameters'
```

myapp は、アプリケーションに対するユニークな ID です。

デフォルトでは、定義される値は（スペースなしでの) アプリケーション名です。しかし、RIAModules¥iOS¥Source¥MagicApp にある **MagicApp-Info.plist** ファイルの **CFBundleURLSchemes** キー値を設定することで変更することができます。



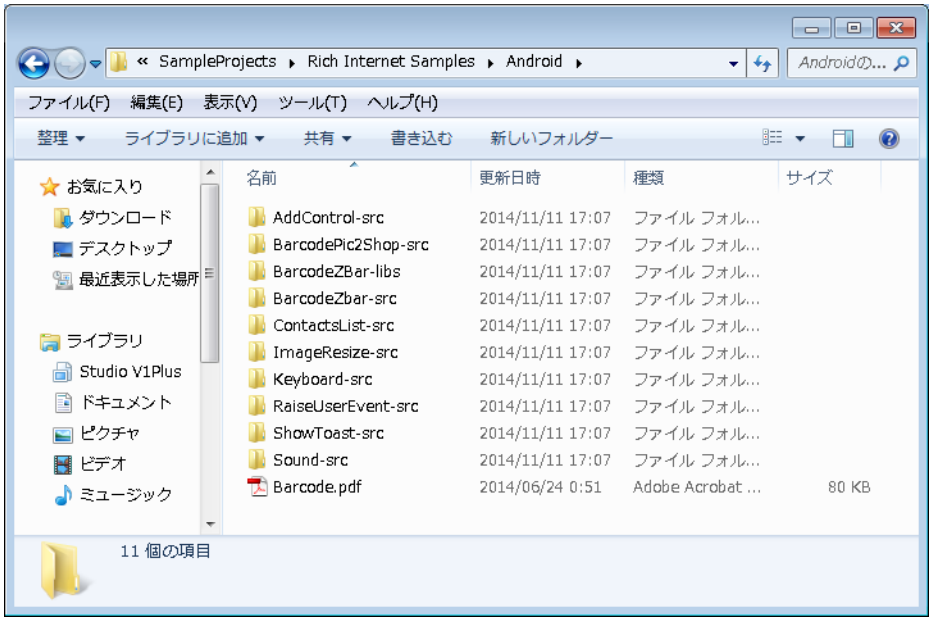
注： **ClientOSEnvGet()** 関数で *device_udf\getargs* を指定することでアプリケーションに渡される照会パラメータ（URL の ? 以降）を取得することができます。

モバイルアプリに定義されたネイティブなコードを利用するには

モバイルアプリに定義されたネイティブなコードを使用する必要があるかもしれません。たとえば、カスタムメイドのハードウェアを使用するときなどです。

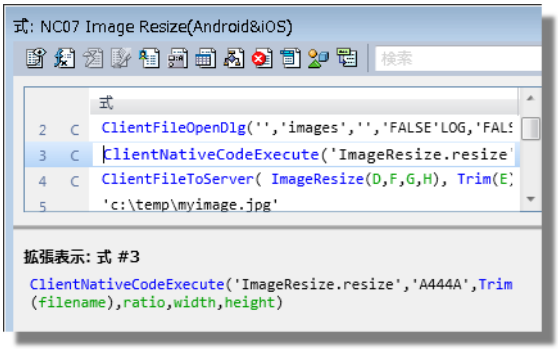
Magic xpa Studio をインストールすると、ネイティブな OS コードを使用するサンプルもコピーされます。以下の図は、サンプルがコピーされている Android 用のフォルダを示しています。同じようなフォルダは、iOS 用にもあります。

各サンプルには、指定されたネイティブコードを使用するための手順を説明するヘルプファイルが同梱されています。



2 つ Magic xpa のアプリからネイティブコードを呼び出すことで、モバイルアプリ内のネイティブコードを実行させることができます。これは、ネイティブコードのメソッドとパラメータ名を指定した **ClientNativeCodeExecute()** 関数を実行することで実現できます。

例えば下の図では、**ClientNativeCodeExecute()** 関数で、モバイルアプリケーション内の ImageResize クラスの resize クラスを実行しています。パラメータとしてファイル名やサイズの比率、幅と高さを渡しています。



iOS の場合は、メソッドに複数の異なるパラメータを送りますが、NSArray タイプの 1 つのパラメータだけで呼び出されるメソッドを定義する必要があります。Magic パラメータは全て、自動的にこの配列に設定します。そして、配列からパラメータを読み出すことができます。

この関数に対するネイティブコードは、以下のようになります。

Andrtoid

Rich Internet Samples¥Android¥ImageResize-src フォルダ内の ImageResize.java ファイル

```
package com.magicsoftware.magicdev;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import java.io.ByteArrayOutputStream;
import java.io.FileOutputStream;
import java.io.File;
import java.io.IOException;
import android.content.Context;
import com.magicsoftware.core.CoreApplication;
public class ImageResize {
    public static String resize(String imagePath,int compressionQuality,int Width,int Height) {
        // syntax: resize:ImagePath,compressionQuality,Width,Height
```

iOS

Rich Internet Samples¥iOS¥ImageResize-MagicApp フォルダ内の ImageResize.m ファイル

```
#import "ImageResize.h"

@interface ImageResize ()

@end

@implementation ImageResize

+(NSString *)resize:(NSArray *)params
{
    // syntax: resize:ImagePath,compressionQuality,Width,Height
    // returns: the new resized image file path
    // in compressionQuality - 0 being the maximum compression, 100 being the best quality
    // in Width and Height, value of 0 means keeping the original size

    NSString *imagePath = [params objectAtIndex: 0];
    CGFloat compressionQuality = [[params objectAtIndex: 1] floatValue];
    float Width = [[params objectAtIndex: 2] floatValue];
    float Height = [[params objectAtIndex: 3] floatValue];
```

注： Android デバイス上で、MainApplication.java ファイルで Unicode 文字を使用する必要がある場合、ファイルを BOM エンコーディングのない UTF-8 として保存しなければなりません。

参照： 第 41 章：「モバイルアプリ内で実行しているネイティブコードからの Magic xpa イベントを発行するには」（846 ページ）

モバイルアプリ内で実行しているネイティブコードからの Magic xpa イベントを発行するには

ネイティブ・コードから Magic xpa アプリケーションで処理される Magic xpa イベントを発行することができます。以下の行をネイティブコードに追加することで実現できます。

Android

1. 宣言を追加します：import com.magicsoftware.core.CoreApplication;
2. イベントを発行します：CoreApplication.getInstance(). invokeUserEvent(event_name,param1,param2);
   - event_name …… **ユーザイベント名**
   - param1 と param2 …… **［イベント］ロジックユニットに渡されるパラメータ**

iOS

1. 宣言を追加します：#import "Magicxpa.h"
2. nil で終わるパラメータ値の全てを保持する配列を追加します：NSArray *params = [NSArray arrayWithObjects:param1, param2, nil];
   - param1 と param2 …… **［イベント］ロジックユニットに渡されるパラメータ**
3. イベントを発行します：[Magicxpa invokeUserEvent:event_name Params:params];
   - event_name …… **ユーザイベント名**

ユーザに、アクセス許可を要求する (Android)

デバイスのカメラを開いたりカレンダ入力を行うようなネイティブコードを実行する場合、ユーザに特定のアクセス許可を要求する必定があります。

以下の例のような行をネイティブコードに追加することで、ユーザに許可を依頼することができます。

boolean requestResult = CoreApplication.getInstance().requestPermission(Manifest.permission.RECORD_AUDIO);

ここで、ユーザは、RECORD_AUDIO のアクセス許可を承認するように要求され、結果は requestResult 変数に格納されます。

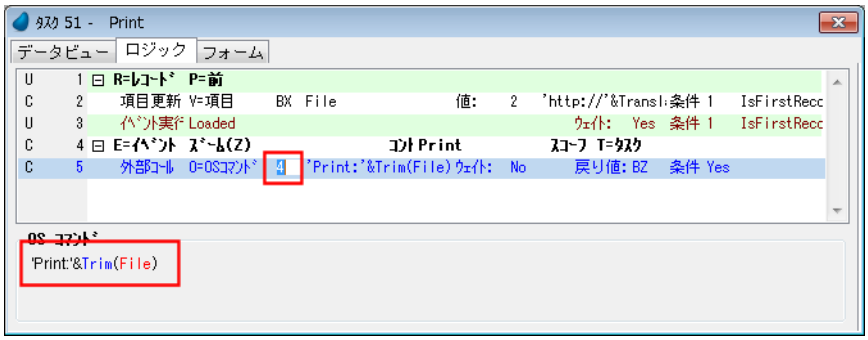
サポートバージョン :3.2

参照 : 第 41 章 :「モバイルアプリに定義されたネイティブなコードを利用するには」（844 ページ）

モバイルアプリケーションから PDF を印刷するには

iOS デバイスでは、以下のようにすることで PDF を印刷することができます。

1. コール OS コマンド処理コマンドを定義します。
2. 実行特性を*クライアント*に設定します。
3. 実行するコマンドとして以下の式を定義します。Print: ファイル名
   ファイル名には以下のどちらかを指定します。
   - パス付きのローカルファイル名（**ServerFileToClient()** 関数の戻り値を利用できます）
   - Web サイトの URL（例 :'http：//1.1.1.1/myfile.pdf'）



印刷は、iOS の Air Print プロトコルを使用して行われます。

サポートバージョン :2.3

Android デバイスでは、以下のようにすることで PDF を印刷することができます。

1. コール OS コマンド処理コマンドを定義します。
2. 実行特性を*クライアント*に設定します。
3. 実行するコマンドとして以下の式を定義します。Print: ファイル名
   ファイル名にはパス付きのローカルファイル名（**ServerFileToClient()** 関数の戻り値を利用できます）を指定します。

印刷は、Google Cloud Print プロトコルを使用して行われます。

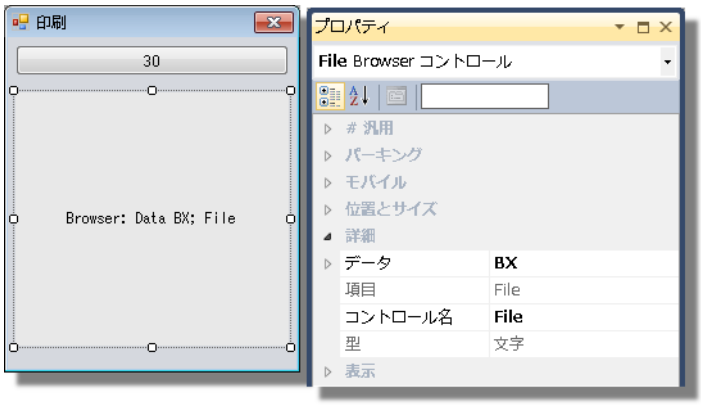
サポートバージョン :3.0

モバイルアプリケーションから PDF を表示するには

iOS

Magic xpa のブラウザコントロールを使用して PDF を表示することができます。

PDF がサーバ上にあり、ローカルデバイスにはないことに注意してください。



Android

ほとんどの Android 用 Web ブラウザは、PDF を表示しません。このため、Magic xpa のブラウザコントロールを使用して PDF を表示させることができません。

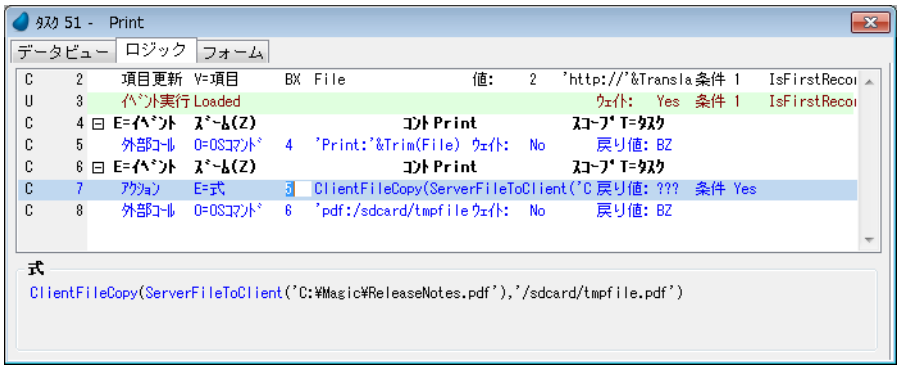
しかし、サードパーティ製の PDF ビューア・アプリを使用することで PDF を表示することができます。これには、以下の通りにしてください。

1. コール OS コマンド処理コマンドを定義します。
2. 実行特性を *クライアント* に設定します。
3. 実行するコマンドとして以下の式を定義します。pdf: ファイル名

ファイル名にはデバイスの公開フォルダ（/SDCARD フォルダ）に保存されたローカルファイルをパス付きで指定します。

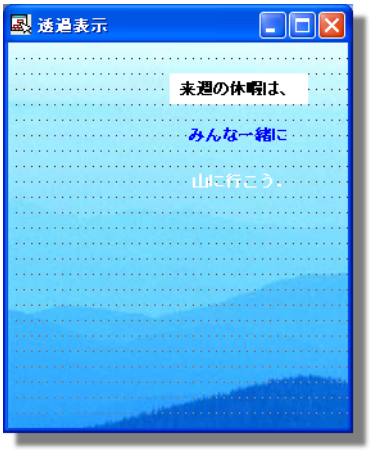
例：以下のコマンドによって、ReleaseNotes.pdf ファイルがサーバ上の C\Magic フォルダから /SDCAD フォルダにコピーされ、デバイス側で表示されます。

アクション処理コマンド	ClientFileCopy(ServerFileToClient('C:¥Magic¥ReleaseNotes.pdf'),'/sdcard/tmpfile.pdf')
コール OS コマンド処理コマンド	'pdf:/sdcard/tmpfile.pdf'



.

コントロールを透過表示にするには

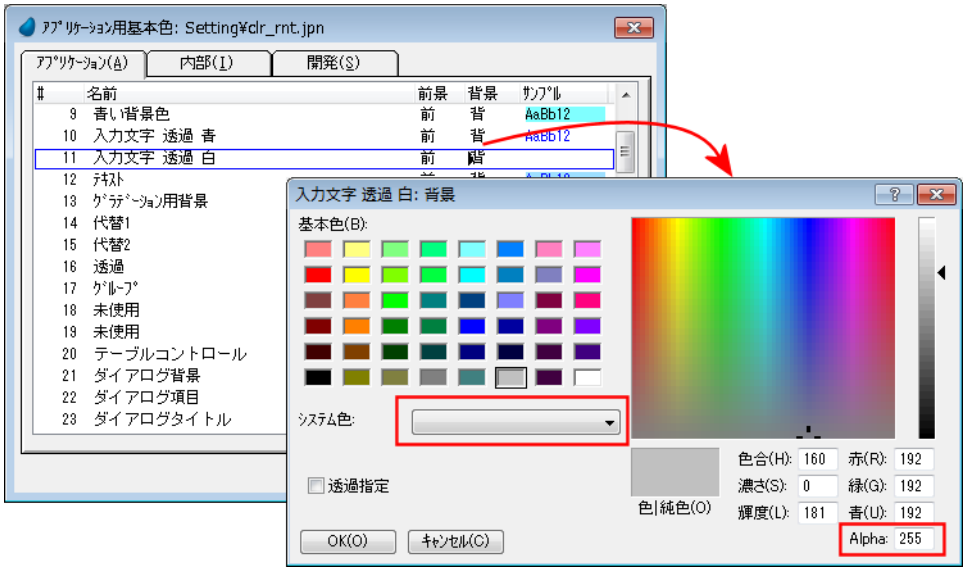


通常、コントロールの背景色は不透明か、Windows から継承された立体色です。例のように背景色を透過にすることで背景の画像を表示させたい場合があります。

最初のラベルコントロール「来週の休暇は、」は、白い背景色を持つ平面のラベルコントロールです。次の２つのコントロールは両方とも透過の背景色になっていて、前景色が異なっています。

このような表示にする方法は、基本的に、透過の背景色を持つ色をコントロールに定義することです。以下にどのように行うかを説明します。

透過色の設定



1. **オプション→設定→基本色** を選択します。
2. 変更したい色を クリック するか、***F4*** を押下して色を追加します。
3. **背景色**カラムで **ズーム** します。
4. **システム色**を 空白 に設定します。ドロップダウンリストの先頭で空白行を選択することで設定できます。.
5. **Alpha** 欄で、透過レベルを指定できます。値は、***0 ～ 255*** まで指定できます。「0」は完全に透過（[透過] チェックボックスをオンにした場合と同じ状態）になります。
6. OK を クリック します。
7. 基本色テーブルに戻り、([透過] 設定を除いた）同じ方法で前景色も設定できます。
8. OK を クリック して色指定のダイアログボックスを終了します。

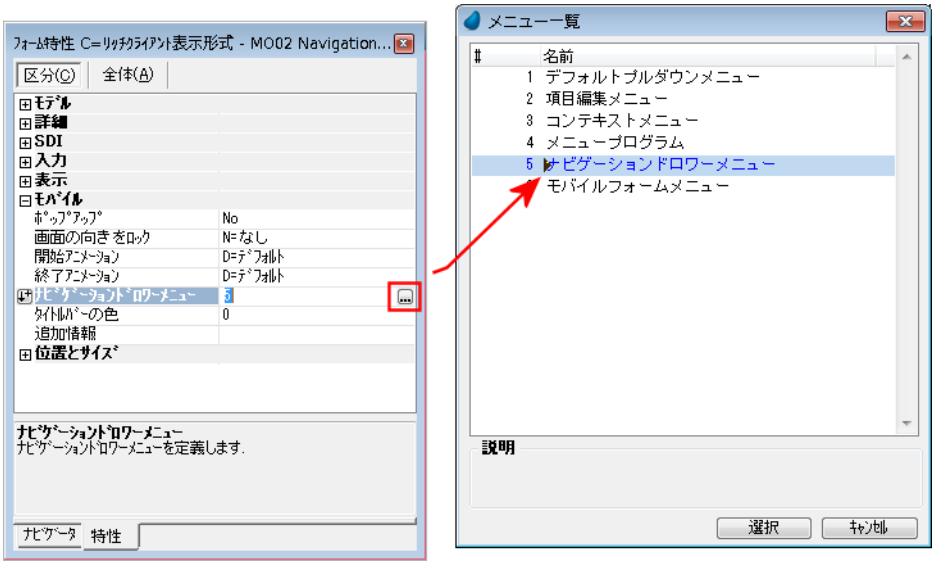
サポートバージョン :2.5

注： 第 5 章 :「コントロールを透過表示するには」（130 ページ）

モバイル上でナビゲーションドローワを使用するには

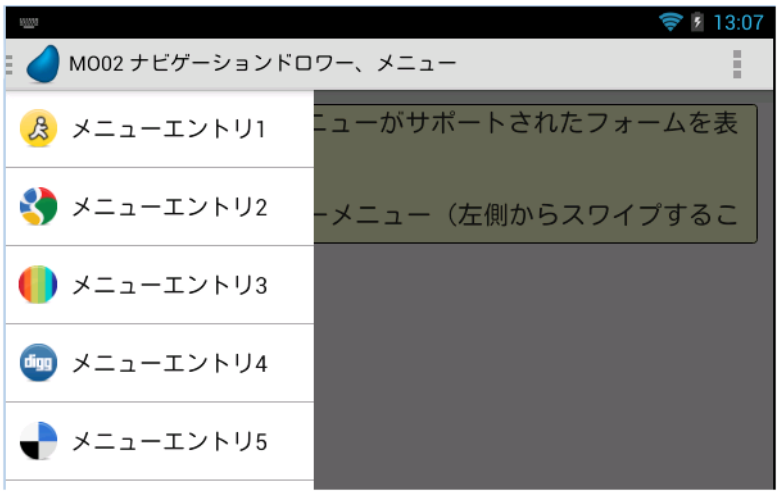
モバイルデバイスでは、ナビゲーションドロワーとして表示されルメニューを定義することができます。これは Google Play のように、左側に表示されるメニューです。メニューは、画面の左側からスワイプすることによってエンドユーザが開くことができます。

フォームの**ナビゲーションドロワー・メニュー**特性を設定することで定義します。数値を入力するか、ズームして**メニュー**リポジトリから選択することができます。この例では、ナビゲーションドロワー・メニューとして使用するために、メニューを定義しています。



注： Android デバイスでは、RIAModules¥Android¥Source¥MgxpaRC¥res¥layout フォルダに配置された navigation_drawer.xml ファイルを変更することで、ナビゲーションドロワーの外観をカスタマイズすることができます。

モバイルデバイス上でアプリケーションを実行して、画面の左側からスワイプすると、以下のように表示されます。

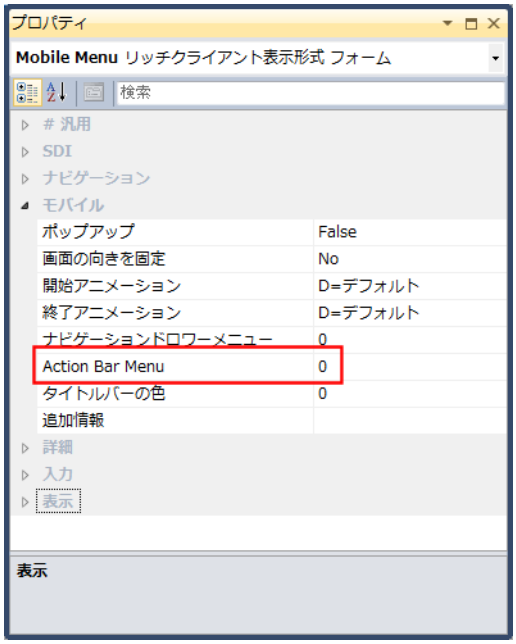


サポートバージョン：2.5

モバイルデバイスのタイトル上にアイコンやテキストを表示せるには

モバイルデバイスのでは、メニューをナビゲーションバーに追加することでアイコンやテキストをタイトルエリアに追加することができます。

フォーム特性の**アクションバーメニュー**特性を設定することで定義することができます。



アイコンをタイトルエリアに追加するには、フォーム特性の**アクションバーメニュー**特性で、アイコンを含むメニューエントリを選択します。イメージはフォーム特性の**ボタンイメージ**特性に定義されたファイルが表示されます。

テキストをタイトルエリアに追加するには、アイコンを含まないメニューエントリを選択します。これで、メニュに定義されたテキストが表示されます。テキストは、メニューの**ユーザメニュー名**に定義された内容が表示されます。

アクションバーメニューに、表示可能な件数を超えたメニューが定義されている場合、Android の場合は表示されないアイコンがフォームのコンテキストメニューに追加されます。iOS の場合は、３つだけが表示されます。

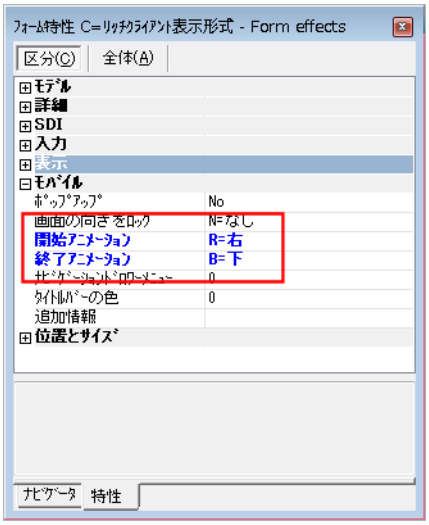
注： アクションバーは、ポップアップウィンドウではサポートされません。

サポートバージョン :3.1

モバイル上でフォームの開始 / 終了のアニメーションを定義するには

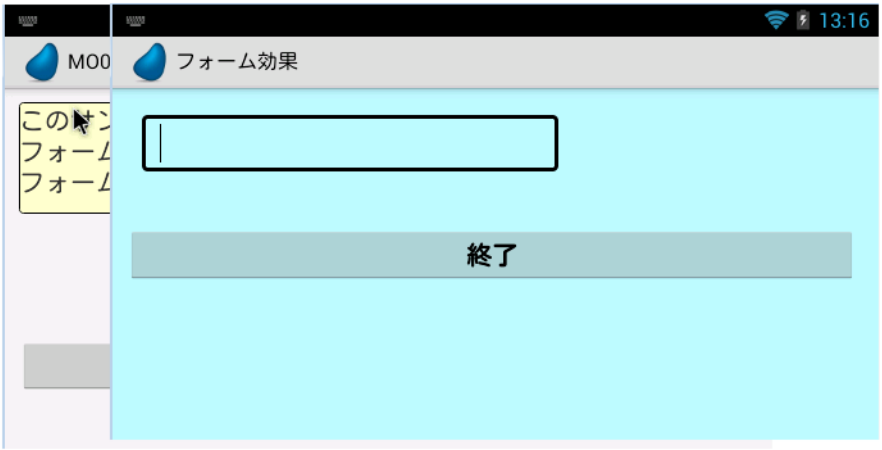
モバイルデバイスでは、フォーム表示を開始したり終了したりする際に表示するアニメーションを定義することができます。開始／終了フォームとして画面を切り替える方向を、左、右、上または下のどれかで指定できます。

フォームの**開始アニメーション**と**終了アニメーション**の各特性を使用して行います。



注： iOS デバイスの場合、開始アニメーションと終了アニメーションは、ポップアップフォーム上ではサポートされません。

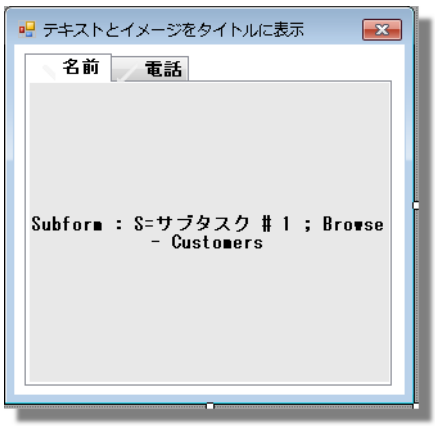
モバイルデバイス上でアプリケーションを実行して、ボタンをクリックすると、以下に表示されているのように、画面の右側から呼び出されるタスクのフォームが表示されます。



サポートバージョン :2.5

モバイル上でタブコントロールを使用するには

Magic xpa 2.5 から、［タブ］コントロールがモバイル上でタブバーのように表示されます。

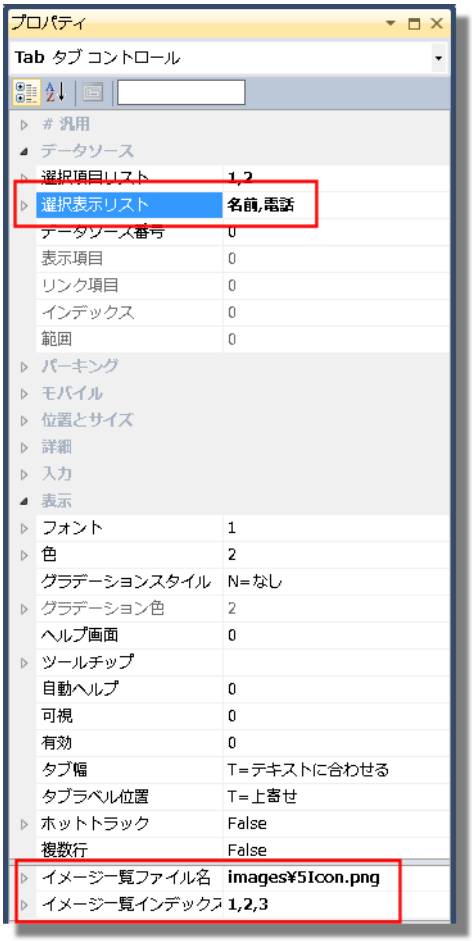


タブバーはフル画面で表示される為、モバイル用にアプリを開発しているとき、フォーム全体に渡って［タブ］コントロールを広げる必要があります。

一般に、タブバーは、デバイスのデフォルト定義による機能を使用します。この表は、各デバイスに対する機能に関する詳細です。

機能	Android	iOS
ナビゲーション	タブバーの項目をクリックするか、タブバーをスワイプします。	タブバーの項目をクリックする。
位置付	フォームの最上位	フォームの底辺
タブ内の水平スクロール	サポートされません。 水平スクロールは、タブ間での操縦になります。画面サイズを超える（タブ上の) コントロールは表示されません。	サポートされます。
タブ内の垂直スクロール	サポートされます。	サポートされます。

イメージ



タブにテキストとイメージを設定することができます。Windows OS のタブのように、イメージは［イメージリストファイル名］と［イメージリストインデックス］特性で定義されます。しかし、いくつかの違いがあります。

Windows OS のタブとの違いは以下の通りです。

•イメージのサイズは、16x16 に制限されていません。しかし、四角形でなければなりません。このため、例えば、イメージリストに複数の 100x100 のイメージを含めることができます。
•イメージファイルは、タブに表示するすべてのイメージを含めなければなりません。

テキストとイメージを持つタブを準備した場合、タブは以下のように表示されます。

Android のみ

Android デバイスの場合、空白のテキストとイメージを指定しないようにすることで、タブヘッダを隠すことができます。この場合、タブはヘッダがなくオブジェクト間でスクロールするために使用されます。［選択表示リスト］特性と［イメージリストファイル名］特性を設定することでテキストとイメージ表示したり隠したりすることができます。

テキストとイメージの異なる組合せを表示するために設定する特性については、以下の表を参照してください。

表示の組合せ	表示項目リスト	イメージリストファイル名
テキストのみを表示	✔	X
テキストとイメージを表示	✔	✔
イメージのみを表示	X	✔
タブバーを隠します (そして、スクロール動作のペインだけがあります)	X	X

テキストだけを表示するためにタブを準備した場合、タブは以下のように表示されます。

例外

デバイスのデフォルト動作のため、以下のような設定はサポートされません。

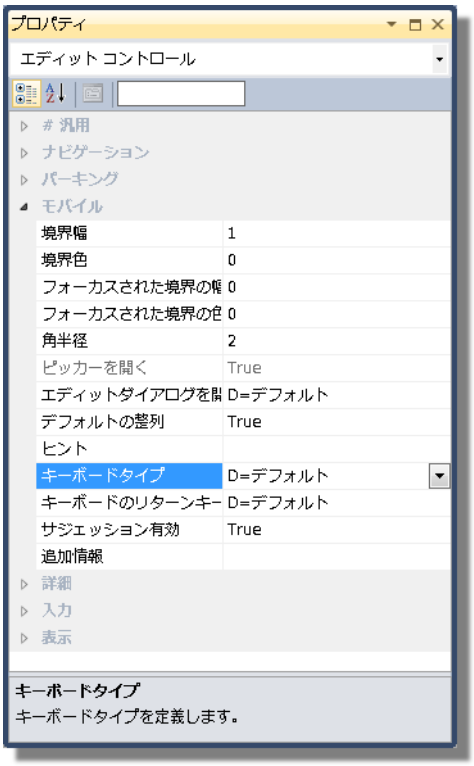
- ［タブ］コントロールの外側に配置されるコントロール
- ［タブ］コントロールのレイヤ 0 にリンクされたコントロール
- フォーム上の複数の［タブ］コントロール
- コンテナコントロール（例えば、［グループ］コントロールや別の［タブ］コントロール) にリンクされた［タブ］コントロール
- ［タブ］コントロールを持つサブフォームタスク

注： **フォームの背景（壁紙や色、グラデーション) は、タブの全で使用されます。**

サポートバージョン :2.5

モバイル上でキーボードをカスタマイズするには

Magic xpa では、モバイル用にキーボードの配置をカスタマイズすることができます。キーボードコンテンツ（リターンキー）を定義することができ、オートコレクトのサジェスチョン機能を有効にするか無効にするかを指定できます。［エディット］コントロールのいくつかの特性でこの設定を行います。



注： ［サジェッション有効］特性は、文字型と Unicode 型のフィールドでのみサポートされます。

キーボードタイプ

いろいろなキーボードタイプを選択することができます。これは、エンドユーザーがどのタイプの情報を入力することができるかを指定できます。以下のオプションから選択できます。

- Default …… デバイスのデフォルト・キーボードが使用されます。
- Numeric…… ユーザは、数値と句読点を入力することができます。
- URL …… ユーザは、テキストと URI を入力することができます。
- Number Pad …… ユーザは、数値テキストと電話番号を入力することができます。
- Phone Pad …… ユーザは、電話番号を入力することができます。
- Name Phone Pad …… ユーザは、数値テキストを入力することができます。
- Email …… ユーザは、数値テキストと電子メールのアドレスを入力することができます。

注： キーボードレイアウトは、使用している仮想キーボードに依存します。

キーボードのリターンキー

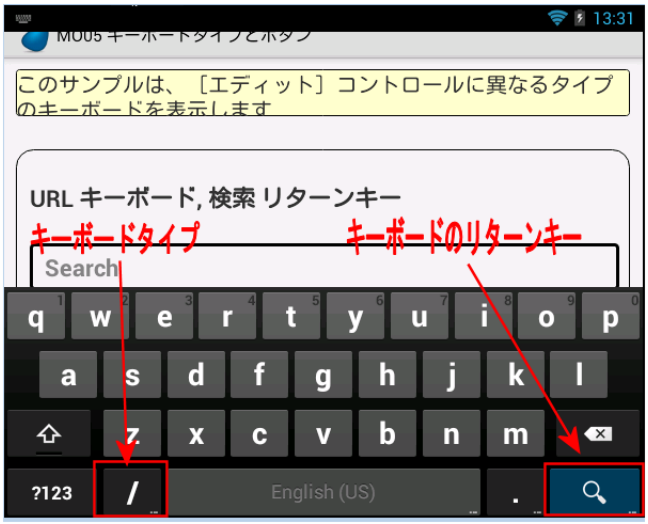
リターンキー上に表示されるイメージ / テキストを定義することもできます。選択されたオプションに従って内部のイベントが発行されます。これらは、オプションです。

- Default …… デフォルト OS のリターンキー値（たとえば、別のコントロールがある場合は、［次へ］。最後のコントロール上の場合は［終了］）。
- Next …… ［次項目］イベントが発行されます。
- Previous …… ［前項目］イベントが発行されます。
- Go …… キーボードを閉じて、［選択］イベントが発行されます。
- Search …… キーボードを閉じて、［選択］イベントが発行されます。

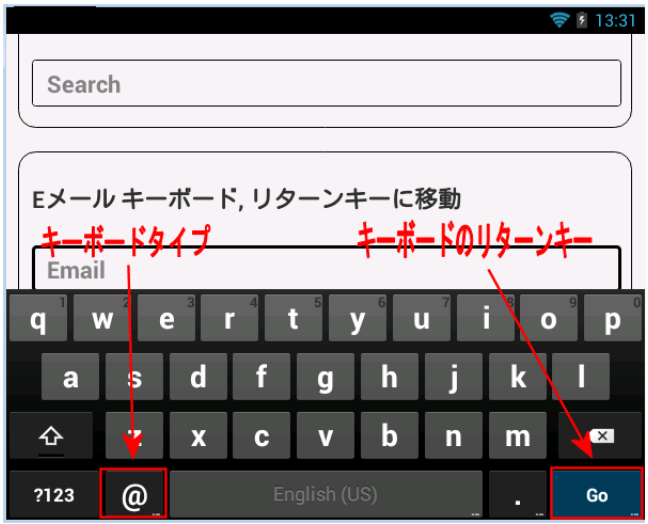
- Done …… キーボードを閉じて、［選択］イベントが発行されます。

注： 表示されるイメージまたはテキストは、使用している仮想キーボードに依存します。
Go、Serach そして、Done はすべて同じイベントが発行されます。しかし、ボタン上に表示されるテキストは異なります。

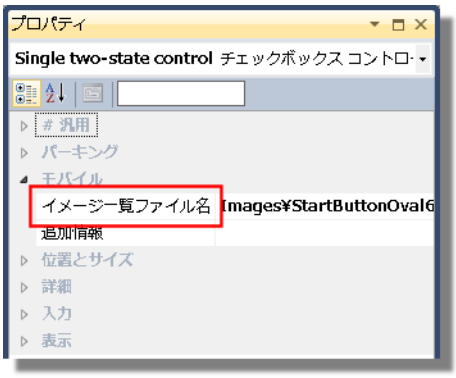
キーボードタイプ = URL
キーボードのリターンキー = 検索



キーボードタイプ = Email
キーボードリターンキー = 移動



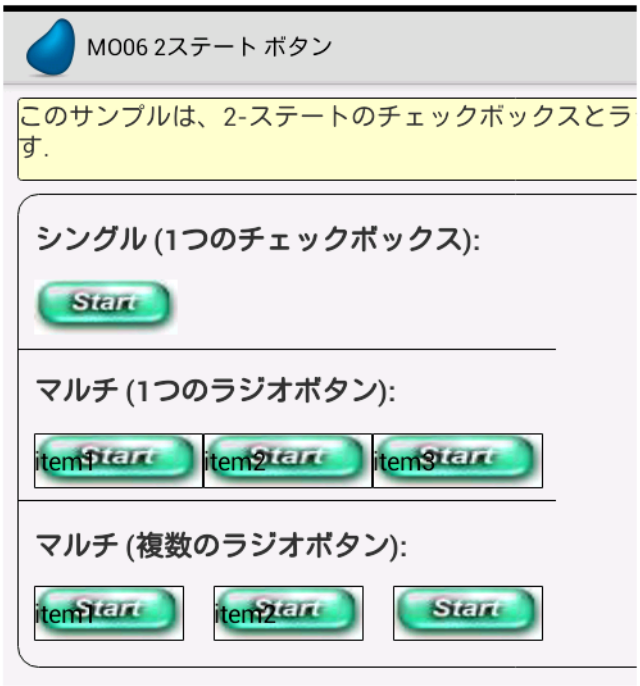
2 ステートのイメージボタンを定義するには



モバイルデバイスでは、2- ステートのイメージを作成することができます。これは、モバイルデバイス上でイメージをクリックすることにで、イメージが代替イメージに変わることを意味しています。エンドユーザーがこれをクリックするたびに、2 つのイメージを切り替え（トグル）ます。**チェックボックス**コントロールの**モバイル**セクションで**イメージリストファイル名**特性を設定することで可能になります。

ラジオボタンコントロールの**モバイル**セクションで**イメージリストファイル名**特性を設定することにで、複数の 2- ステートのイメージを作成することもできます。

モバイルデバイス上でアプリケーションを実行すると、以下のように表示されます。たとえば、シングル（1 つの**チェックボックス**コントロール）セクションでは、イメージは明るい緑色で表示されます。エンドユーザーがそれをクリックすると、異なるイメージ（サンプルプロジェクトではより暗い緑の［Start］ボタン) に変わります。



サポートバージョン :2.5

iOS アプリをビルドするためのプロビジョニングファイルを作成するには

条件

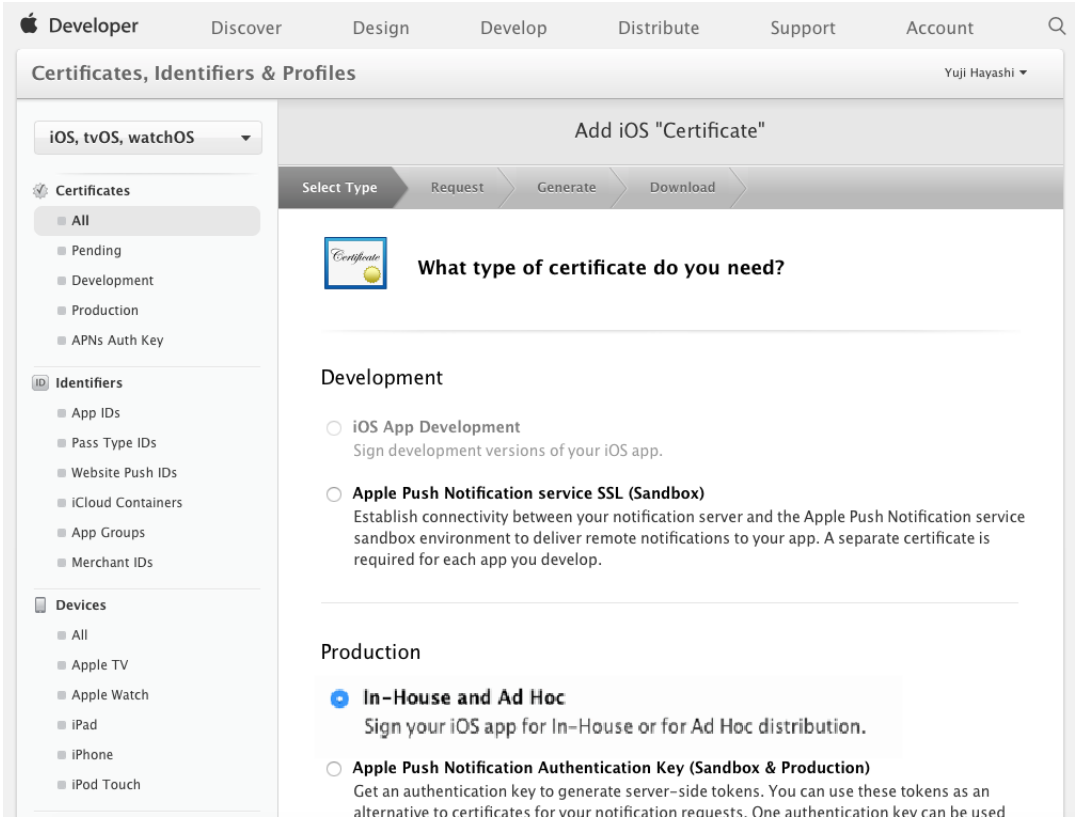
Magic xpa 用の iOS 版モバイルクライアントの開発環境をセットアップするには、以下が必要です。

- Xcode がインストールされた Mac マシン
- Apple Developer か、Enterprise Developer のアカウント。

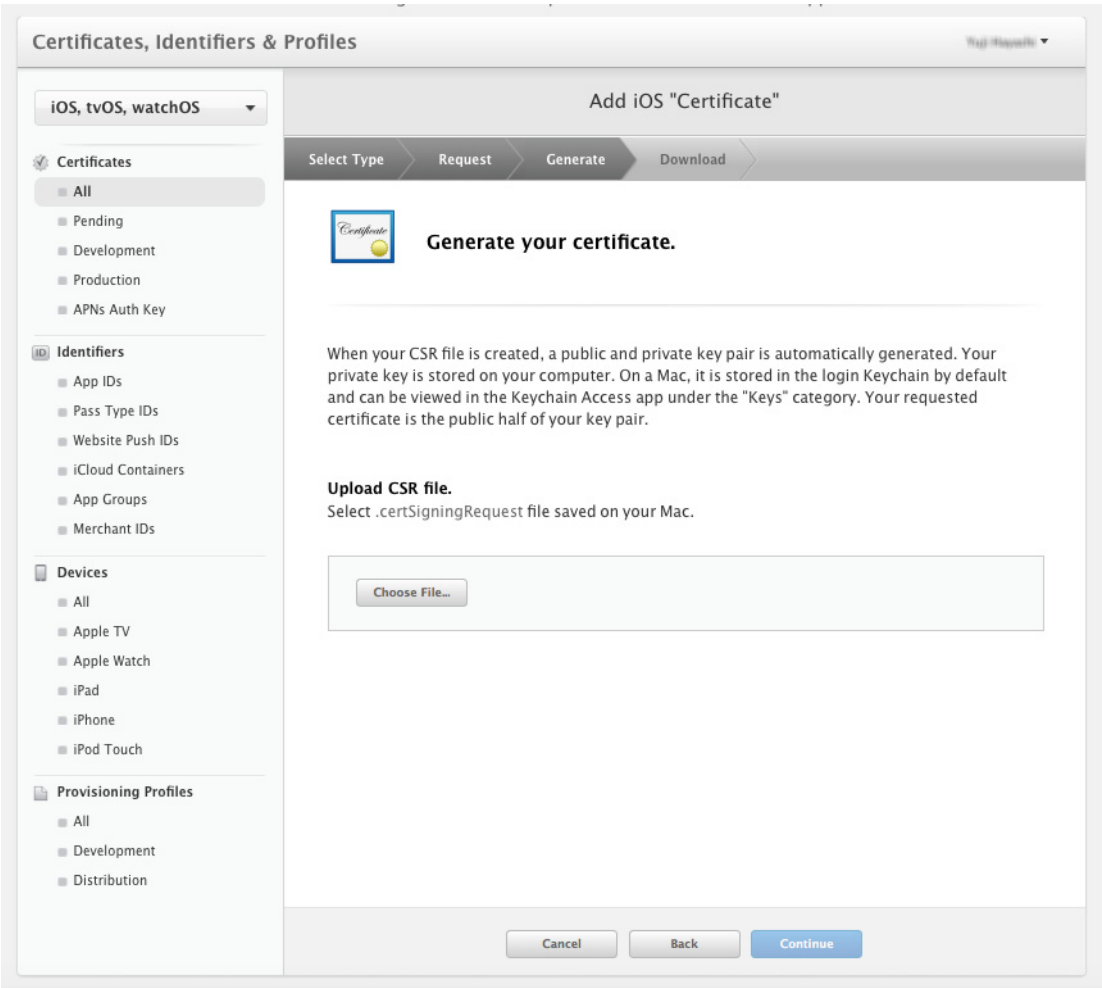
証明書

最初のステップは、iOS 開発環境に証明書を設定することです。

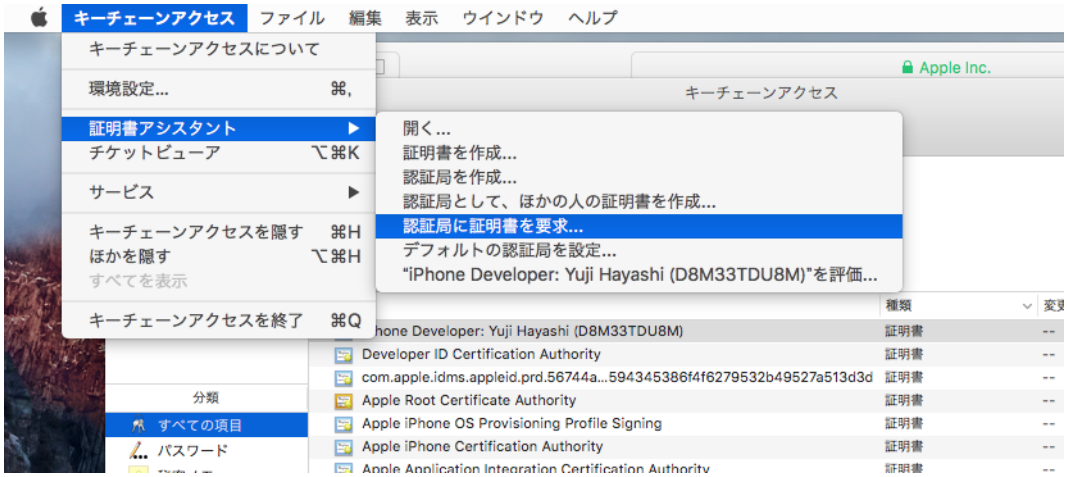
1. Apple Developer のアカウントで http://developer.apple.com にログインしてください。
2. iOS,tvOS,watchOS の下の ***Certificates*** セクションの ***All*** をクリックしてください。
3. **Add iOS Certificate** 画面の Production エリアで、必要な証明書のタイプを選択してください。



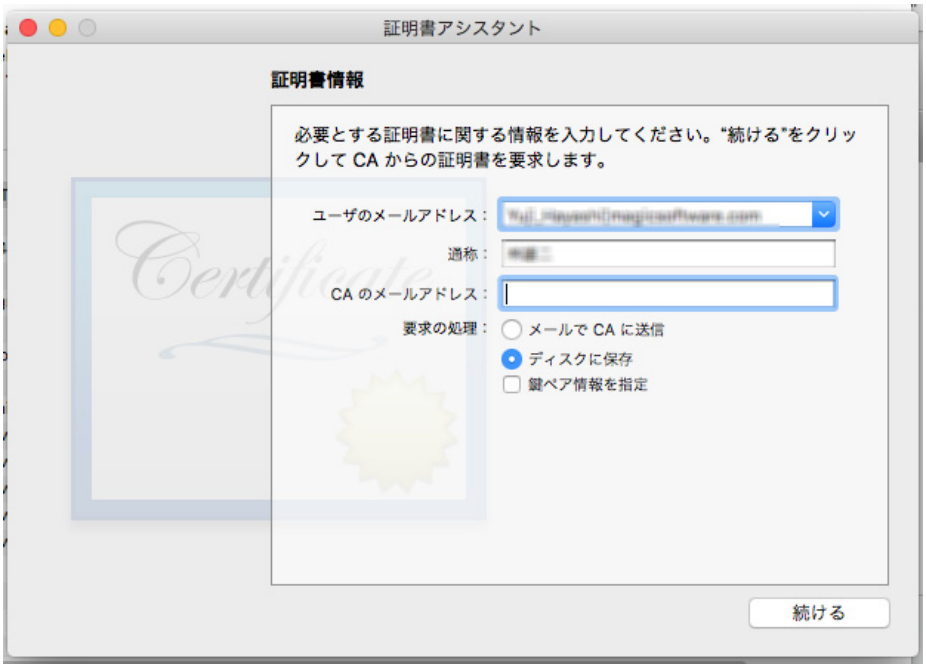
4. Continue をクリックしてください。
5. 次に、証明書リクエストを作成するように要求されます。Continue をクリックしてください。
6. 次の画面で、certSigningRequest ファイルを作成するように要求されます。このファイルは、Mac マシン上で生成され、Developer アカウントで Mac マシンを接続する処理のこのステップの間に更新されます。



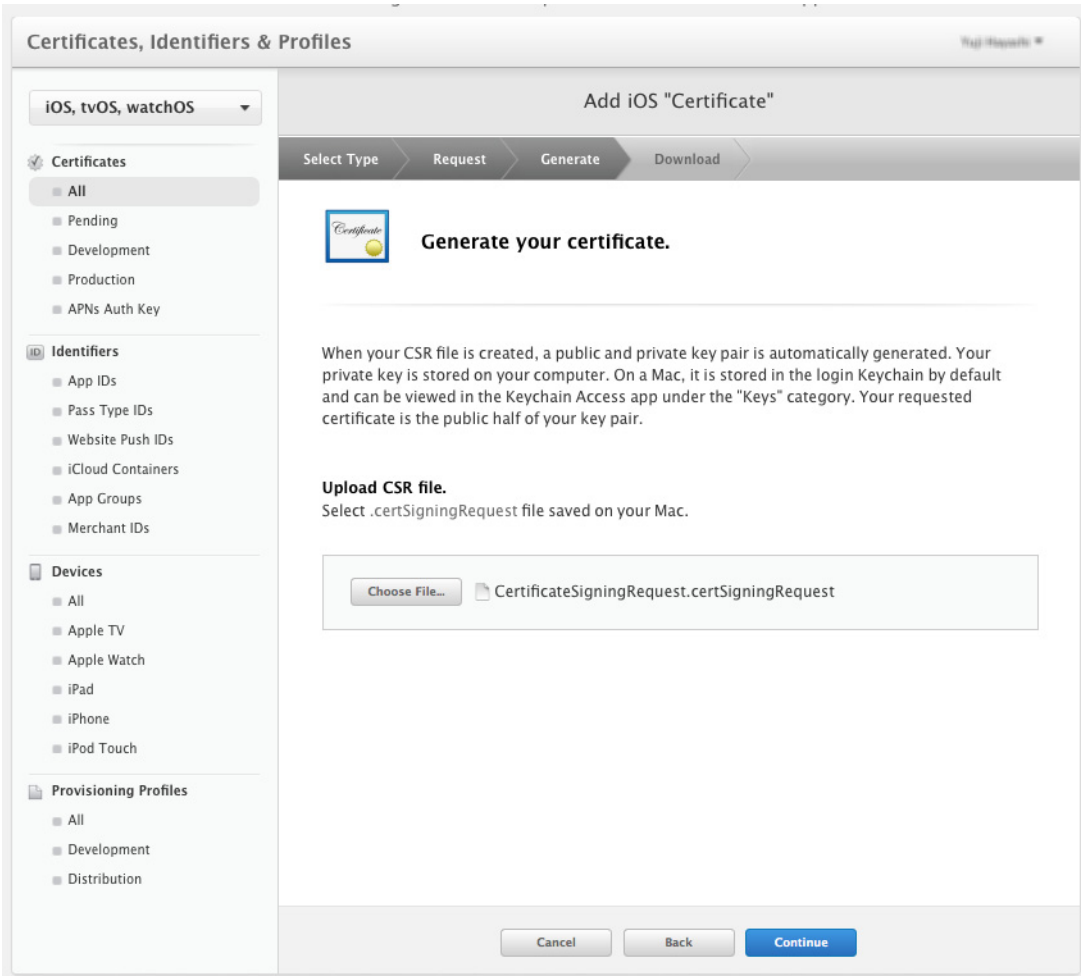
7. .certSigningRequest ファイルを作成するには、Mac マシン上で**キーチェーンアクセス**アプリケーションを開いてください。**キーチェーンアクセス**のプルダウンメニューをクリックし、**証明書アシスタント**を選択します、次に**認証局に証明書を要求…**を選択してしてください。



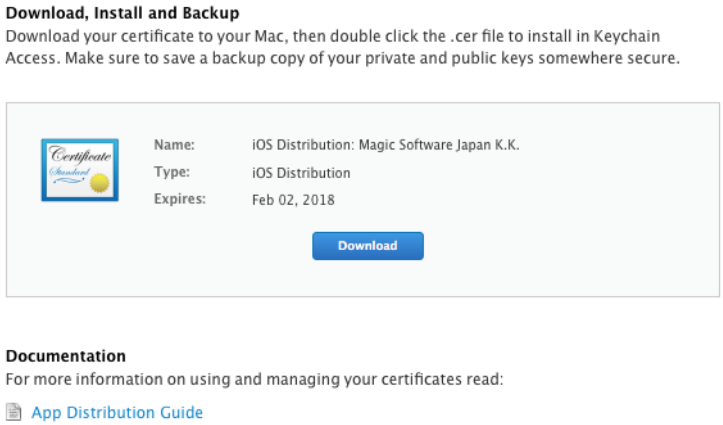
8. 次の画面上で、Apple Developer アカウントとして登録した E メールアドレスを入力し、***ディスクに保存***を選択してください。



9. 続けるをクリックしてファイルをローカルディスクに保存します。これが .certSigningRequest ファイルになります。



10. 次に、ポータルから証明書をダウンロードしてください。



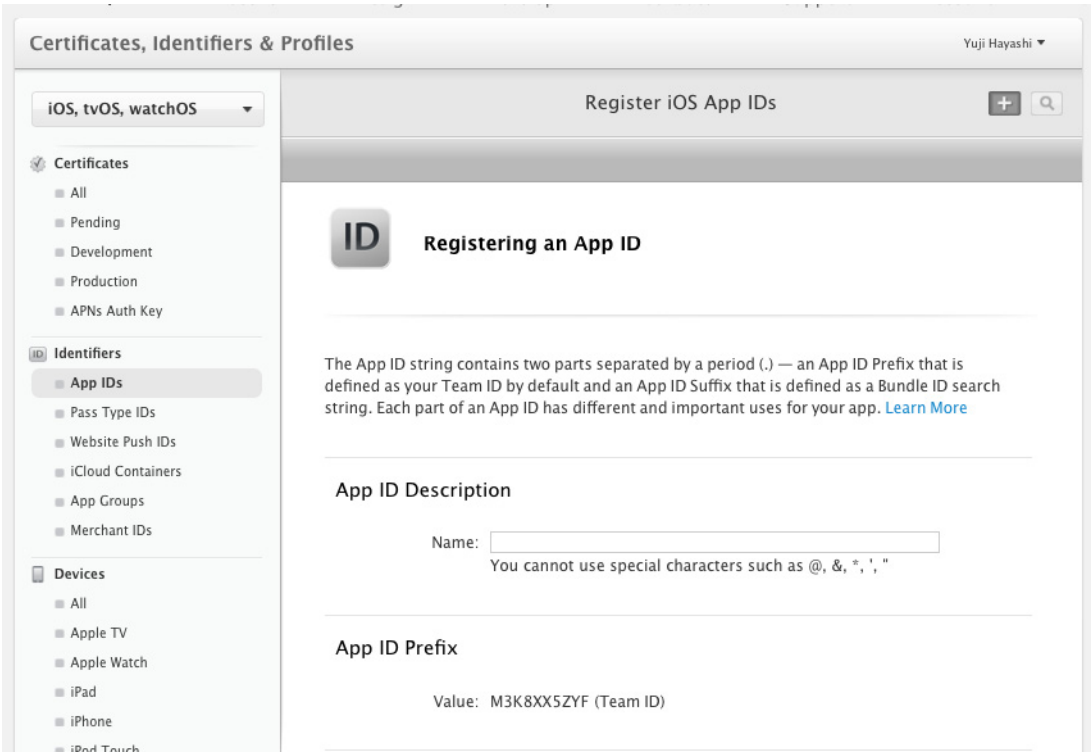
11. 証明書をインストールするために、ダウンロードされた .cer ファイルをダブルクリックしてください。キーチェーンアクセスユーティリティで証明書を確認することができます。

App ID

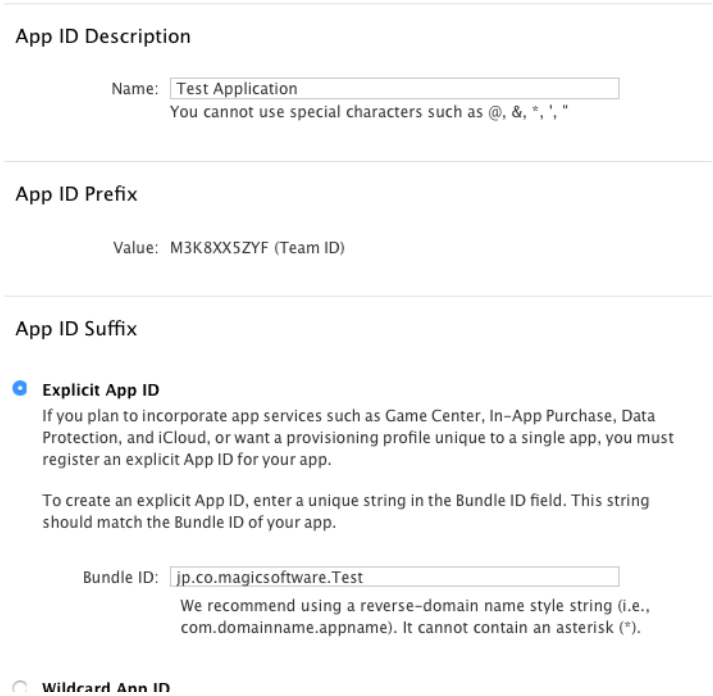
証明書の処理が完了すると、Mac の開発環境は Apple Developer アカウントによって保証され、そして結果として、Apple Developer ポータルによって保証されます。

次に、Apple Developer アカウントで作成するアプリケーションを定義する必要があります。

1. iOS,tvOS,watchOS の下の ***Identifiers*** セクション内で ***App IDs*** をクリックしてください。



2. 次の画面で、以下のように定義してください。
   - App ID Description …… これは Apple Developer ポータルの説明目的のためです。
   - App ID Prefix …… これは、Xcode プロジェクトと Apple Developer アカウント内でのアプリケーション定義をリンクするインデックスです。すべての iOS アプリにおいてユニークにする必要があります。推奨される設定は、com.domainname.appname スタイルの次に、バンドル ID を置く方法です。
   - App Service …… これは、利用可能な Apple のサービスの中からプロジェクトが使用するものを選択することができます。



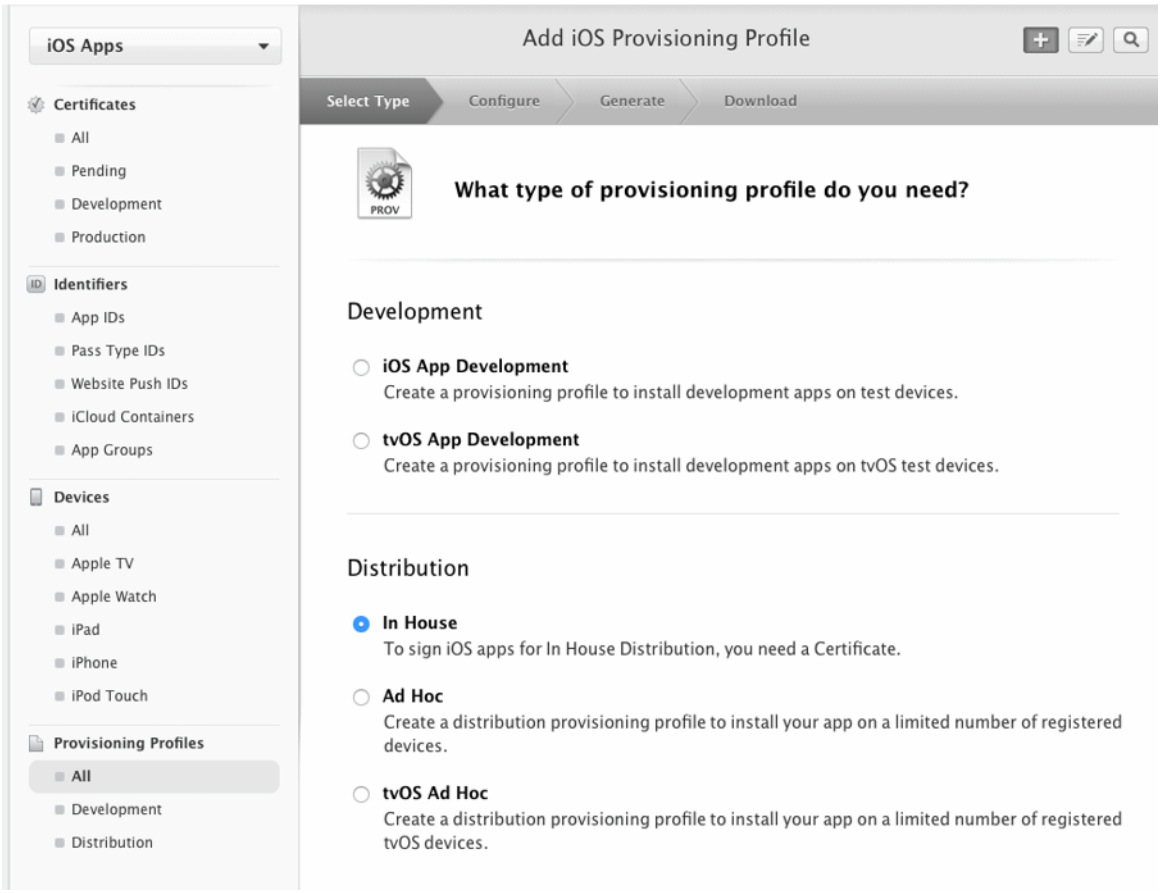
3. Continue をクリックしてください。
4. 次の画面上で、Submit をクリックしてください。これで Apple Developer 内でアプリケーションインスタンスを作成します。

Provisioning Profile

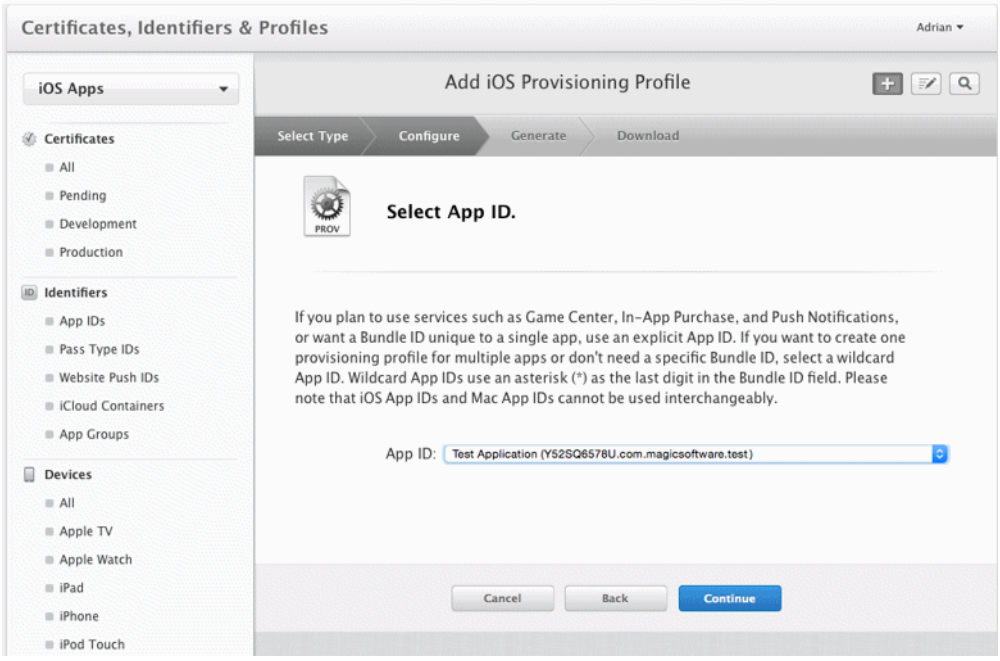
App ID を作成したら、最後のステップは、アプリケーションのためにプロビジョニングプロファイルを作成することです。プロビジョニングプロファイルは、アプリケーションがどのように使用され、配備されるかを定義します。

この処理は、開発マシン上で Apple Developer のアカウント証明書のもとでアプリケーション定義を作成するであろうアプリケーションプロビジョニングプロファイルを提供します。

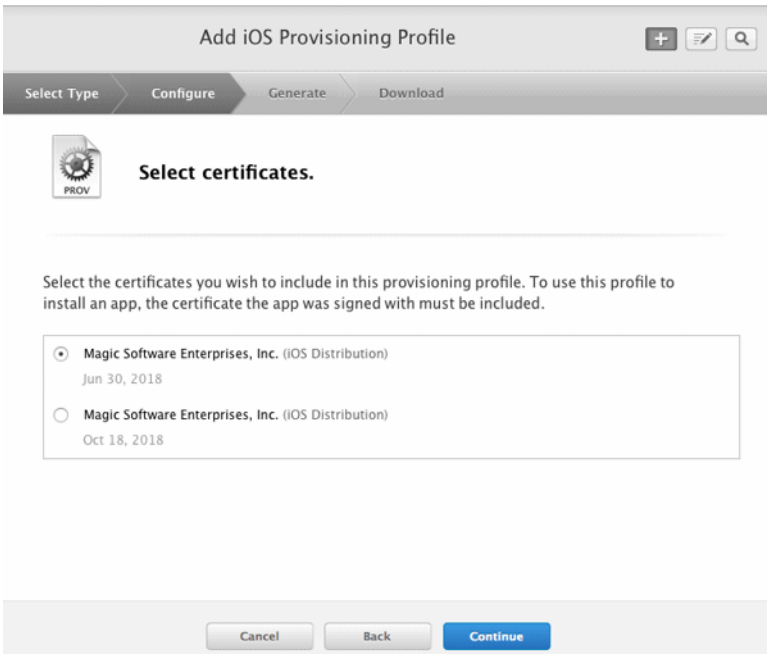
1. iOS,tvOS,watchOS の下の ***Provisioning Profiles*** セクションで、***All*** をクリックしてください。
2. この例は、内部向けに配布されるエンタープライズアプリケーションの作成に関係しています。従って、配布エリアとして「In House」を選択する必要があります。



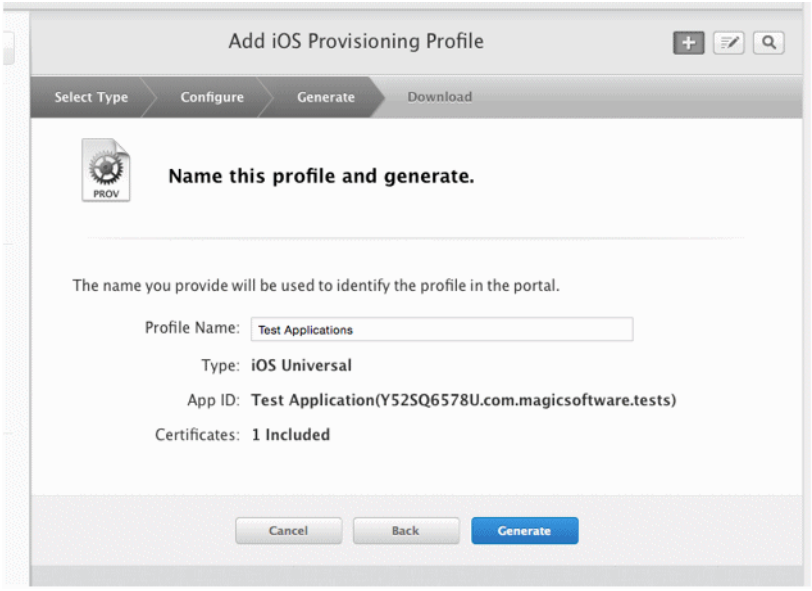
3. Continue をクリックしてください。
4. このプロビジョニングプロファイルがどのアプリケーション（Developer ポータルの App ID アーカイブからの）で使用するかを定義するために、以前作成したアプリケーションを選択してください。各プロビジョニングプロファイルは、1 つのアプリケーションとのみリンクできます。



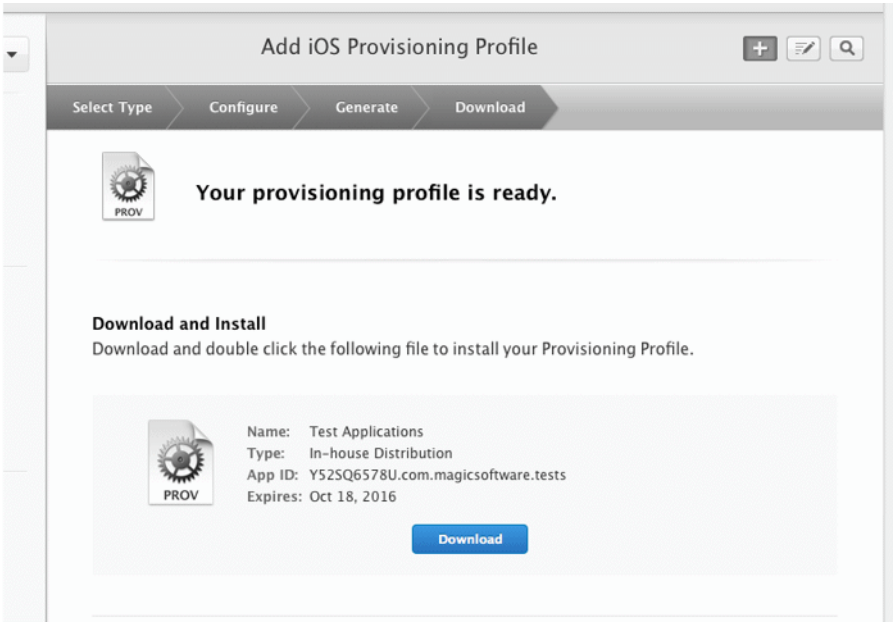
5. Continue をクリックしてください。
6. 次に、このプロビジョニングプロファイルが関連するう証明書を選択してください。これは、認可された開発マシンまたはそこにインストールされた Apple Developer アカウントの証明書を持つマシンだけが .ipa ファイルまたは iOS アプリのインストーラを作成することができると保証するものです。



7. 記録保持の目的のためにこのプロフィールの名前を入力し、Generate をクリックしてください。



8. 開発マシンに .mobileprovisions ファイルをダウンロードするために、Download をクリックしてください。



9. プロビジョニングプロファイルを Xcode の環境にインストールするために、ダウンロードされた .mobileprovisions ファイルをダブルクリックしてください。

参照： 第 41 章：「Xcode を使用して編集するために、手動で iOS プロジェクトを変更するには」（867 ページ）

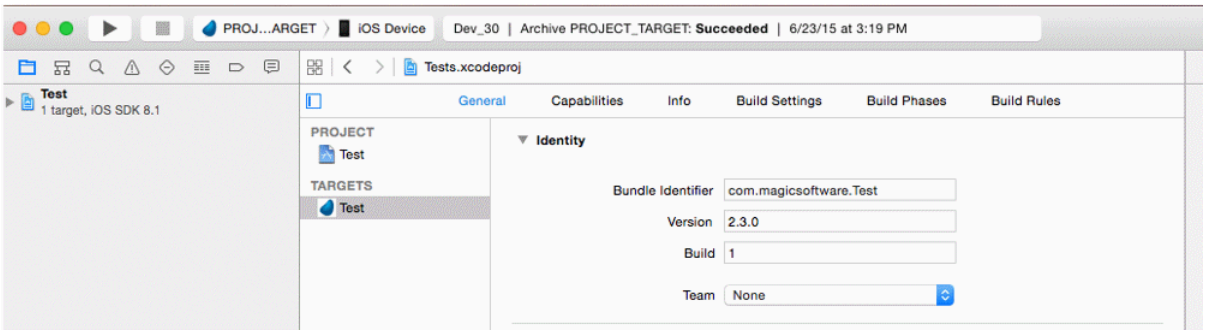
Xcode を使用して編集するために、手動で iOS プロジェクトを変更するには

証明書や App ID、およびプロビジョニングファイルを作成したら、それらを使用するために iOS プロジェクトを変更する必要があります。

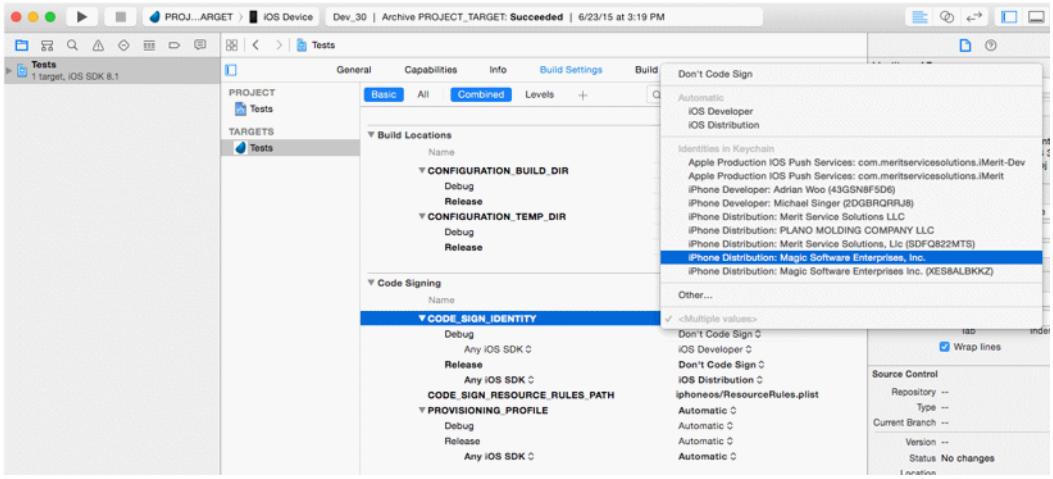
これは、RIA インタフェースビルダを使用したり、settingsTRAGETS.properties ファイルを変更することで、または直接 Xcode を使用して設定を変更することができます。

Xcode で変更するには :

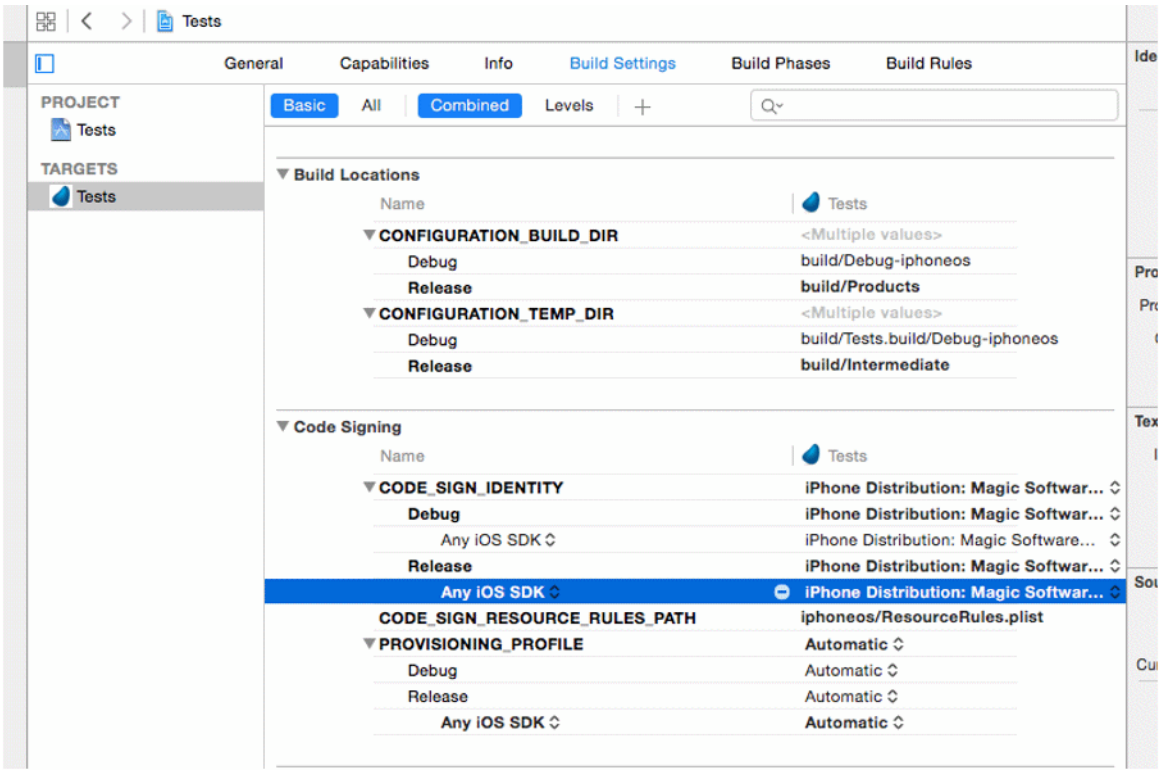
1. プロジェクトの TARGETS の General セクションを開き、**Bundle Identifier** の値を、App ID のセクションで定義した Bundle ID と合わせるようにします。



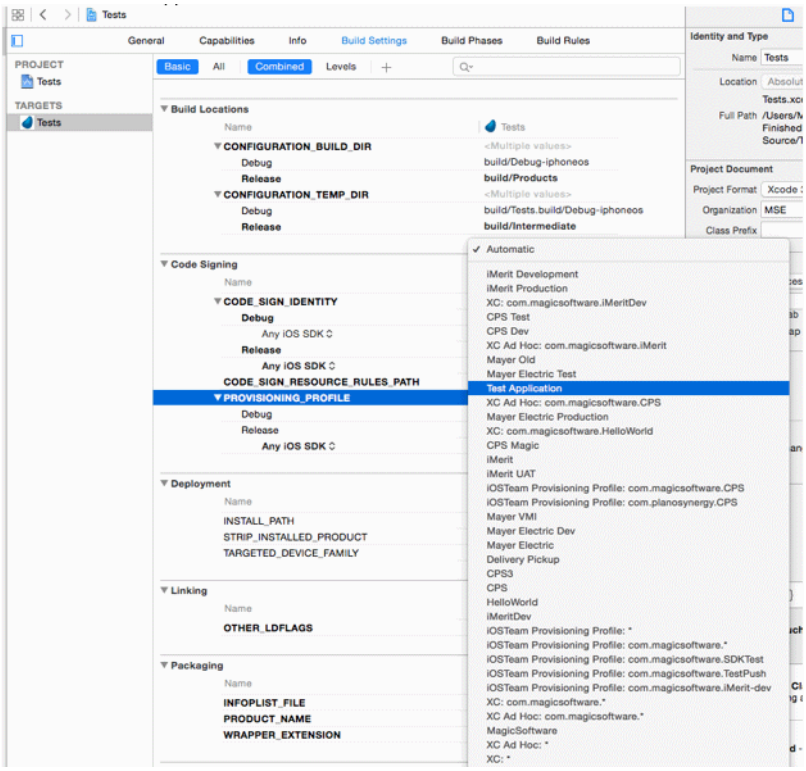
2. プロジェクトの TARGETS の Build Settings で、Code Signing セクションに **CODE_SIGN_IDENTITY** 設定が含まれています。ここには、以前定義した証明書の内容を選択します。



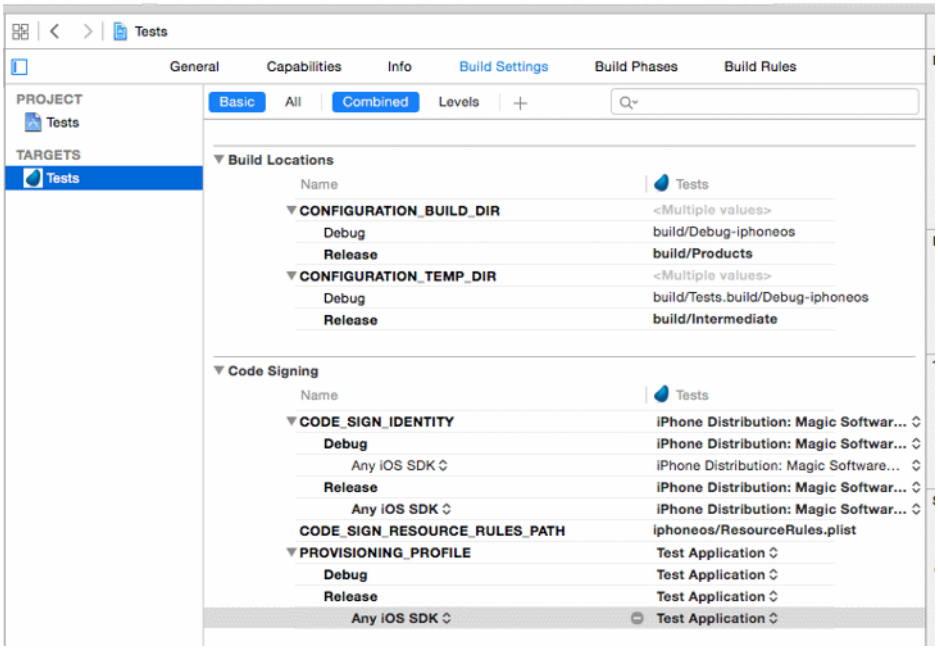
3. 他のフィールドの値は、証明書を変更すると自動的に変わります。5 つのフィールドが以下のような場合（iPhone Distribution: Magic Softwar…）。それらの値として証明書を表示しないでください。証明書と合わせるためにこれらの値を変更してください。



4. プロジェクトの TARGETS の Build Settings で、Code Signing セクションに **PROVISIONING_PROFILE** 設定が含まれています。ここには、以前定義したプロビジョニングプロファイルを選択します。



5. **PROVISIONING_PROFILE** セクション内のすべてのフィールドが、プロビジョニングプロファイルと合うように設定されることを確認してください。?

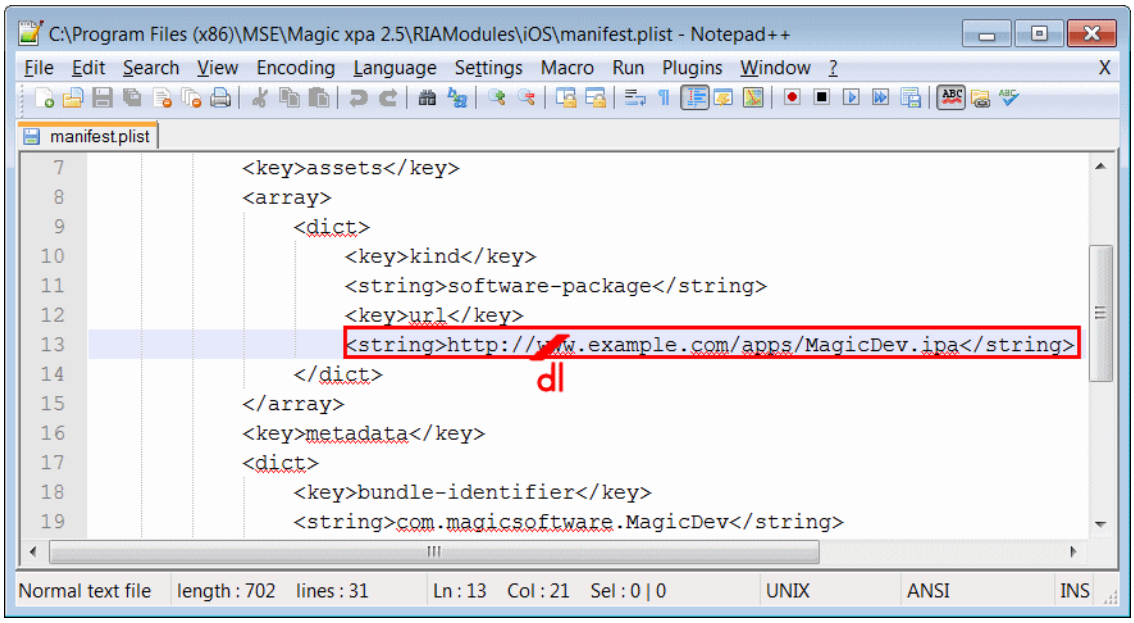


Xcode のプロジェクトは、これで Apple Developer のアカウントで認証された .ipa ファイルをビルドする準備ができました。

ドロップボックスを使用して iOS のインストールファイルを公開するには

ドロップボックスを使用して https サーバで IPA ファイルに対するリンク定義するには、下記の手順に従ってください。

1. PC 上にフォルダを作成して、ドロップボックス上のクラウドにフォルダを同期させるようにドロップボックスを定義してください。
2. このフォルダに、install.html、.plist と .ipa ファイルを配置してください。
3. ipa ファイルで右クリックして、共有ドロップボックスのリンクをコンテキストメニューから選択してください（クリップボード内に‘https://www.dropbox.com/s/qgknrfngaxazm38/app.ipa’のように書き込まれているはずです）。
4. テキストエディタで plist ファイルを開き、ipa ファイルに対する http アドレスを含む文字列を、ipa ファイルに対するリンクとして貼り付けてください。必ず dl（‘https://dl.dropbox…’が貼り付けられます）を www に置き変えてください。



5. plist ファイルで右クリックして、コンテキストメニューから共有ドロップボックスのリンクを選択してください。
6. テキストエディタで HTML ファイルを開き、.plist ファイルに対する http アドレスを含む文字列をリンクに貼り付けてください。必ず dl（‘https://dl.dropbox…’が貼り付けられます）を www に置き変えてください。
7. HTML ファイルで右クリックして、コンテキストメニューから共有ドロップボックスのリンクを選択してください。
8. テキストエディタでリンクを貼り付けてください。必ず dl（‘https://dl.dropbox…’が貼り付けられます）を www に置き変えてください。

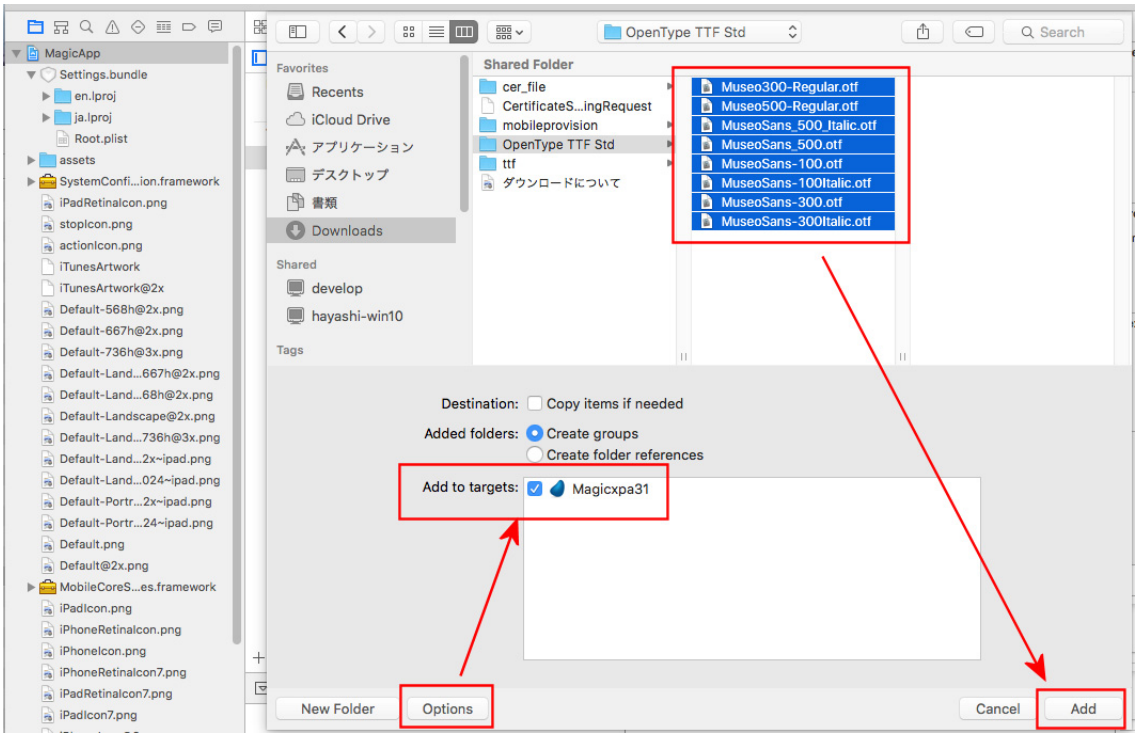
これがアプリをインストールするリンクとなります。

サードパーティー製のフォントを iOS アプリにインストールして使用するには

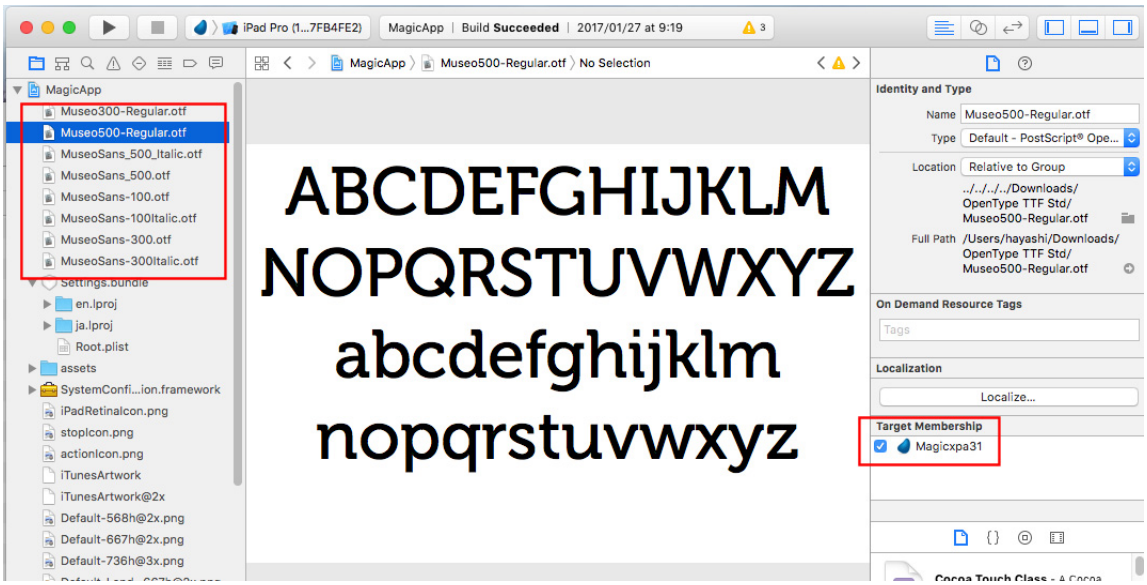
iOS アプリでサードパーティー製のフォントを使用するには、最初に、それらをクライアントアプリに含めて、次に Magic xpa アプリでそれらを使用する必要があります。

サードパーティ製フォントをクライアントアプリに追加する

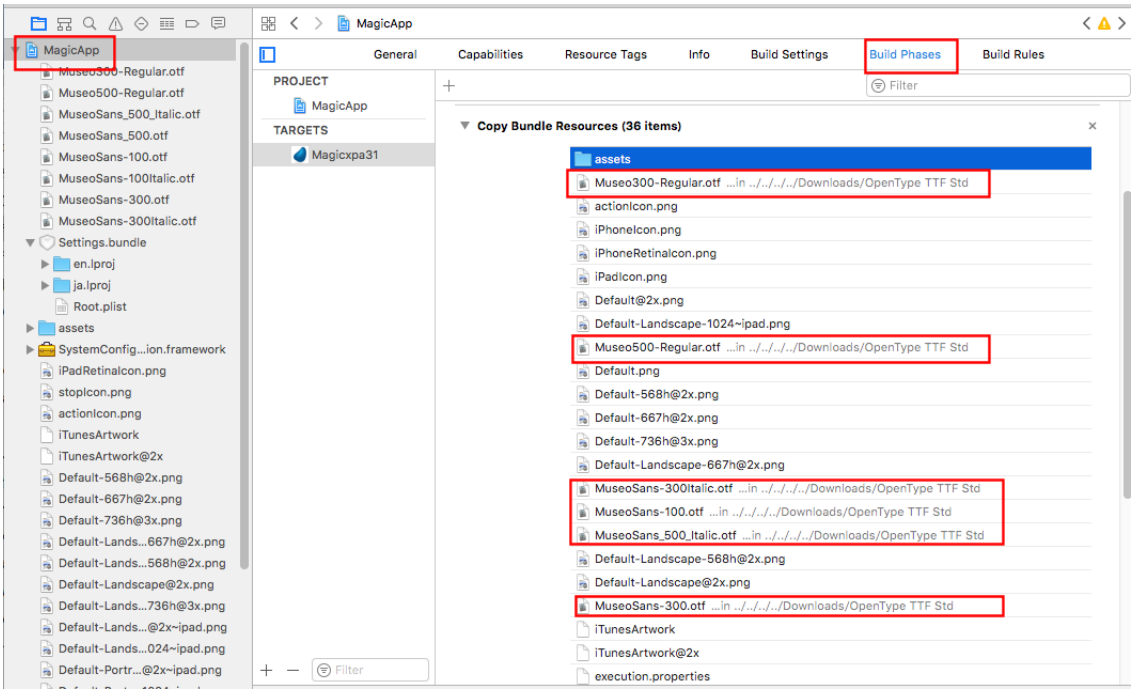
1. %EngineDir%RiaModules¥iOS¥Source から Mac に iOS クライアントアプリをコピーします。
2. Mac 上で XCode を起動します。
3. **File** メニューから **Open** をクリックし、iOS クライアントアプリの Source フォルダを選択してください。
4. MagicApp.xcodeproj ファイルを選択し、Open をクリックして開いてください。
5. 左側のプロジェクトツリービューでプロジェクト名（MagicApp）を右クリックし、**File** メニューの Add Files to MagicApp を選択してください。
6. 必要なフォントのすべてを選択し、Add をクリックしてください。Add to targets セクションに選択されたプロジェクトが表示されていることを確認してください。



7. 選択したフォントが build target に含まれていることを確認してください。左側のツリー表示上の MagicApp の下で、プロジェクト内にリストされた各フォントに対して、右側の properties ペインに表示される **Target Membership** に、正しい bundle id が選択されていることを確認してください。

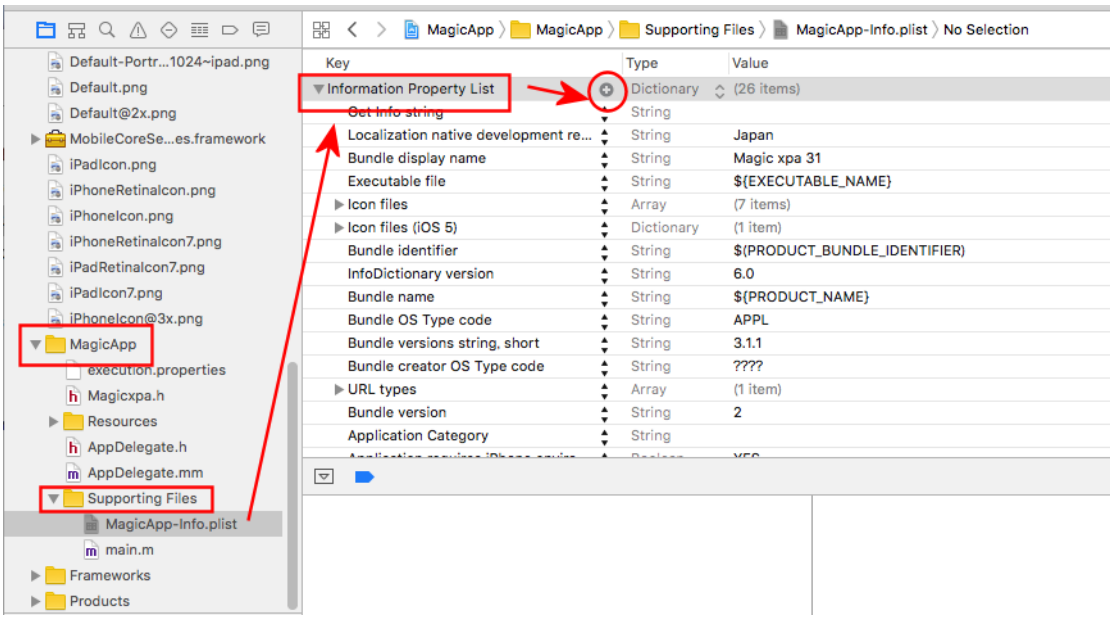


8. フォントが bundle 内の Resources として含まれることを確認してください。左側のツリー表示ペインでプロジェクトを強調表示し、中間のペインで Build Phases タブを選択し、追加されたフォントが **Copy Bundle Resources** セクションに表示されていることを確認してください。



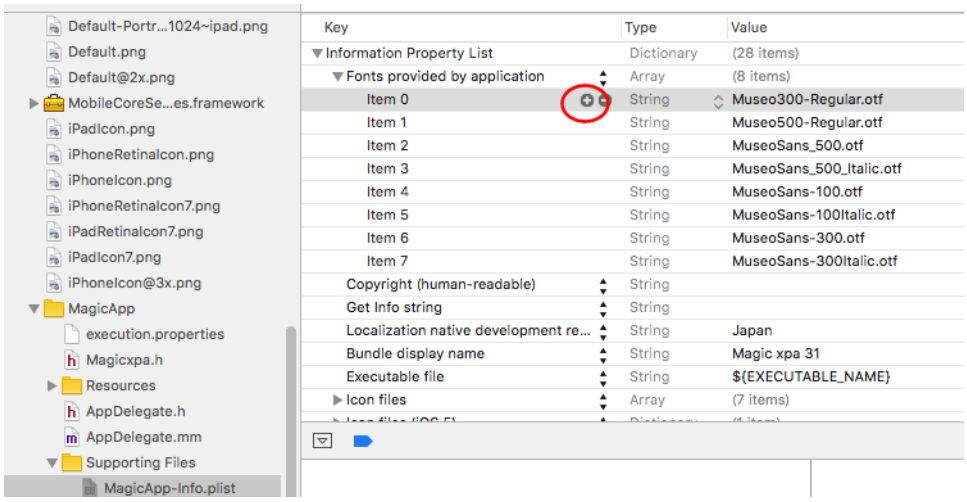
次に、サードパーティー製のフォントをアプリの .plist ファイルに含めたいと思っています：

1. 左側のツリー表示ペインで MagicApp/Supporting Files サブフォルダ内にあるアプリの .plist ファイルをクリックします（Magic xpa のデフォルトの .plist のファイル名は、MagicApp-info.plist です）。
2. 中間のペインの Information Property List セクションを強調表示し、+（プラス）アイコンをクリックしてください。



3. 新規行で次の値を正確に入力してください：Fonts provided by application
4. Array タイプのこの新規行にサブ行を追加してください。

各サブ行は、新しいサードパーティー製フォント用です。拡張が含まれていることや、ファイル名に誤りがないことを確認してください。



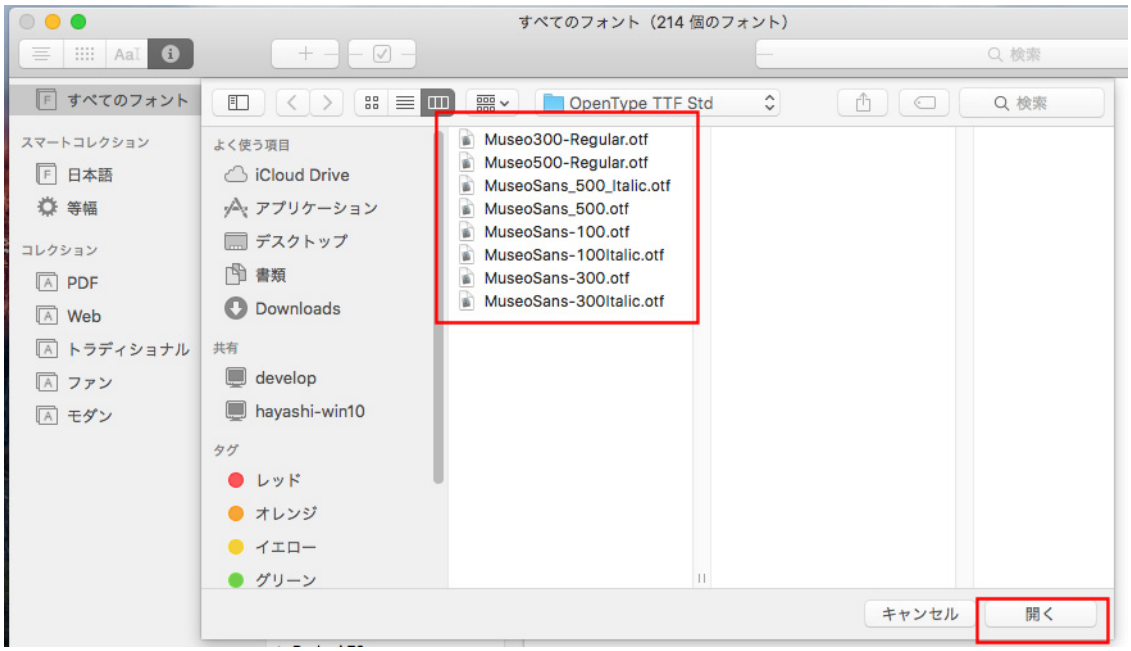
5. クライアントアプリをビルドしてください。

Magic xpa でサードパーティー製フォントを使用する

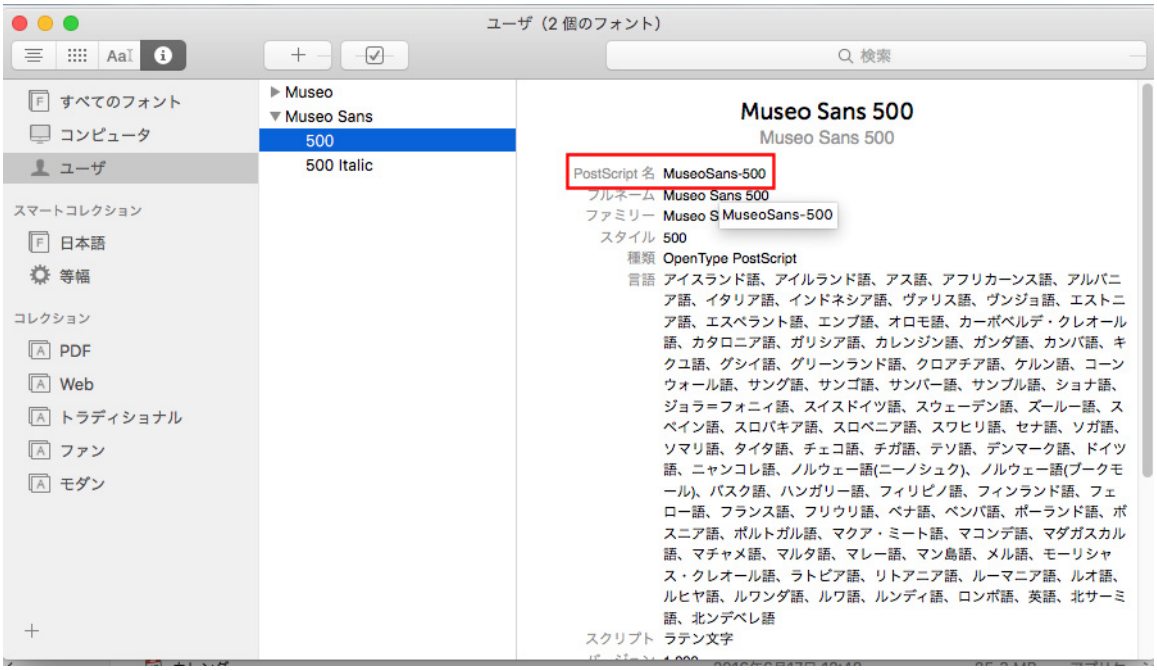
次のステップは、フォントの PostScripts 名を識別し、Magic xpa の環境に定義することです。

フォントの PostScripts 名を見つけるには：

1. OS X 上で、**Font Book** というユーティリティを起動します（Launchpad/ その他）。
2. ＋（プラス）をクリックし、必要なサードパーティー製フォント用を選択します。［開く］をクリックすると追加されます。



3. サードパーティー製フォントのうち 1 つを選択すると、その PostScript 名が下に表示されます。

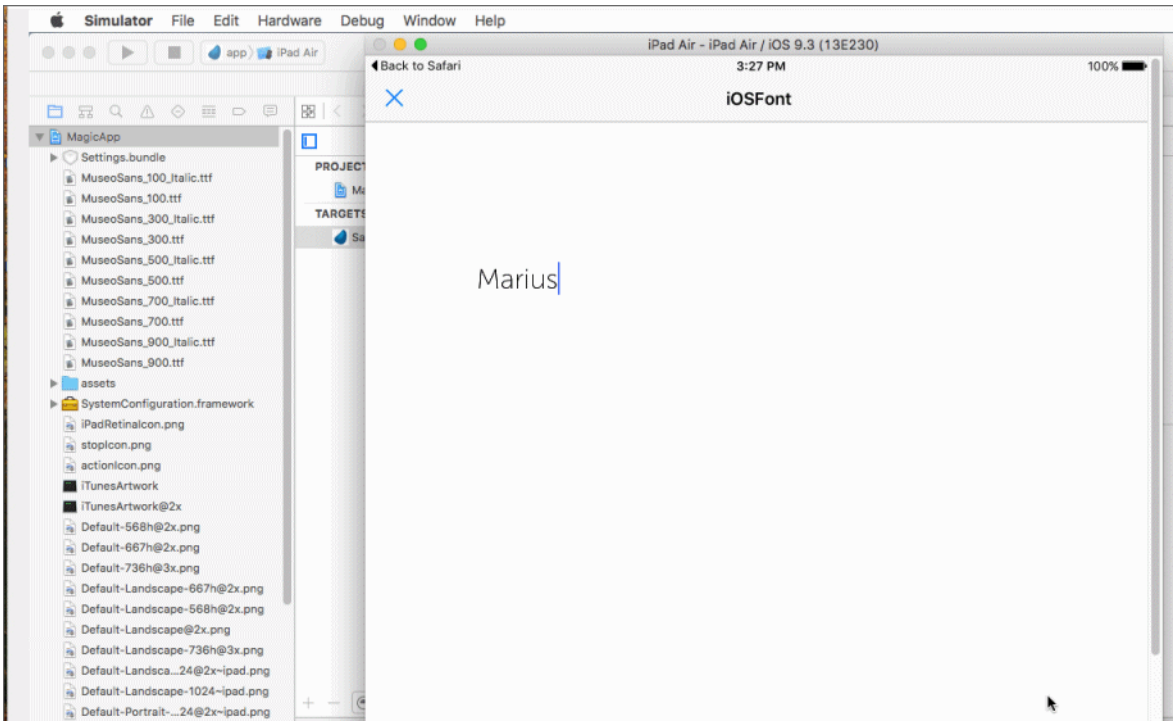


これは、Magic xpa のフォント定義ファイルで指定します。

アプリでこのフォントを使用するには：

4. Magic xpa アプリケーション内で使用するフォント定義ファイル（デフォルトでは、fnt_rnt.jpn）をテキストエディタを使用して開いてください。
5. 1 行追加して、各行に対して PostScript 名を使用して、サードパーティー製フォントを追加し、ファイルを保存してください。例 :: MyMuseoFont100,MuseoSans-100,12,0,0
6. ファイルの保存後に、Magic Studio を再起動します。
7. Magic xpa のアプリケーションで、表示コントロールに表示させたいフォントを定義します。

これで終了です。プロジェクトを実行し、クライアントアプリで接続すると新しいフォントが表示されます。


Simulator File Edit Hardware Debug Window Help
iPad Air - iPad Air / iOS 9.3 (13E230)
Back to Safari
3:27 PM
100%
iOSFont
Marius
MagicApp
Settings.bundle
MuseoSans_100_Italic.ttf
MuseoSans_100.ttf
MuseoSans_300_Italic.ttf
MuseoSans_300.ttf
MuseoSans_500_Italic.ttf
MuseoSans_500.ttf
MuseoSans_700_Italic.ttf
MuseoSans_700.ttf
MuseoSans_900_Italic.ttf
MuseoSans_900.ttf
assets
SystemConfiguration.framework
iPadRetinaIcon.png
stopIcon.png
actionIcon.png
iTunesArtwork
iTunesArtwork@2x
Default-568h@2x.png
Default-667h@2x.png
Default-736h@3x.png
Default-Landscape-667h@2x.png
Default-Landscape-568h@2x.png
Default-Landscape@2x.png
Default-Landscape-736h@3x.png
Default-Landscape-1024~ipad.png

サードパーティー製のフォントを Android アプリにインストールして使用するには

Android アプリの中でサードパーティー製フォントを使用するには、最初に、それらをクライアントアプリに含めて、次に Magic xpa アプリケーション側でそれらを使用する必要があります。

サードパーティ製フォントをクライアントアプリに追加する

1. %EngineDir% RIAModules¥Android¥Source¥app¥src¥main¥assets フォルダにフォントファイルをコピーします。
2. クライアントアプリをビルドしてください。

Magic xpa でサードパーティー製フォントを使用する

アプリケーションでこのフォントを使用するには：

1. Magic xpa アプリケーション内で使用するフォント定義ファイル（デフォルトでは、fnt_rnt.jpn）をテキストエディタを使用して開いてください。
2. 1 行追加して、サードパーティー製フォントを追加し、ファイルを保存してください。2 番目のパラメータには、フォントファミリー名（フォントファイル名に、拡張子を削除した名前）を指定します。
   例えば、MuseoSans-100.TTF フォントでは、以下のように定義します：MyMuseoFont100,MuseoSans-100,12,0,0
3. ファイルを保存し、Magic Studio を再起動します。
4. Magic xpa のアプリケーションで、表示コントロールに表示させたいフォントを定義します。

これで終了です。プロジェクトを実行し、クライアントアプリで接続すると新しいフォントが表示されます。

サポートバージョン :3.2

［このページは意図的に空白にしています。］

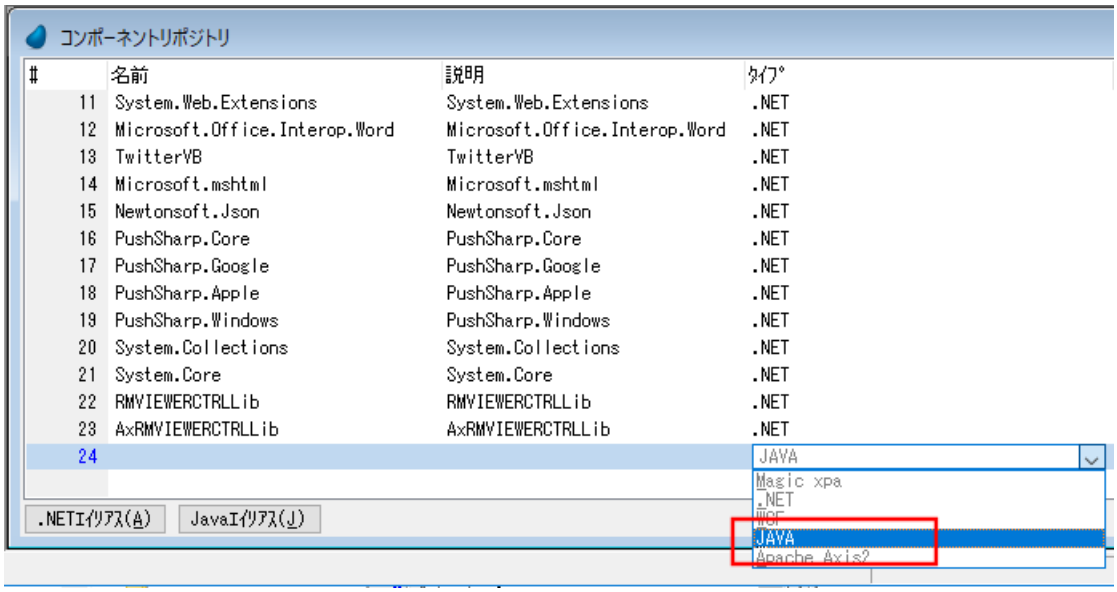
第 42 章 : Java 統合

Java オブジェクトを定義するには

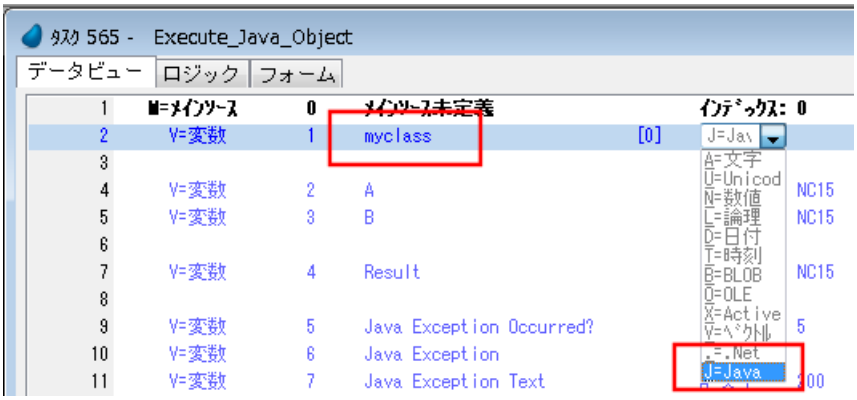
Jar の定義

Java オブジェクトを使用するには、最初にオブジェクトが定義された Jar（Java Archive）を読み込む必要があります。

この作業は、［コンポーネント］リポジトリで行います。



1. ［コンポーネント］リポジトリで F4 を押下して一行追加します。
2. ［タイプ］カラムで「Java」を選択します。
3. ［名前］カラムでズームして Jar ファイルを選択します。Jar が読み込まれると、［名前］欄に自動的に名前が設定されます。

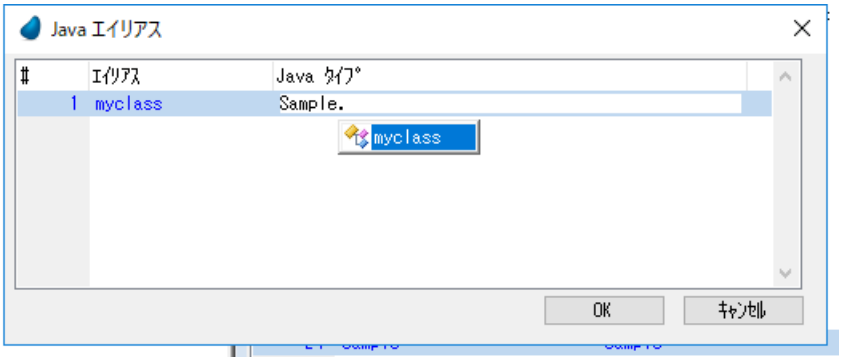


注： Jar を読み込むには、Magic.ini の [MAGIC_JAVA] セクションで以下の設定が必要です。

- "JVM_PATH=" 32 ビット版の jvm.dll をフルパスで定義
- "CLASSPATH=.;Support*;"

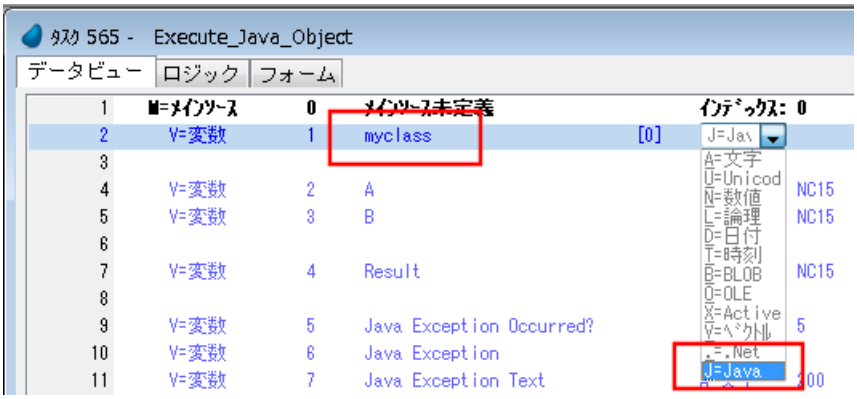
エイリアスの定義

よく使用する Java オブジェクトにエイリアス（別名）を設定することで、プログラム内で定義しやすくなります。エイリアスは、［コンポーネント］リポジトリの下部にある［Java エイリアス］ボタンをクリックすることで定義ダイアログを表示して行います。

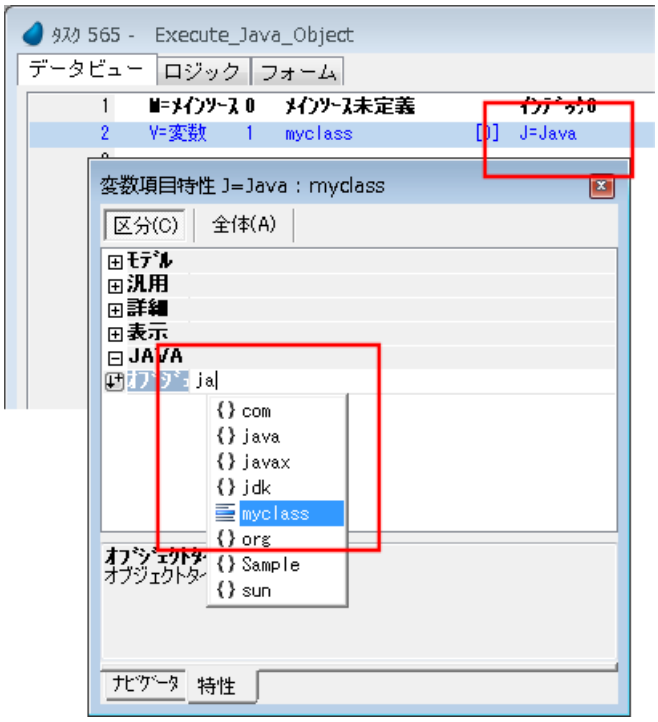


Java 変数の定義

プログラムで Java オブジェクトを使用する場合は、Java 型のデータ項目を定義する必要があります。



1. ［データビュー］タブで、変数項目を作成します。
2. Java データの型を選択します。
3. Java 変数の特性で、［オブジェクトタイプ］を入力します。ここで、"Ja" という単語を入力すると、選択リストに候補が表示されます。
4. リストから Java を選択します。



ここには、［コンポーネント］リポジトリに定義されている、Java オブジェクトやエイリアスの名前も表示されます。

Java 型のデータ項目を定義することで、Java オブジェクトとして使用することができます。内容によってはフォームに配置したり、イベントで使用することができます。

注： モデルを使用することで、上記の作業のほとんどを省略することができます。

Java のメンバ変数やメソッドにアクセスするには

Jar ファイルをロードし Java 変数を作成したら、［ロジック］エディタで Java 変数を使用することができます。

Java クラス内の public 定義された変数やメソッドに直接アクセスすることができます。

Java 変数のインスタンス化

Java 変数は、コンストラクタによって手動で初期化する必要があります。コンストラクタは、オブジェクトをインスタンス化して初期化します。コンストラクタは、クラスと同じ名前のがメソッドです。

Java 変数を既存の Java オブジェクトで更新するか、Java オブジェクトを作成する Java メソッドを実行して、Java 変数を初期化することができます。

たとえば、"myclass" を Java クラスとして定義した 変数項目 A をインスタンス化するには、Java.sample.myclass（）で更新します。

コンストラクタの使用は、一度だけ実行する必要のある操作です。これがもう一度実行されると、オブジェクトの新しいインスタンスが作成されます。コンストラクターを定義する場所は特別な注意を払う必要があります。

メンバ変数に値を設定する

Java オブジェクト内のメンバ変数に値を設定する場合は、JavaSet 関数を使用します。

myclass 内の "st2" という public 定義されたメンバ変数に値を設定する場合は、［アクション］処理コマンドで以下の式を呼び出します。

例： JavaSet(A.st2,B)

- **A …… Java 変数**
- **B …… 数値項目**

メンバ変数の値を取得する

Java オブジェクト内のメンバ変数の値を取得する場合は、Java 項目を使用します。

myclass 内の "st2" というメンバ変数の値を取得する場合は、［項目更新］処理コマンドで以下の式を実行します。

例： 例： 項目更新 D A.st2

- **A …… Java 変数**
- **D …… 戻り値（数値項目）**

メンバメソッドの呼び出し

Java オブジェクト内のメソッドを呼び出すには、Java 変数を使用した Magic 関数のようにして使用します。

myclass 内に add という加算機能を持つメンバメソッドを呼び出す場合は、項目更新で以下のように定義します。

例 : 項目更新 D　A.add(B,C)

- A …… Java 変数
- B/C …… 数値項目
- D …… 戻り値（数値項目）

スタティックな変数やメソッドにアクセスするには

Java クラス内に定義されたスタティック変数やメソッドは、直接アクセスすることができます。

Java 変数の作成やインスタンス化を行う必要はありません。

スタティック変数に値を設定する

Java オブジェクト内のスタティック変数に値を設定する場合は、JavaSet 関数を使用します。

myclass 内の “st“ というスタティック変数に値を設定する場合は、［アクション］処理コマンドで以下の式を呼び出します。

例： JavaSet(Java.myclass.st,A)

- A …… 数値項目

スタティック変数の値を取得する

Java オブジェクト内のスタティック変数の値を取得する場合は、Java 項目を使用します。

myclass 内の “st“ というスタティック変数の値を取得する場合は、［項目更新］処理コマンドで以下の式を実行します。

例： 例： 項目更新 B　　Java.myclass.st

- B …… 戻り値（数値項目）

スタティックメソッドを呼び出す

スタティックメソッドを呼び出すには、式の中で Java のメソッドを定義することで呼び出すことができます。

例 : 項目更新 B　Java.java.lang.Integer.toHexString(A)

- A …… 数値
- B …… 文字列（戻り値）

文字列で Java クラスを定義するには

Magic xpa の Ver3.2 以前で Java クラスにアクセスするには、OS 上でクラスパスを定義するか、MAGIC.INI でクラスファイルを定義する定義する必要があります。

MAGIC.INI でクラスを定義する

1. MAGIC.INI ファイルをテキストエディタで開きます。
2. [MAGIC_JAVA] セクションの ”CLASSPATH=” パラメータにクラスファイルか、ファクラスファイルが配置されているフォルダを指定します。指定されたフォルダ名と同じファイルのクラスファイルを探します。

例： [MAGIC_JAVA]CLASSPATH=SampleProjects¥¥OnlineSamples¥¥Resources¥¥MyClass;

OS 上で定義する場合は、環境変数「CLASSPATH」で定義します。

JAR ファイルの場合は、ファイル名で指定します。

Java クラスを確認する

Magic xpa でアクセスできる Java クラスを確認するには、JExplore 関数を使用します。

例えば、上記の myclass を確認するには、以下のように関数を実行します。

JExplore(myclass)

戻り値は、Blob 型で クラス内に定義されているメソッドやパラメータの内容が返ります。

指定したクラスにアクセスできない場合は、空白になります。

Ver3.2 以前の方法で Java オブジェクトにアクセスするには

Ver3.2 以前の方法で Java オブジェクトにアクセスすることもできます。

Java クラス内に定義されたインスタンス変数やメソッドにアクセスするには、事前に Java クラスをインスタンス化する必要があります。

Java クラスをインスタンス化する

Java クラスをインスタンス化するには、JCreate 関数を使用します。

JCreate(クラス名 , コンストラクタのシグニチャ , パラメータ)

戻り値として、インスタンスの疑似参照（実際にインスタンスが存在するメモリ上の位置）が BLOB 形式で返ります。

例 ; JCreate ('myclass','()V')

インスタンス変数に値を設定する

インスタンス変数に値を設定するには、JSet 関数を使用します。

JSet(疑似参照 , 変数名 , 変数のシグニチャ , 値)

例 :JSet(<JCreate の戻り値 >,'cc','I',< 値 >)

インスタンス変数から値を取得する

インスタンス変数から値を設定するには、JGet 関数を使用します。

JGet(<JCreate の戻り値 >,'cc','I')

例 :JGet(<JCreate の戻り値 >,'cc','I')

インスタンスメソッドを呼び出す

インスタンスメソッドを呼び出すには、JCall 関数を使用します。

JCall(疑似参照 , メソッド名 , メソッドのシグニチャ , パラメータ)

例 :JCall (<JCreate の戻り値 >,'add','(II)I',< パラメータ 1>,< パラメータ 2>)

Ver3.2 以前の方法でスタティックな変数やメソッドにアクセスするには

Ver3.2 以前のように　Java 関数を使用することでクラス内に定義されたスタティック変数やメソッドは、直接アクセスすることができます。

スタティック変数に値を設定する

スタティック変数に値を設定するには、JSetStatic 関数を使用します。

JSetStatic（クラス名．変数名，変数のシグニチャ，値）

例：JSetStatic('myclass.st','I',<値>)

スタティック変数から値を取得する

スタティック変数から値を設定するには、JGetStatic 関数を使用します。

JSetStatic（クラス名．変数名，変数のシグニチャ）

例：JSetStatic('myclass.st','I')

スタティックメソッドを呼び出す

スタティックメソッドを呼び出すには、JCallStatic 関数を使用します。

JCallStatic(クラス名．メソッド名，メソッドのシグニチャ，パラメータ)

```
例：JCallStatic(
  'java.lang.Integer.toHexString',
  '(I)Ljava/lang/String;',D)
```

第 43 章 : Web Client の開発

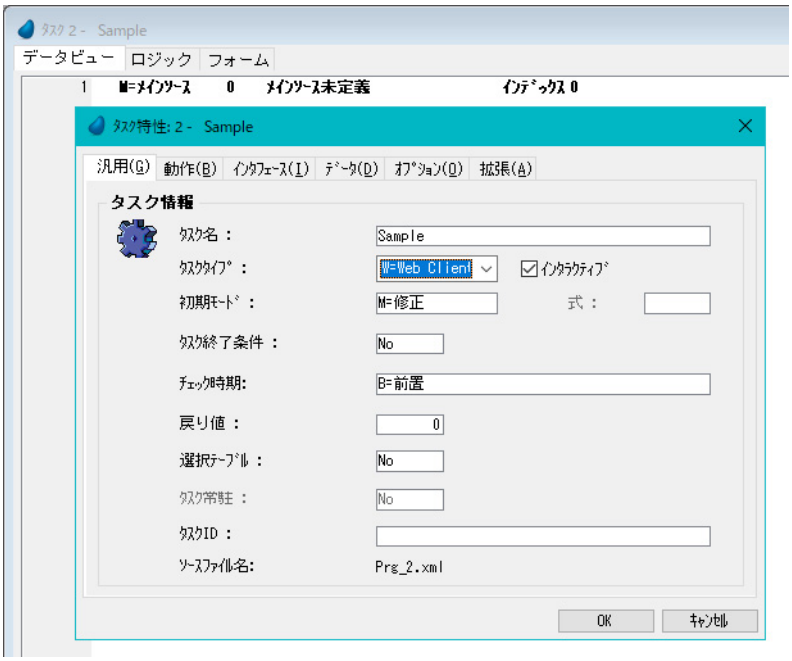
Web Client アプリケーションを生成するには

Web Client アプリケーションを生成するには、以下の手順に従います。

注 : ［動作環境］の［実行モード］を「B= バックグラウンド」に設定して Magic xpa Studio を起動してください。

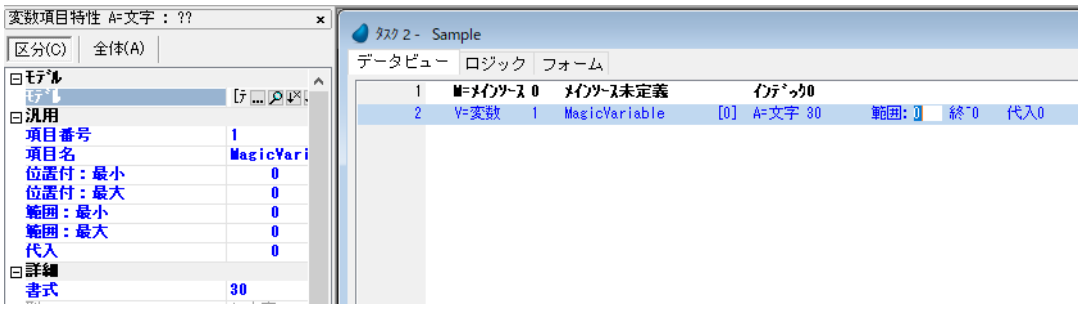
Web Client アプリケーションを作成する

1. ［ファイル］メニューをクリックします。
2. ［新規作成］メニューをクリックします。
3. プロジェクト名を入力します。例えば、「HowToSamples」とします。
4. ［プログラム］リポジトリを開き、F4 を押下します。これで新しいプログラムが作成されます。
5. プログラムの名前を「Sample」とします。F5 を押下します。［タスク特性］ウィンドウが表示されます。
6. ［タスクタイプ］を「Web Client」に変更して、「OK」をクリックします。



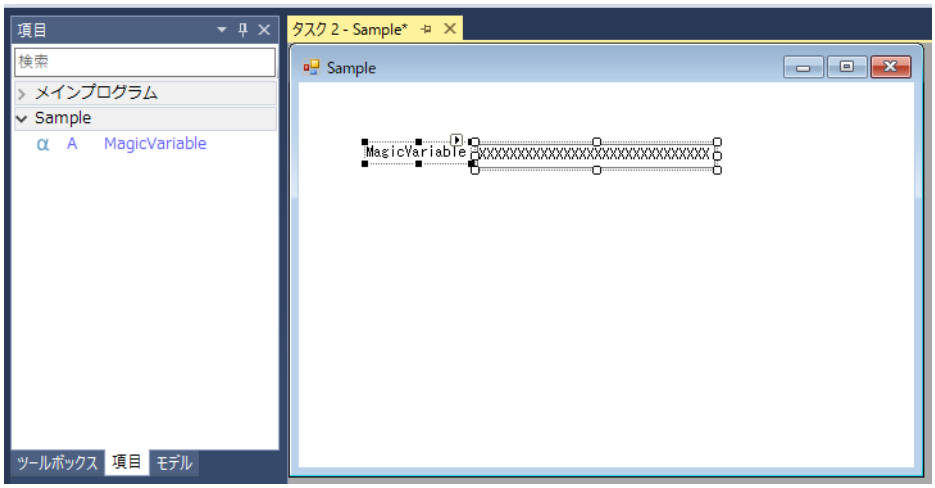
変数項目の作成

1. F5 を押下してプログラム「Sample」を開きます。
2. ［データビュー］タブを開き、F4 を押下します。
3. 変数項目の名前を「MagicVariable」とし、必要に応じて型と書式を設定します。ここでは §「A= 文字 : 30」としました。
4. 任意の値を初期値として設定します。



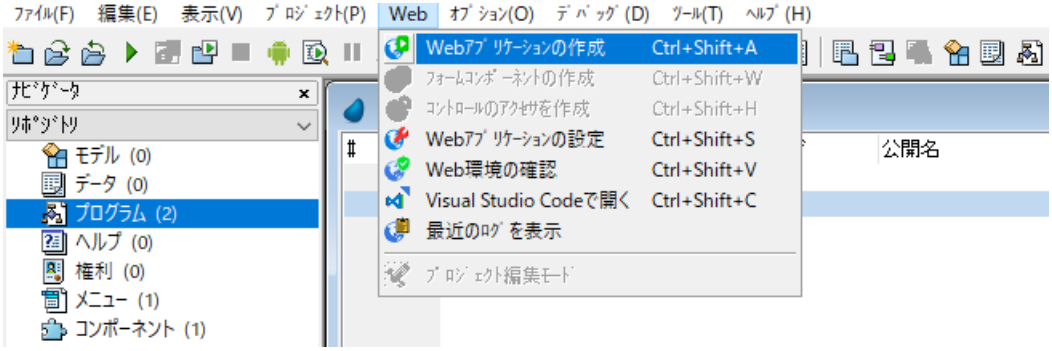
項目コントロールの作成

1. ［フォーム］タブを開きます
2. F5 を押下します。
3. ［項目］タブから項目コントロールをフォームにドラッグ＆ドロップします。

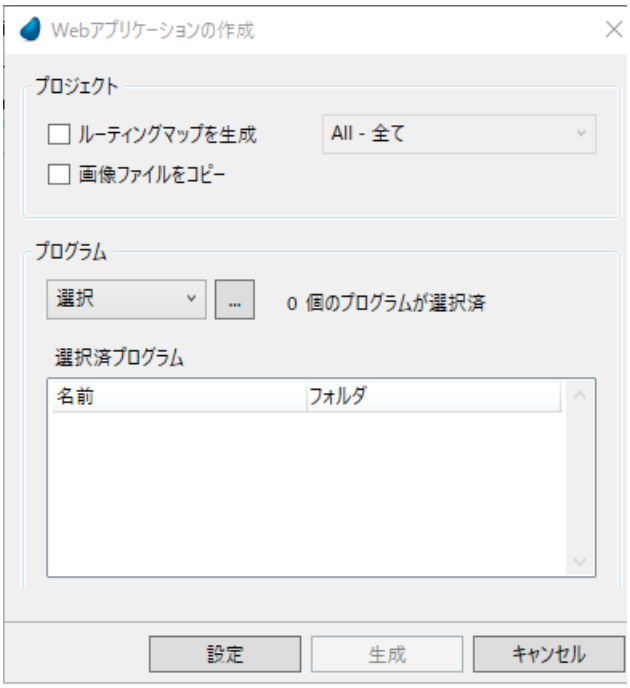


Angular プロジェクトを作成する

1. ［Web］メニューをクリックします。
2. メニューの「Web アプリケーションの作成」を選択します。



3. 次のような［Web アプリケーションの生成］ダイアログボックスが開きます。



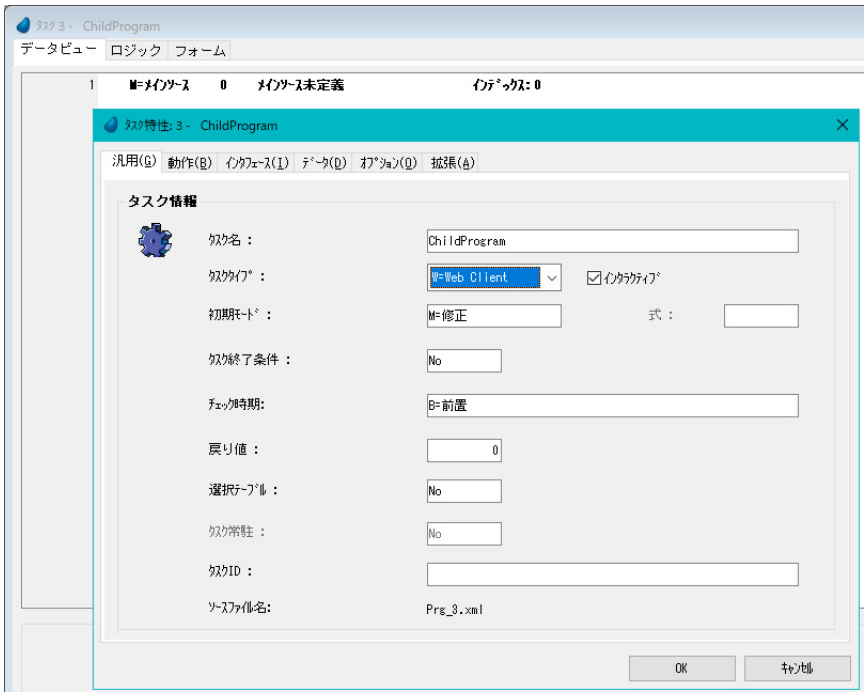
4. ［プロジェクト］グループから任意のオプションを選択します。
   - ルートマッピングを生成 …… ルーティングマップを作成します。
   - 画像ファイルをコピー …… コントロールやフォームに割り当てられている画像ファイルをコピーします。これらのファイルは、生成されたアプリケーションの Source Images フォルダから assets\images フォルダにコピーされます。
5. ［プログラム］グループからオプションを選択します。
   - N= なし …… プログラムを選択したくない場合に選択します。
   - 選択 …… 各フォルダに表示されているすべてのプログラムのリストからプログラムを選択する場合に選択します。複数のプログラムを選択して、リストをフォルダごとにフィルタすることができます。
   - All - 全て …… Web アプリケーションを生成するためにすべての Web Client プログラムを選択する場合に使用します。
6. 選択済みプログラム …… プログラムを選択すると、このリストボックスに選択したプログラムのリストが表示されます。
7. ［生成］ボタンをクリックします。アプリケーションは、［Web アプリケーションの設定］ダイアログの［Web アプリケーションフォルダ］で指定されたフォルダに生成されます。
8. 生成プロセスをキャンセルするには、［キャンセル］をクリックします。

Web Client プログラムを追加／修正するには

既存の Web Client アプリケーションにプログラムを追加したり修正したりするには、以下の手順に従います。

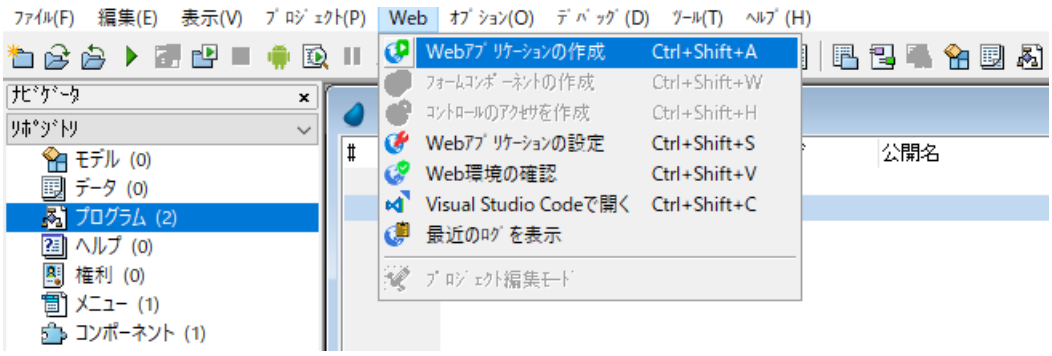
Web Client プログラムを追加する

1. ［プログラム］リポジトリを開き、F4 を押下し新しいプログラムが作成されます。
2. プログラムの名前を「ChildProgram」とします。F5 を押下します。［タスク特性］ウィンドウが表示されます。
3. ［タスクタイプ］を「Web Client」に変更して、「OK」をクリックします。
4. 必要に応じて［データビュー］や［ロジック］、［フォーム］を定義してください。

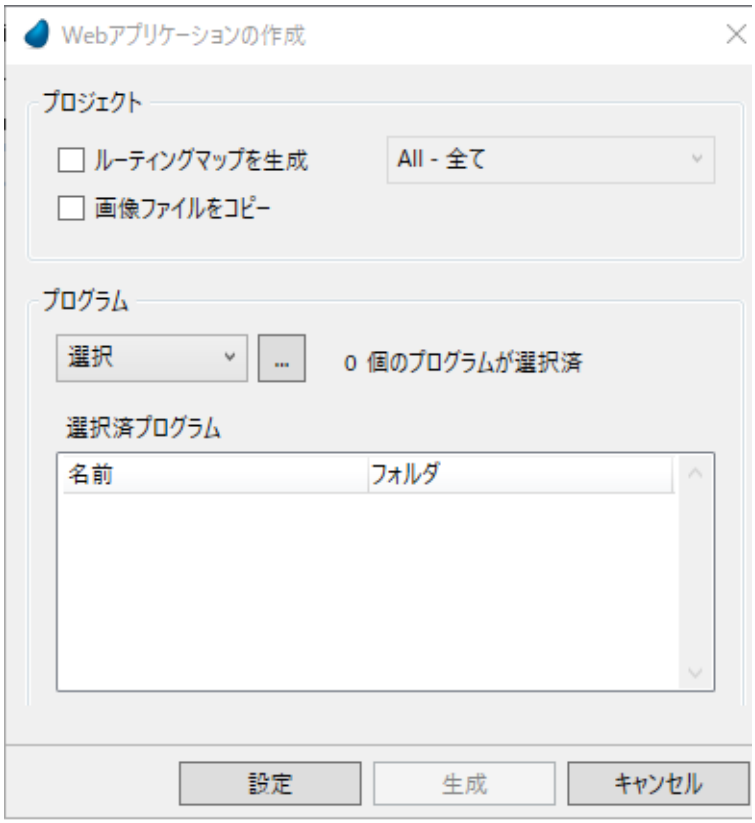


Angular プロジェクトを更新する

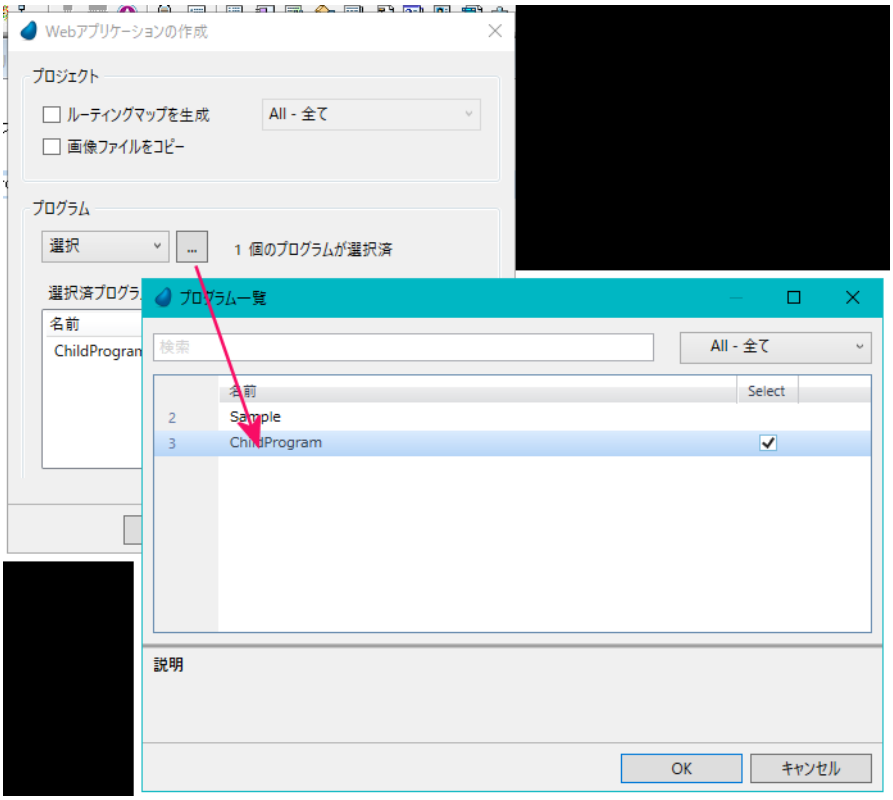
1. ［Web］メニューをクリックします。
2. メニューの「Web アプリケーションの作成」を選択します。



3. 次のような［Web アプリケーションの生成］ダイアログボックスが開きます。



4. ［プログラム］グループで「選択」を選択します。
5. ［選択 . ボタンをクリックして［プログラム一覧］を開き、追加したいプログラムを右側の［Select］カラムをチェックします。



6. 選択済みプログラム …… プログラムを選択すると、このリストボックスに選択したプログラムのリストが表示されます。

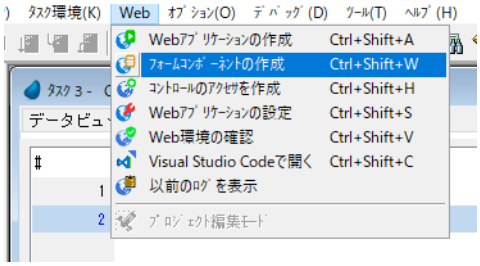
7. ［生成］ボタンをクリックすると、［Web アプリケーションの設定］ダイアログの［Web アプリケーションフォルダ］で指定されたフォルダにアプリケーションが生成されます。
8. 生成プロセスをキャンセルするには、[キャンセル］をクリックします。

Web Client プログラムを修正する

Web Client プログラムのフォームを修正した場合は、Angular コードに反映させる必要があります。

注：　プログラムのロジックを修正した場合は、必要ありません。

1. ［プログラム］リポジトリを開き、修正したいプログラムを開きます（F5）。
2. ［フォーム］タブを開き、フォームデザイナを開きます。
3. 必要に応じてフォームを修正します。
4. ［フォーム］タブを開いた状態で［Web］メニューをクリックします。
5. メニューの「フォームコンポーネントの作成」を選択します。

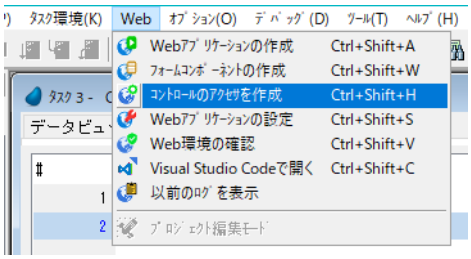


これで、フォームの修正内容が反映されます。

コントロールのアクセサコンポーネントを修正する

コントロールのアクセサコンポーネント（***.g.ts）は、フォーム上に定義されたコントロールのコントロール名などが定義されているファイルです。

1. このファイルのみを修正したい場合は、［フォーム］タブを開いた状態で［Web］メニューをクリックします。
2. ［Web］メニューの「コントロールのアクセサを作成」を選択します。



これで、アクセサファイルのみ修正されます。html ファイルは変更されないため、ビルドエラーになる場合があります。

サポートバージョン：4.5

Angular クライアントの npm バージョンを管理および制御するには

Angular クライアントの npm バージョンは、"npmVersion.json" というファイルに mpmic-xpa/Cli バージョンと magic-xpa/angular バージョンを個別に設定することで管理することができます。

この設定ファイルは、<Magic xpa のインストールフォルダ >WebClient¥Scripts フォルダ内にあります。指定されたファイルには 2 つのエントリがあります。

```
{
"CliVersion": "MajorVersion.MinorVersion*10.BuildNumber",
"AngularVersion": "MajorVersion. MinorVersion*10. BuildNumber" }
```

例えば

```
{
  "CliVersion": "4.801.5",
  "AngularVersion": "4.801.5"
}
```

Angular（Web Client ランタイム）および cli（Web Client Generation）パッケージに対応するバージョンは、Web サイトからダウンロードすることができます。

この目的は、Web Client アプリケーションを生成するために使用される Angular Client パッケージの特定のバージョンを利用するための開発者へのメソッドを提供することです。

スピナーをカスタマイズするには

スピナーをカスタマイズするには、Angular のテンプレートとのバインディングを使用する必要があります。

指定された手順に従ってスピナーをカスタマイズしてください。

1. テンプレートを作成します。
2. magic-root の SpinnerTemplate にバインドします。

1. テンプレートの作成

- <ng-template> …… ng-template ディレクティブは、Angular テンプレートを表します。<ng-template> はデフォルトでは表示されない（コメント化される) テンプレートを作成します。# で名前をつけます。

以下のテンプレートは、app.component.html に記述されています

スピナーテンプレートの 2 つの例を示します。

HTML/CSS を使用したテンプレート

HTML

app.component.html に以下を追加します。

```
<ng-template #mySpinner1>
   <div class="spinner"> </div>
</ng-template>
```

CSS

app.component.css に以下を追加します。

```
.spinner {
  width: 40px;
  height: 40px;
  margin: 100px auto;
  background-color: blue;
  border-radius: 100%;
  -webkit-animation: sk-scaleout1.0s infinite ease-in-out;
  animation: sk-scaleout 1.0s infiniteease-in-out;
}
@-webkit-keyframes sk-scaleout {
  0% { -webkit-transform: scale(0)}
  100% {
    -webkit-transform:scale(1.0);
    opacity: 0;
  }
}
@keyframes sk-scaleout {
  0% {
    -webkit-transform:scale(0);
    transform: scale(0);
  } 100% {
      -webkit-transform:scale(1.0);
      transform:scale(1.0);
      opacity:0;
    }
}
```

実行すると、青色の円形がスピナーとして表示されます。

GIF を使用したテンプレート

```
<ng-template #mySpinner2>
  <div>
    <img src="./assets/Spin.gif" width="64px" height="64px" >
  </div>
</ng-template>
```

2. テンプレートを magic-root にバインドする

- [SpinnerTemplate] …… これは templateRef（テンプレートの参照）であり、Magic のルートコンポーネントへの入力として機能します。

ユーザがこの templateRef を指定すると、スピナーとして使用されます。

app.component.html でこれを行うには、手順 1 で以前に作成したテンプレートを指定します。

```
<div>
    <magic-root [SpinnerTemplate]="mySpinner1" ></magic-root>
</div>
```

最新の Angular バージョンを使用して Web Client アプリケーションをアップグレードするには

ここでは、異なるバージョンの Angular を使用した Web Client アプリケーションの開発について説明します。

前提条件

- NodeJS のバージョンは 16.15.1-x64.msi です。

NodeJS のインストール

1. NodeJS を https://nodejs.org/en/download からダウンロードしてインストールします。
2. 次のコマンドを実行します。
   ```
   npm i g npm
   ```
3. NodeJS のバージョンが前提条件と合っていることを確認します。.

Angular で新規に Web Climnet アプリケーションを作成したり、既存の Web Client アプリケーションを Angular の最新版にアップグレードする場合は、以下のコマンドを実行してください。

```
npm config set legacy-peer-deps true
```

上記のアップグレードが完了したら、アプリケーションを Angular 15 にアップグレードする準備が整いました。

Angular 15 のインストール

1. スムーズな移行作業を行うために、以下のファイルがチェックアウトされていることを確認します。(読込 / 書込モード)。

- Angular.json
- Package.json
- Tsconfig.json
- App.routes.ts

2. 以下のコマンドを 1 つずつ実行します。

   a. ng-dynamic-component のバージョンが更新されているので、magic.gen.lib.module ファイルから 'withComponents' を削除する必要があります。
   ```
   DynamicModule.withComponents(magicGenComponents)
   ```
   b. アプリケーションに Lazy Loaded Modules がある場合、Lazy Loaded Modules の app.routes.ts ファイルでは、LoadOnDemandModuleMagicComponent

   i.import 文から削除します。
   ```
   import { Routes, RouterModule } from '@angular/router';
   import { RouterContainerMagicComponent, ?LoadOnDemandModuleMagicComponent?} from "@magic-xpa/angular";
   ```
   ii. ルートマップ (app.routes.ts) のコンポーネント名「LoadOnDemandModuleMagicComponent」を「RouterContainerMagicComponent」に置き換えます。
   ```
   export const routes: Routes = [
       {path: '',
        component:~~LoadOnDemandModuleMagicComponent~~  RouterContainerMagicComponent ,
        children: [
                  ]
       }
   ];
   ```
3. ここで、以下のコマンドを使用して、Angular のバージョンを 15 にアップデートします。

方法 1：以下のコマンドを使用します。

```
ng update @angular/core@9 @angular/cli@9 @angular/material@9 @angular/forms@9 @angular/common@9 @angular/cdk@9 --force

ng update @angular/core@10 @angular/cli@10 @angular/material@10 @angular/forms@10 @angular/common@10 @angular/cdk@10 --force

//For updating to 15, use the following commands:
npx @angular/cli@11 update @angular/core@11 @angular/cli@11 @angular/material@11 --force
```

```
npx @angular/cli@12 update @angular/core@12 @angular/cli@12 @angular/material@12 --force
npx @angular/cli@13 update @angular/core@13 @angular/cli@13 @angular/material@13 --force
npx @angular/cli@14 update @angular/core@14 @angular/cli@14 @angular/material@14 --force
npx @angular/cli@15 update @angular/core@15 @angular/cli@15 --force
// Please use version from Scripts/npmVersion.json

ng update @magic-xpa/cli@xxx?@magic-xpa/angular@xxx?--force
```

ここで、xxx は＜インストールフォルダ＞\WebClient\Scripts にある npmVersion.json ファイルに記述されているバージョンです。

例えば、Magic xpa 4.10＝4.100.0 となります。

```
npm i ng-dynamic-component@^7.0.1 --save
npm i rxjs-compat@6

npm i ngx-currency // This is an optional command

//For updating to 15, use the following commands:
ng update ngx-infinite-scroll@10.0.0 --force
ng update ng-dynamic-component@~10.1.0 ?force
```

すべてのライブラリが正しくインストールされ、package.json が更新されていることを確認します。?

ここで、以下のコマンドを使用して、Angular のバージョンを 15 にアップデートします。

方法 2：package.json を修正する

アップグレード前	アップグレード後
"@magic-xpa/cli": "4.801.6", "@magic-xpa/angular": "4.801.6", "@magic-xpa/cli": "^4.801.6",	"@magic-xpa/cli": "4.1000.0", "@magic-xpa/angular": "4.1000.0", "@magic-xpa/cli": "^4.1000.0",

1. package_lock.json ファイルを削除します。
2. node_modules フォルダを削除します。
3. 次のコマンドを実行します。:
   ```
   npm i
   ```

上記のコマンドを全て実行すると、以下のように変更されるはずです。

- Package.json: べての Angular ライブラリのバージョンは 15 になります (ただし、@angular/material と @angular/cdk は 14 になります)。
- ngx-infinite-scroll と ng-dynamic-component のバージョンは 10 です。
- Angular.json: lazyModules のような非推奨のシンボルは削除されます。

AppModule.ts で必要な変更

MagicModule.forRoot() の import から、forRoot() の部分を削除すします。

```
imports: [
:
MagicModule.forRoot()
```

ルーティングのないモジュールのレイジーローディングをサポートする

ルーティングで呼び出されない Magic xpa モジュールのレイジーローディングをサポートするためには、以下の手順が必要です。

i. 生成されたアプリケーションの \Magic フォルダ内で、ng g s lazy-loader を使？してサービスを作成します。

ii. 作成したサービスを以下のコードに置き換えます。

```
import { Injectable } from '@angular/core';
import { MagicLazyLoaderService } from '@magic-xpa/angular';
@Injectable()
export class LazyLoaderService extends MagicLazyLoaderService
```

```
{
    Load(path: string): Promise<any>
  {
    return import('./' + path + '/magic.gen.lib.module');
 }
}
```

iii 作成したサービスを、app.module.ts の provider セクションで、図のようにインジェクトします。

```
providers: [
    {
     provide: MagicLazyLoaderService, useClass : LazyLoaderService
    }
],
```

iv. インポートが正しく追加されていることを確認する。

NgxCurrencyModule に必要な変更点

ngx-currency モジュール最新版をご利用の方は、以下の変更を行ってください。

- magic.gen.lib.module の customCurrencyMaskConfig 宣言にデータ型 CurrencyMaskConfig を追加し、import 文を追加します。
- customCurrencyMaskConfig で、min と max の値を null ではなく undefined に設定します。

```
export const customCurrencyMaskConfig: CurrencyMaskConfig = {
    align: "right",
    allowNegative: true,
    allowZero: true,
    decimal: ".",
    precision: 2,
    prefix: "",
    suffix: "",
    thousands: ",",
    nullable: true,
    min: undefined,
    max: undefined,
    inputMode: CurrencyMaskInputMode.FINANCIAL
};
```

Advanced Magic xpa Projects は、以下の手順でインストールします。:

インストールが完了したら、プロジェクトをコンパイルします。

1. 生成されたプロジェクトに、デコレータのない '@Input/@Output' を持つクラスがある場合、Angular 10 は同じ内容のエラーをスローします。
2. そのクラスに対して、適切なデコレータを追加してください。例えば、クラスが Input.Output を持つ場合、@Input を追加します。

```
@Input()
export abstract class MyClass {
@Input() abstract ABC;
```

3. プロジェクト内で ViewConatinerRef を使用してビューにアクセスする場合、ツリーからビューを取得するように実装を変更する必要があります。例えば

```
(<any>this.vcRef)._view.component as XYZ
```

に変更する必要があります。

```
((<any>this.vcRef)._hostView).find(v => !isNullOrUndefined((v as XYZ)));
```

Since version: 4.8.1

システムの時間形式に関係なく、24 時間制の時間形式を使用するには

システムの時刻の所 r 式が 12 時間制に設定されていて、アプリケーションで 24 時間制の書式を使う必要がある場合、HTML の <input> タグの中に [maskito] という属性を追加することができます。

以下の手順を実行します。

1. <input> タグから type="time" と step=1 の属性を削除します。
2. HHMM 値に対して [maskito]="mg.hhmm" を追加します。
   また、コントロールに与えられたマスクに従って、HHMMSS 書式用の値 "mg.hhmmss" を使用することもできる。を使用することもできます。

属性を追加すると、HTML は以下のようになります。

時間を 24 時制で表示するには、<label> フィールドの 'magicTime' というパイプを 'magicTime24' と変更する必要があります。

```
<div>
  <mat-form-field *ngIf="mg.isRowInRowEditing(row)">
    <div>
      <input
        *ngIf="mg.isRowInRowEditing(row)"
        matInput
        [maskito]="mg.hhmmss"
        [magic]="mgc.dayTime_evening"
        [rowId]="row.rowId"
        [formControlName]="mgc.dayTime_evening"
      >
        <mgError
          [magic]=mgc.dayTime_evening
          [rowId]="row.rowId"
        >
      </mgError>
  </div>
  </mat-form-field>
      <label
          *ngIf="!mg.isRowInRowEditing(row)"
          [magic]="mgc.dayTime_evening"
         [rowId]="row.rowId"
        >
        {{mg.getValue(mgc.dayTime_evening, row.rowId) |
            magicTime24: mgc.dayTime_evening}}
        </label>
 </div>
```

サポートバージョン：4.10

日付と時刻の書式をカスタマイズするには？

Magic xpa の Web Client タスクのフォームで定義されているものとは異なる書式を指定することができます。これは、生成された HTML から mgFormat タグを削除することで可能になります。

日付書式のカスタマイズ

Magic xpa のフォーム以外の書式を使用する必要がある場合は、Angular コンポーネントから次のように変更してください。

日付の書式を変更する方法を紹介します。

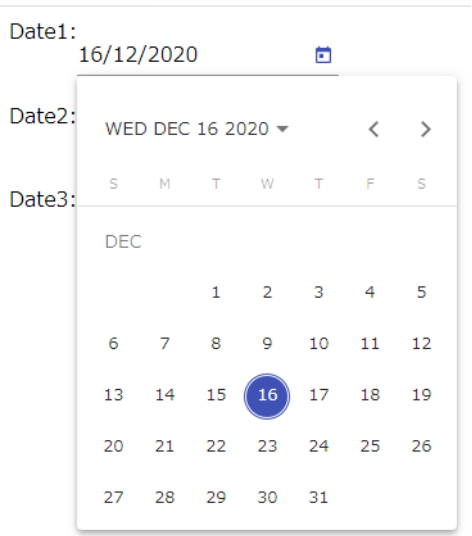
1. Test1 という名前の Web Client プログラムを作成して、日付型項目の書式をを 'MMM-DD-YYYY' に設定します。
2. アプリケーションを生成します。
3. 生成されたコンポーネントフォルダには、Test1.component.ts というファイルが作成されます。

```
export class Test1 extends TaskBaseMagicComponent {
mgc = MgControlName;
mgcp = MgCustomProperties;
mgfc:MgFormControlsAccessor;
createFormControlsAccessor(formGroup: FormGroup) {
        this.mgfc = new MgFormControlsAccessor(formGroup, this.magicServices);
    }
}
```

4. また、日付ピッカについては、Test1.component.html ファイル内の以下のコードを参照してください。

```
<input
    matInput
    [matDatepicker]="Date1"
    [magic]="mgc.Date1"
    [formControlName]="mgc.Date1"
    mgFormat
>
```

フォームの日付表示は、以下のようになります。



5. 次に、Magic xpa で指定したものとは異なる書式を使用する必要があります。フォームを使用する場合は、以下の 2 つのステップを実行する必要があります。
   a. 日付ピッカのファイル test1.component.html から mgFormat タグを削除します。.

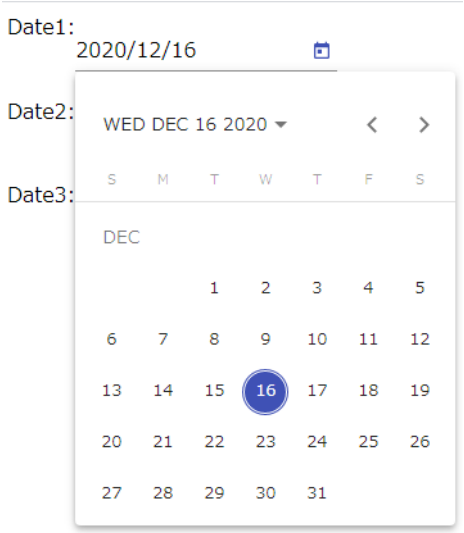
```
<input
    matInput
    [matDatepicker]="Date1"
    [magic]="mgc.Date1"
    [formControlName]="mgc.Date1"
>
```

   b. Test1.component.ts ファイル内の setInputDateFormat() メソッドを以下のようにオーバーライドします。:

```
export class Test1 extends TaskBaseMagicComponent {
   mgc = MgControlName;
```

```
    mgcp = MgCustomProperties;
    mgfc:MgFormControlsAccessor;
    createFormControlsAccessor(formGroup:FormGroup) {
        this.mgfc = new MgFormControlsAccessor(formGroup, this.magicServices);
    }
setInputDateFormat() {
    this.magicServices.task.mgInputDateFormat = "yyyy/MM/dd";
    }   //use any supported format here.
}
```

6. mgFormat タグを削除すると、以下のように結果の日付書式が表示されます。



時刻書式のカスタマイズ

日付書式と同様に、Magic xpa は Web Client タスクの時刻表示のために 2 つの書式をサポートしています。'HH:MM:SS' と 'HH:MM' です。

Magic xpa の Web Client フォームは、デフォルトのフォーマット 'HH:MM:SS' で時刻委フィールドを表示します。

時刻間の部分については、以下のような HTML が生成されます。

```
<input
    matInput
    type='time'
    step=1
    [magic]="mgc.Time1"
    [formControlName]="mgc.Time1"
>
```

時刻書式から秒 (SS) の部分を削除したい場合は、作成された HTML から ‘step=1’ を削除します。

Magic xpa フォームの時刻書式が 'HH:MM' で、秒 (SS) を追加したい場合は、HTML に "step=1" という行を追加して表示させる必要があります。

注： コンポーネントごとに単一の日付書式をサポートすることができます。これは、同じコンポーネント内で 2 つの異なる書式を表示する 2 つの日付ピッカを持つことはできないことを意味します。しかし、メインフォームとサブフォームには 2 つの異なる日付書式 8 を表示することができます。

Magic xpa Ver4.7 から、フォーム上に定義された書式がサポートされるようになりましたが、以下の日付書式はサポートされていません。

- mm、DDD、DDDD、W

setInputDateFormat() では、以下の日付書式はサポートされていません。

- mm、DDD、DDDD、W、WWW...

サードパーティのタイムピッカーを使用するには

12/24 時間制の時刻ォーマットをサポートする OS に依存しないタイムピッカーが必要になる場合があります。

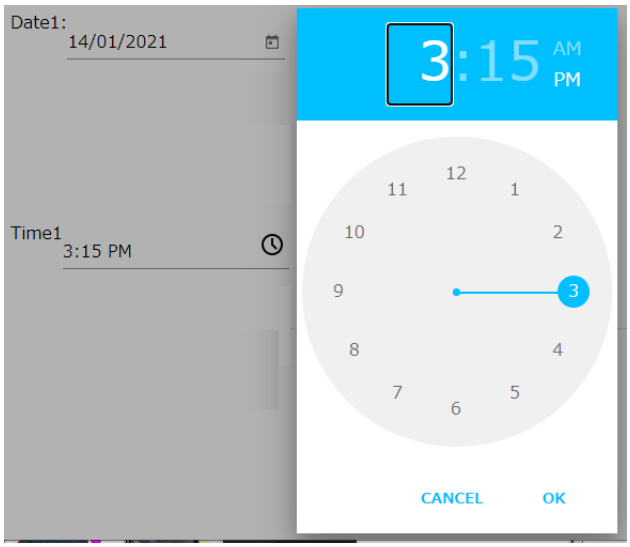
ここでは、Web Client アプリケーションでサードパーティのタイムピッカーを使用する方法を説明します。このトピックでは、'ngx-material-timepicker' という名前のタイムピッカーを使用します。

a. ngx-material-timepicker を使用する

ngx-material-timepicker を使用できるようにするために、指定された手順に従ってください。

1. Angular プロジェクトのフォルダにカレントを移動した状態で、パッケージをインストールします。
```
npm install --save ngx-material-timepicker
```
2. モジュールを magic.gen.lib.module.ts にインポートします。
```
import { NgxMaterialTimepickerModule} from "ngx-material-timepicker";
@NgModule({
  declarations: [
imports: [
// Angular Modules
  CommonModule,
  ReactiveFormsModule,
  RouterModule,
.....
 // Material Modules
  MagicAngularMaterialModule,
  NgxMaterialTimepickerModule
],
```
3. コンポーネントの html を変更します。
```
<mat-form-field
    [magic]="mgc.Time"
    [eventsOnly]=true>
    <input
       matInput
       [ngxTimepicker]="Time"
       [magic]="mgc.Time1"
       [formControlName]="mgc.Time1"
       id="Time1"
       [format]="12"
       MgControlType="MgEdit"
    >
    <ngx-material-timepicker-toggle matSuffix [for]="Time1">
    </ngx-material-timepicker-toggle>
    <ngx-material-timepicker #Time1>
    </ngx-material-timepicker>
</mat-form-field>
```

実行すると、以下のように表示されます。

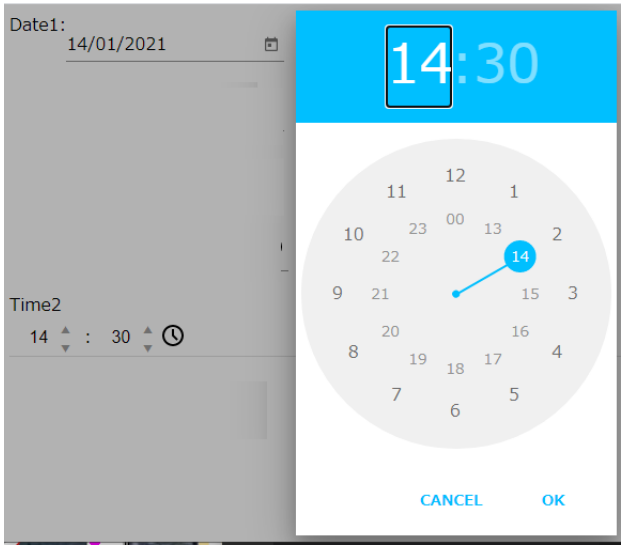


タイムピッカー : ngx-timepicker-field

以下のように、同じパッケージの ngx-timepicker-field のような別のタイムピッカーを使用することができます。

```
<div class="ngx-timepicker-field-example">
    <ngx-timepicker-field
        [magic]="mgc.Time"
        [formControlName]="mgc.Time"
        id="Time"
        [format]="24"
        MgControlType="MgEdit">
    </ngx-timepicker-field>
</div>
```

実行すると、以下のように表示されます。



*"Time" は、時刻を表示する［エディット］コントロールのコントロール名に置き換えてください。

注 : 上記のタイムピッカーを使用するには、時刻型の書式を 'HH:MM' 形式 に設定する必要があります。書式が 'HH:MM:SS' の場合でも、秒数はゼロでなければなりません。

Cookie を使用するには

Magic xpa では、Magic と Angular の Cookie 関数を使用することができます。

Magic xpa の Cookie 関数を使用する

Magic xpa では、Cookie を扱うために以下の関数が利用できます。

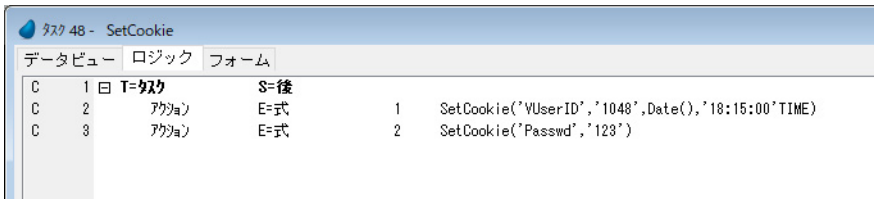
- SetCookie()
- GetCookie()
- DeleteCookie()

Web ブラウザに Cookie を設定する

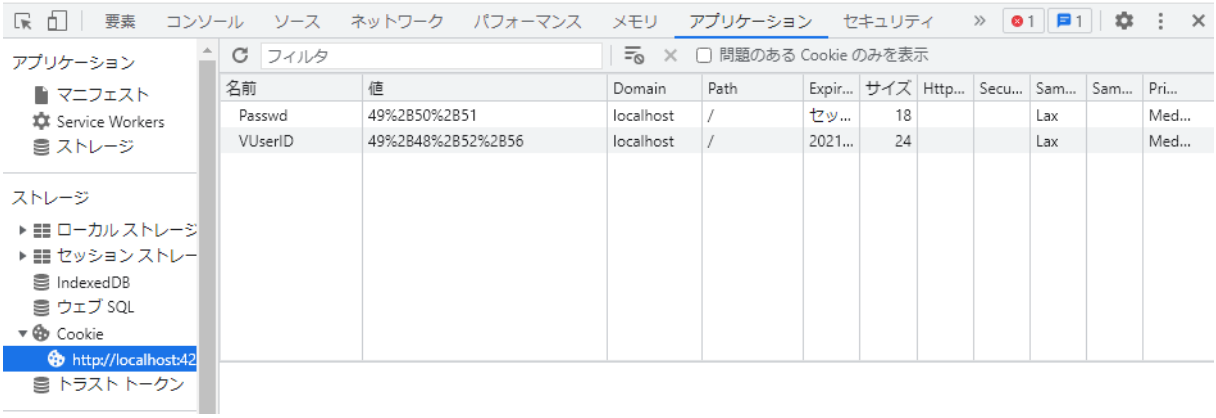
SetCookie() 関数は、Web Client プログラム内で Web ブラウザ内の Cookie を作成するために使用することができます。

この関数は、以下のように様々なパラメータを受け取ることができます。

1. Cookie に ユーザ ID を設定する場合は、SetCookie() 関数を使用します。
   `SetCookie( 'VUserID' , '1048' , Date(), ' 18:15:00' TIME)`
2. Cookie にパスワードを設定するには、以下のように定義します。
   `SetCookie( 'Passwd' , '123' )`

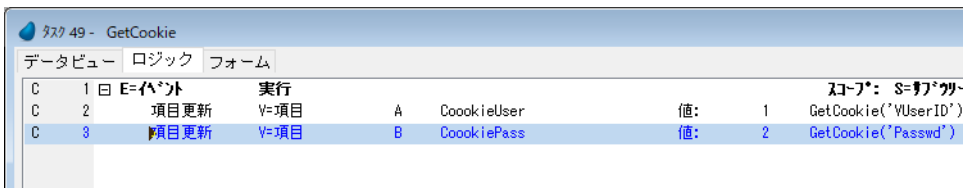


3. Web ブラウザのコンソール (F12) > Application を開くと、2 つの Cookie が設定されていることが確認できます。



Web ブラウザから Cooki を取得する

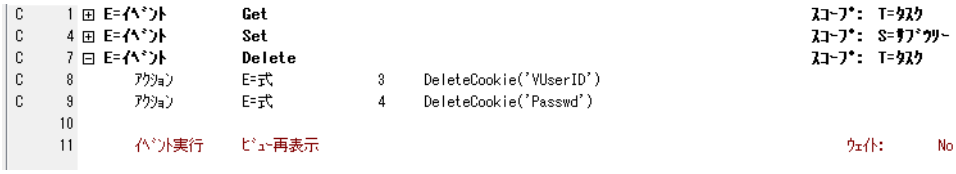
4. ［イベント］ロジックユニット内に GetCookie() 関数を定義して、ユーザ ID の Cookie を取得します。
   `GetCookie( 'VUserID' )`
5. ［イベント］ロジックユニット内に GetCookie() 関数を定義して、パスワード の Cookie を取得します。
   `GetCookie( 'Passwd' )`



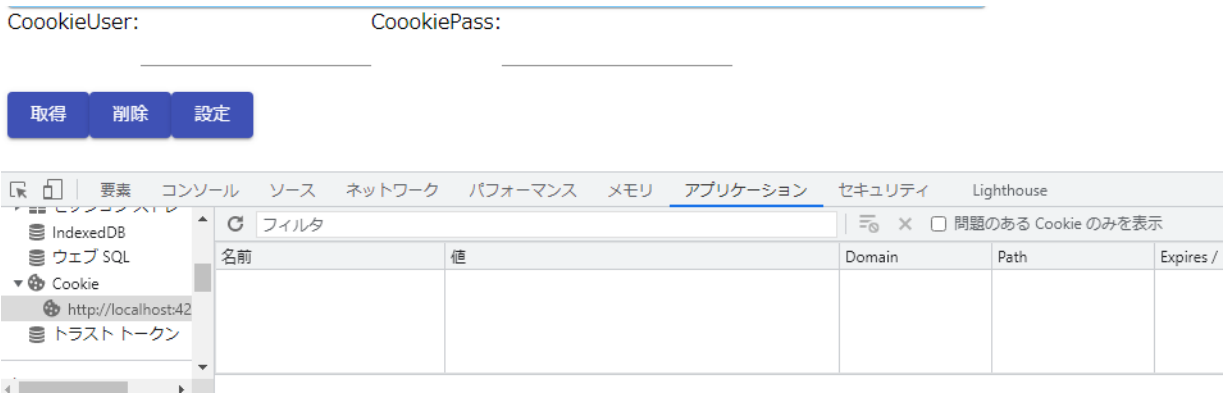
6. 両方の Cookie が取得できます。

Cookie の削除

7. ［イベント］ロジックユニット内に DeleteCookie() 関数を定義して、ユーザ ID の Cookie を削除します。
   `DeleteCookie( 'VUserID' )`
8. ［イベント］ロジックユニット内に DeleteCookie() 関数を定義して、パスワードの Cookie を削除します。
   `DeleteCookie( 'Passwd' )`



9. Web ブラウザのコンソール > アプリケーション で、2 つの Cookie が削除されていることを確認します。
10. 両方の Cookie を削除した後、GetCookie 関数を実行してください。Cookie の値が表示されなくなります。



Angular 関数を使用する

Magic xpa は Angular で Cookie を扱うために、以下の Angular 関数を提供しています。

- mg.SetCookie()
- mg.GetCookie()
- mg.DeleteCookie()

Angular 関数を定義する

Angular の TypeScript ファイルに各関数を呼び出すメソッドを定義します。

ValidateLogin.component.ts

```
xport class ValidateLogin extends TaskBaseMagicComponent {
   .......
   setCookie()
```

```
    {
        this.mg.SetCookie('VUserID','1048')
        this.mg.SetCookie('Passwd','123')
    }
    delCookie()
    {
        this.mg.DeleteCookie('VUserID');
        this.mg.DeleteCookie('Passwd');
    }
    getCookie()
    {
        var UserVal:any;
        var PassVal:any;
        UserVal = this.mg.GetCookie('VUserID');
        this.mg.setValueToControl('V_CoookieUser',UserVal,true);
        PassVal = this.mg.GetCookie('Passwd');
        this.mg.setValueToControl('V_CoookiePass',PassVal,true);
    }
}
```

メソッドを呼び出す処理を定義する

HTML ファイルにボタンをクリックすると各メソッドを呼び出す処理を定義します。

ValidateLogin.component.html

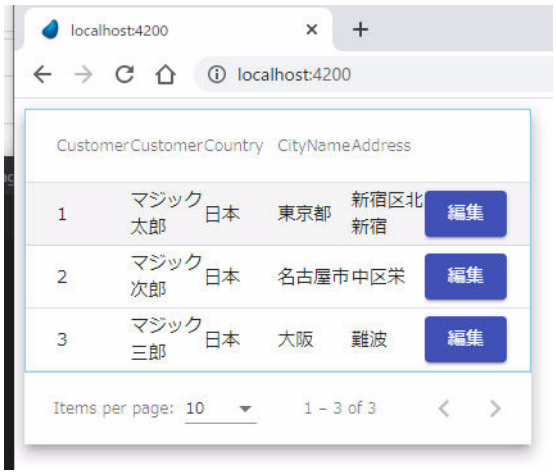
```
<div style="display: flex; flex-direction: row">
    <button
        mat-raised-button
        color="primary"
        [magic]="mgc.Set"
        (click)="setCookie()"
    >
        設定
    </button>
    <button
        mat-raised-button
        color="primary"
        [magic]="mgc.Get"
        (click)="getCookie()"
    >
      読込
    </button>
    <button
        mat-raised-button
        color="primary"
        [magic]="mgc.Delete"
        (click)="delCookie()"
    >
      削除
    </button>
</div>
```

実行結果は、Magic xpa の関数を使用する場合と同じになります。

サポートバージョン：4.8

テーブルのフッタを日本語化するには

Web Client アプリケーションで「マテリアルグリッド」に設定された［テーブル］コントロールを含んだプログラムを表示させると、以下のように表示されます。



ここでは、テーブルの下に表示されるフッタが英語表記になっています。

この表記は、AngularMaterial の Paginator を使用していますがこれを日本語化するには、以下のようにします。

MatPaginatorIntl を継承したクラスを作成する

ファイル追加する

app\shared フォルダ内に MatPaginatorIntlJa というクラスを作成します。

app フォルダ上でコマンドプロンプトを開き、以下のコマンドを実行します。

```
ng g class shared/matPaginatorIntlJa
```

mat-paginator-int-ja.ts というファイルが作成されます。

実装する

mat-paginator-int-ja.ts ファイルを以下のように定義します。

```
import { MatPaginatorIntl } from '@angular/material';
export class MatPaginatorIntlJa extends MatPaginatorIntl {
  itemsPerPageLabel = '件数';
  nextPageLabel     = '次へ';
  previousPageLabel = '戻る';
  getRangeLabel = (page: number, pageSize: number, length: number) => {
    if (length === 0 || pageSize === 0) { return `${length} 件中 0`; }
    length = Math.max(length, 0);
    const startIndex = page * pageSize;
    const endIndex = startIndex < length
      Math.min(startIndex + pageSize, length) :
      startIndex + pageSize;
    return `${length} 件中 ${startIndex + 1} - ${endIndex}`;
  }
}
```

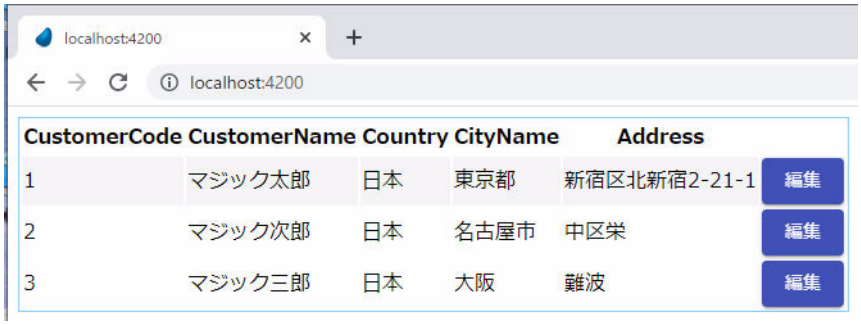
MatPaginatorIntl クラスを継承して、それぞれのラベルの値をメンバ変数またはメソッドをオーバーライドして変更します。getRangeLabel メソッドは、上記のスクリーンショットの「11 - 3 of 3」の部分の表示を行うものです。app.module.ts で読み込み、MatPaginatorModule に加え、MatPaginatorIntl モジュールをインポートし、今回作成した多言語化用クラスを次 @NgModule の providers に次のように追加します。

```
import { MatPaginatorModule, MatPaginatorIntl } from '@angular/material';
import { MatPaginatorIntlJa } from './shared/mat-paginator-int-ja';
...
```

```
@NgModule({
...
providers: [
    { provide: MatPaginatorIntl, useClass: MatPaginatorIntlJa }
  ],
```

実行結果

ビルドして実行すると、以下のように表示されます。



レコードのフォーカスを目的の行に更新するには

Magic xpa Ver4.9 までは、マテリアルデザインの［テーブル］コントロールが定義されたフォームに作成された #Paginator で pageSizeOptions を使用してページを変更した場合、選択したレコードがページネーション後のページの行にフォーカスが変更されませんでした。レコードのフォーカスは、マテリアルデザインテーブルの同じ選択された行に留まっていました。これは正しい動作ですが、レコードのフォーカスを任意の行に変更することも可能です。

ページネーションでレコードのフォーカスを任意の行に更新するために、tableService.mgOnPaginateChange() というサービスに、changeSelectedRow という新しいオプションパラメータを指定できる受け入れるようになりました。生成された html でこのパラメータを 'true' に設定すると、フォーカスは目的の行に更新されます

```
<mat-paginator
          #paginator
          [pageSize]="10"
          [pageSizeOptions]="[5, 10, 20]"
          (page)="tableService.mgOnPaginateChange($event, true)"
  >
  </mat-paginator>
```

キーボードを使用してマテリアルデザインのテーブルを操作するには

gUp キー、PgDn キー、Ctrl+Home キー、Ctrl+End キー、および 矢印キーを使用して、Web Client アプリケーションのマテリアルデザインテーブルを移動することができます。

以下の手順に従ってください：

1. row-with-keyboard-selection.directive.ts ファイルを src\app\magic フォルダにコピーします。
2. src\app\magic\magic.gen.lib.module.ts を以下のように修正します。
   a. 以下のように RowWithKeyBoardSelectionDirective の Import 定義を追加します。
   ```
   import {RowWithKeyBoardSelectionDirective} from "./row-with-keyboard-selection.directive";
   ```
   b. 以下のように CUSTOM_ELEMENTS_SCHEMA の Import 定義を追加します。
   ```
   import {CUSTOM_ELEMENTS_SCHEMA, NgModule, NgModuleRef} ?from '@angular/core';
   ```
   c. 'declarations' セクション内に RowWithKeyBoardSelectionDirective を追加します。
   ```
   @NgModule({
      declarations: [
         ...magicGenComponents,
         RowWithKeyBoardSelectionDirective
      ],
   ```
   d. 'exports' セクション内に RowWithKeyBoardSelectionDirective を追加します。
   ```
   exports: [
      ...magicGenComponents,
      RowWithKeyBoardSelectionDirective,
      MagicModule
      ],
   ```
   e. 'providerss' セクション内に RowWithKeyBoardSelectionDirective を追加します。
   ```
   providers: [
      ExitMagicService, RowWithKeyBoardSelectionDirective
      ...
   ],
   ```
   f. 'providers' セクション内に以下のように 'schemas' を追加します。
   ```
   schemas: [
      CUSTOM_ELEMENTS_SCHEMA,
   ]
   ```
   g. テーブルを含む生成された Web Client プログラムでは、以下のように HTML の mat-row 内にディレクティブを使用する必要があります：。
   ```
   <mat-row
      *matRowDef="let row; columns: displayedColumns;"
         class="table_row"
         [ngClass]="{ 'selected': selection.isSelected(row)}"
         [magicRow]="row?.rowId"
         (click)="row? tableService.selectRow(row.rowId) : null"
         [rowWithKeyBoardSelection]="row?.rowId"
         >
   </mat-row>
   ```

推奨事項

新しい rowWithKeyBoardSelection ディレクティブを使用する場合、Paginator を使用して Mat テーブルをトラバースする際に、新しいページの行にフォーカスを移す機能を使用することをお勧めします。

mat-paginator 要素の tableService.mgOnPaginateChange() に 2 番目のパラメータを 'true' として渡すことで実現できます。

```
<mat-paginator
#paginator
[pageSize]="10"
[pageSizeOptions]="[5,10,20]"
(page)="tableService.mgOnPaginateChange($event, true)"
> </mat-paginator>
```

サポートバージョン：4.10

Web Client のページをスタマイズを行うには

Magic xpa でビルドされる Web Client のページをカスタマイズする方法について説明します。

タイトルバーの表示

app.module.ts を修正する

src\app\app.module.ts に太字を追加します。

```
import { MagicRoutingModule } from './app.routes';
import { MatToolbarModule } from '@angular/material';
@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    BrowserAnimationsModule,
    ReactiveFormsModule,
    MagicModule.forRoot(),
    MagicAngularMaterialModule,
    MagicGenLibModule,
    MagicRoutingModule,
    MatToolbarModule
  ],
```

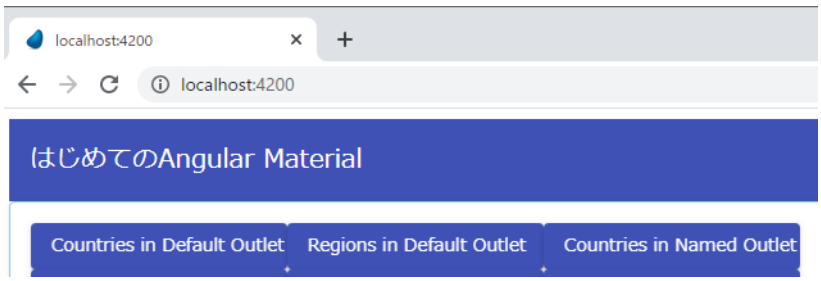
app.component.html を修正する

src\app\app.component.html に太字を追加します。

```
<div>
  <mat-toolbar color="primary">
    <span>はじめての Angular Material</span>
  </mat-toolbar>
  <magic-root></magic-root>
</div>
```

実行結果

以下のように表示されます。



ダッシュボードの実装

Angular Material のコンポーネントを作成する

src\app フォルダ上で次のコマンドを実行します。

サイドナビを作成します。

```
ng generate @angular/material:nav nav
```

ダッシュボードを作成します。

```
ng generate @angular/material:dashboard dashboard
```

コンポーネントを組み合わせて画面を作成する

src\app.component.html を以下のようにします。(<magic-root></magic-root> は、nav.component.html に移動させます)

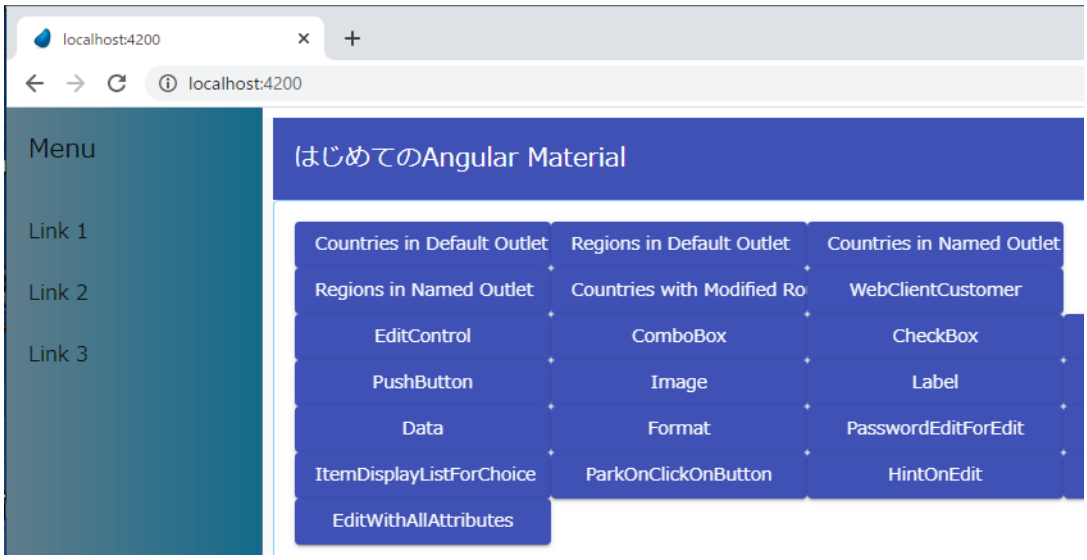
```
<app-nav>
  <app-dashboard></app-dashboard>
</app-nav>
```

src/app/nav/nav.component.html を太文字の処理を修正します。" routerLink" には、ルート名を割り当てます。

```
<mat-sidenav-container class="sidenav-container">
  <mat-sidenav #drawer class="sidenav" fixedInViewport
      [attr.role]="(isHandset$ | async) ? 'dialog' : 'navigation'"
      [mode]="(isHandset$ | async) ? 'over' : 'side'"
      [opened]="(isHandset$ | async) === false">
    <mat-toolbar>Menu</mat-toolbar>
    <mat-nav-list>
      <a mat-list-item routerLink="EditControl" >Link 1</a>
      <a mat-list-item routerLink="Data">Link 2</a>
      <a mat-list-item routerLink="CheckBox">Link 3</a>
    </mat-nav-list>
  </mat-sidenav>
  <mat-sidenav-content>
    <mat-toolbar color="primary">
      <button
        type="button"
        aria-label="Toggle sidenav"
        mat-icon-button
        (click)="drawer.toggle()"
        *ngIf="isHandset$ | async">
        <mat-icon aria-label="Side nav toggle icon">menu</mat-icon>
      </button>
      <span>はじめての Angular Material</span>
    </mat-toolbar>
    <magic-root></magic-root>
  </mat-sidenav-content>
</mat-sidenav-container>
```

実行結果

ビルドして実行すると、以下のように表示されます。



参照： サイドナビの色などを変更する場合は、nav.componet.css を修正してください。

区切り線の表示

水平方向の区切り線を表示させる場合は、html ファイルに以下を追加します。

```
<mat-divider></mat-divider>
```

背景色の設定

Angular のページ全体の背景色を設定する場合は、以下のように設定します。

app.component.ts を以下のように修正します。

```
import {Component, ViewEncapsulation} from '@angular/core';
@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
  encapsulation: ViewEncapsulation.None
})
export class AppComponent {
  title = 'WebSamples';
}
```

app.component.css に以下のように背景色を追加します。

```
body {
    background-color: green;
  }
```

これで、実行されるページのすべての背景色が緑色に表示されます。

背景画像の設定

Angular のページ全体の背景となる画像を設定する場合は、以下のように設定します。

app.component.html を以下のように修正します。

```
<div [ngStyle]="{'background' : 'url(./assets/images/Sample.jpg)'} " >
  <magic-root> </magic-root>
</div>
```

これで、実行されるページの背景画像が表示されます。この例では、"src\assets\images" フォルダ内の "Sample.jpg" が背景となります。

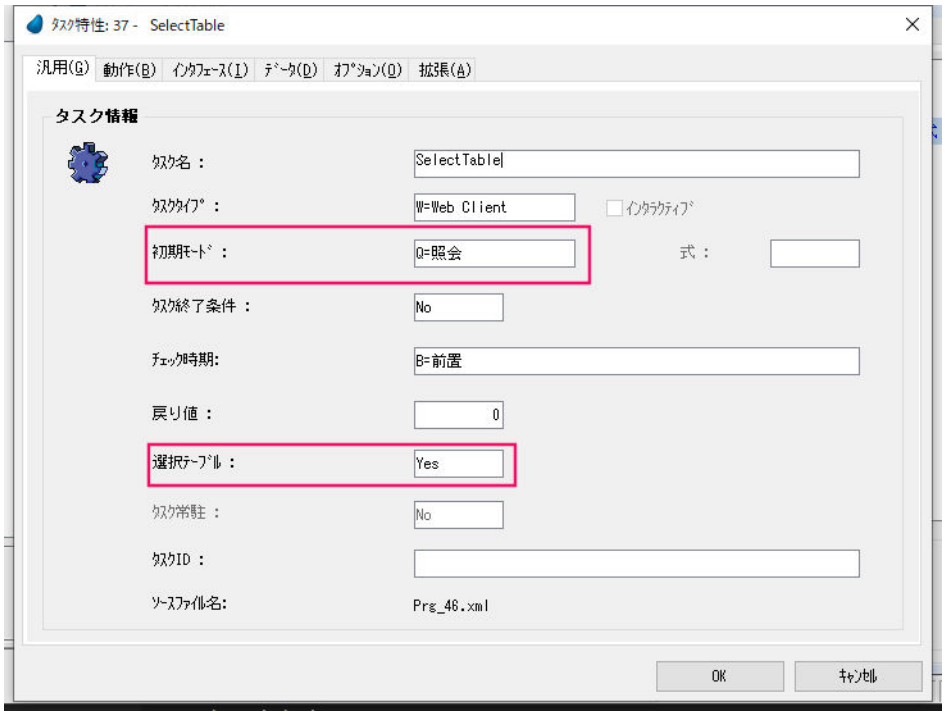
選択プログラムを作成するには

オンラインプログラムと同じように Web Client プログラムでも選択用のプログラムを作成することができます。

基本的には、オンライン /RIA の場合と同じように作成しますが、Web Client プログラムとしての違いを含めて説明します。

選択用プログラムを作成する

1. ［プログラム］リポジトリで、F4 を押下し一行追加します。
2. プログラムに名前を入力します。この名前には半角の英数字のみを使用するようにしてください。

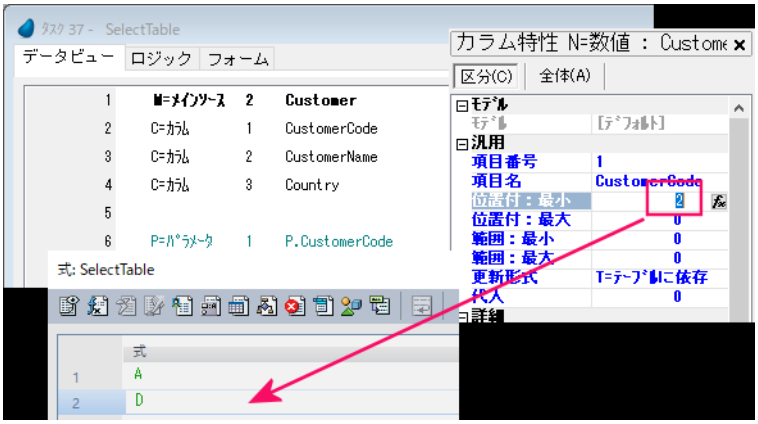


3. ［名前］カラムで ズーム （ F5 ）します。新規プログラムとなるため、［タスク特性］ダイアログが表示されます。ここで以下のパラメータを設定します。

- 初期モード …… 照会
- 選択テーブル …… Yes

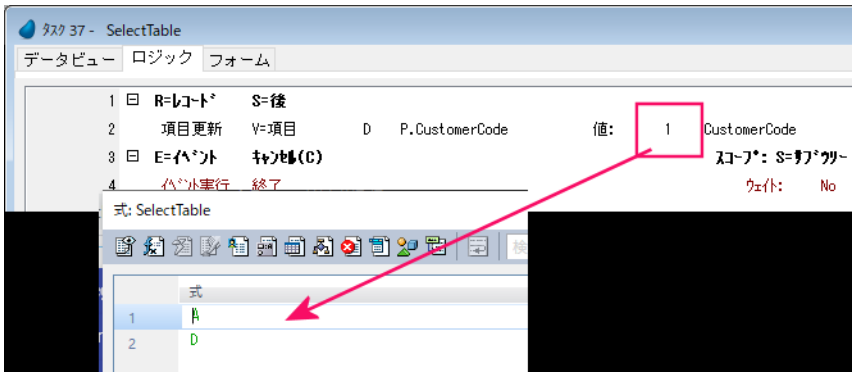
選択処理ではデータの修正を行わないため、照会モードに設定してください。

［選択テーブル］特性は、エンジンの動作を変更します。「Yes」に設定された場合、［選択］イベントが発生すると［レコード後］を実行しタスクが終了します。



4. ［データビュー］エディタ では、一覧表示させたいデータソース（ここでは、"Customer"）を指定します。［データソース］特性では、アクセス中のレコードがロックされないように、［アクセス］特性を「R= 読込」に設定します。
5. また、パラメータ項目を 1 つ定義します。このパラメータは、起動元に返す値になります。この例では、顧客番号として 9 桁の数値型項目が使用されています。間違った項目を更新しないように項目の名前の付け方を工夫してください。この場合、パラメータであることを表すため、接頭辞「P.」が使用されています。

6. データソースの［位置付：最小］特性にパラメータ項目を設定します。この例では、データソースの CustomerCode カラムを位置付けるためにパラメータの「P.CustomerCode」を設定しています。

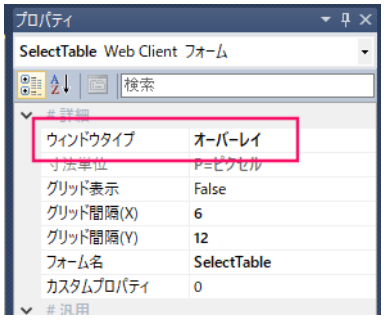


7. ［ロジック］エディタ 内では、［レコード後］ロジックユニットを作成します。この ロジックユニット では、データソースの値でパラメータを更新します。これは選択された値を起動元に返すための処理です。
8. ［キャンセル］イベントによる［イベント］ロジックユニットも作成します。この ロジックユニット では、［イベント実行］処理コマンドを定義して［終了］イベントを割り当てます。［キャンセル］イベントが発行されたら選択処理を行わずにタスクを終了するための処理になります。

フォームを作成する

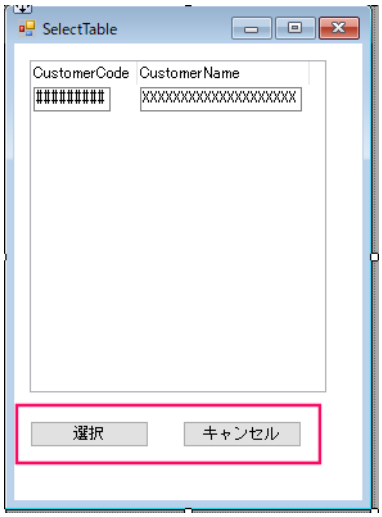
1. 最後に、フォームを作成します。これは簡単な［テーブル］コントロールを使用して作成します。
2. フォームジェネレータを使用してラインモードのフォームを作成します。作成手順は、「デフォルトのフォームレイアウトを自動的に作成するには」を参照してください。

フォームの［ウィンドウタイプ］プロパティを「オーバレイ」に設定します。これによって、ポップアップ表示になります。



フォームに［ボタン］コントロールを２つ配置します。各々、以下の内部イベントを割り当てます。

- 選択
- キャンセル



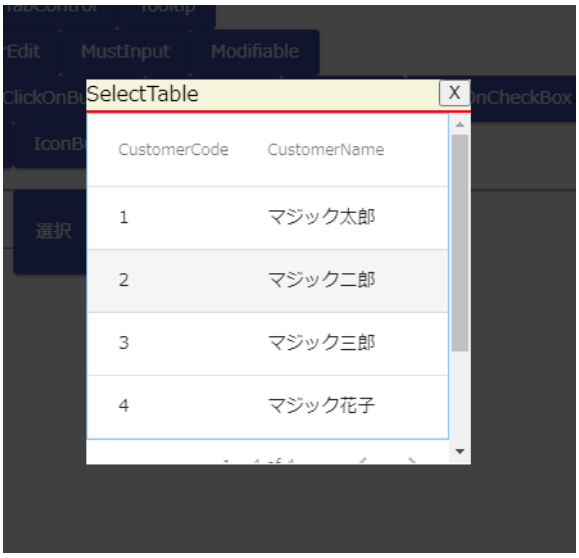
選択プログラムを使用する

Web Client の選択プログラムは、Web Client からでしか呼び出せません。[コールプログラム] 処理コマンドで呼び出します。その際、パラメータを明示的に指定する必要があります。暗黙のうちに渡されるようになります。

動作を確認する

プログラムをもとに、Angular コードをビルドして確認してみます。次のような選択プログラム画面が表示されます。

ウィンドウの高さが足りず、ボタンが表示されていません。



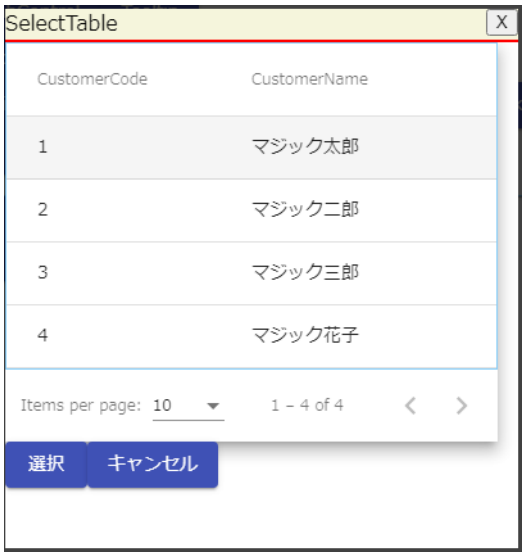
フォームをカスタマイズする

フォームの幅と高さを拡張する

プログラム（SelectTable）によって作成された、SelectTable.ts ファイルを以下のように修正します。

```
export class SelectTable extends BaseMatTableMagicComponent implements MagicModalInterface {
    mgc = MgControlName;
    mgfc: MgFormControlsAccessor;
    createFormControlsAccessor(formGroup: FormGroup) {
        this.mgfc = new MgFormControlsAccessor(formGroup, this.magicServices);
    }
    private static readonly formName: string = "SelectTable";
    private static readonly showTitleBar: boolean = true;
    private static readonly x: number = 0;
    private static readonly y: number = 0;
    private static readonly width: string = "400px";
    private static readonly height: string = "420px";
    private static readonly isCenteredToWindow: boolean = true;
    private static readonly shouldCloseOnBackgroundClick = true;
```

実行すると以下のように表示が変わります。



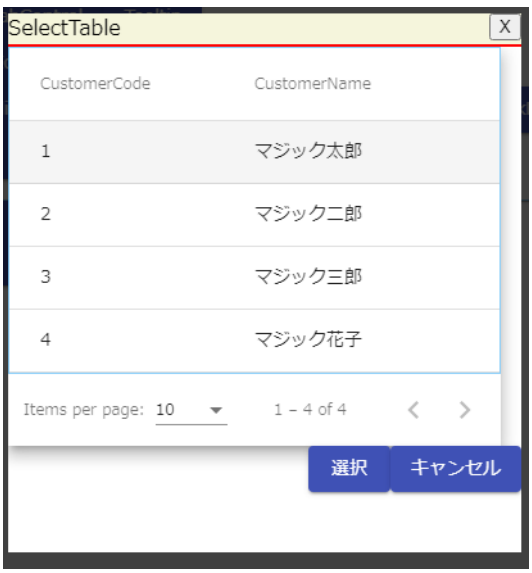
ボタンの位置を右端に移動する

プログラム（SelectTable）によって作成された、SelectTable.html ファイルを以下のように修正します。

"justify-content:flex-end" は、オブジェクトを末尾に寄せるように指定します。

```
<div style="display: flex; justify-content: flex-end;">
    <button
        mat-raised-button
        color="primary"
        magic="{{mgc.Button10}}"
    >
        選択
    </button>
.......
</div>
```

実行すると以下のように表示が変わります。



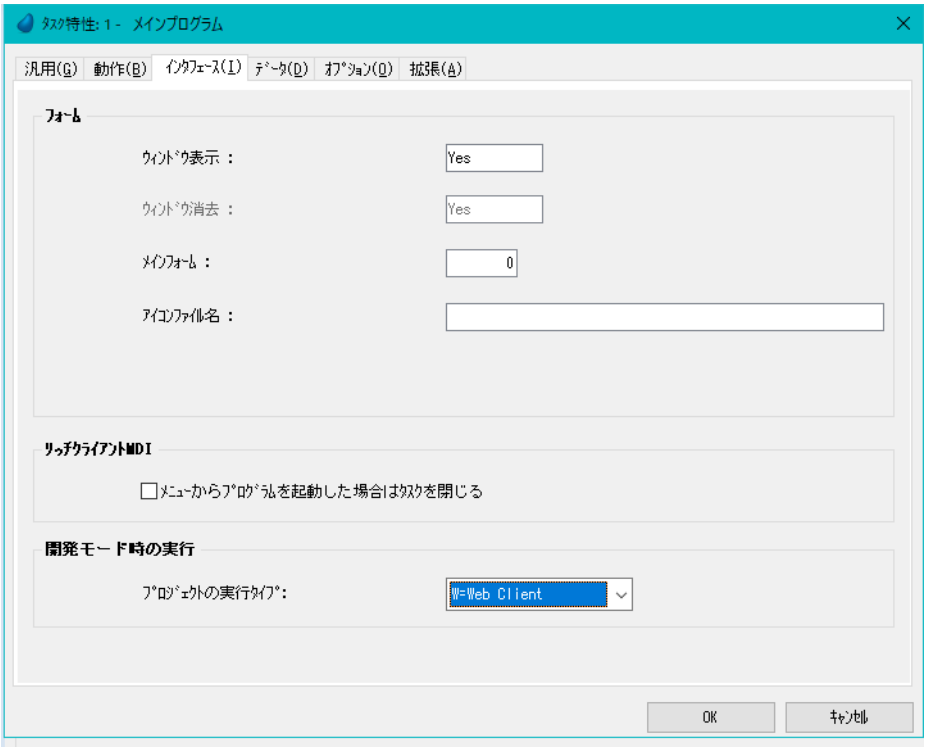
このように実行結果を確認しながら調整するようにします。

SmartUX Studio を使用して Web Client プロジェクトを作成するには

Magic xpa と SmartUX の統合により、Web Client プロジェクトを新規作成したり、既存の Web Client プロジェクトを SmartUX で開くことができるようになりました。

最初にメインプログラムの［タスク特性］ダイアログを開き、［インタフェース］タブの［開発モード時の実行／プログラムの実行タイプ］を「W=Web Client」に設定します。

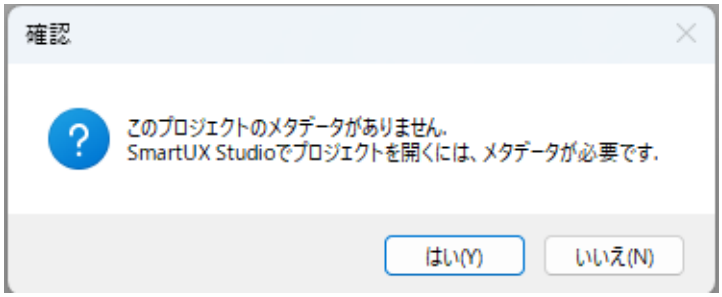
これによって、SmartUX のサービスが起動されます。



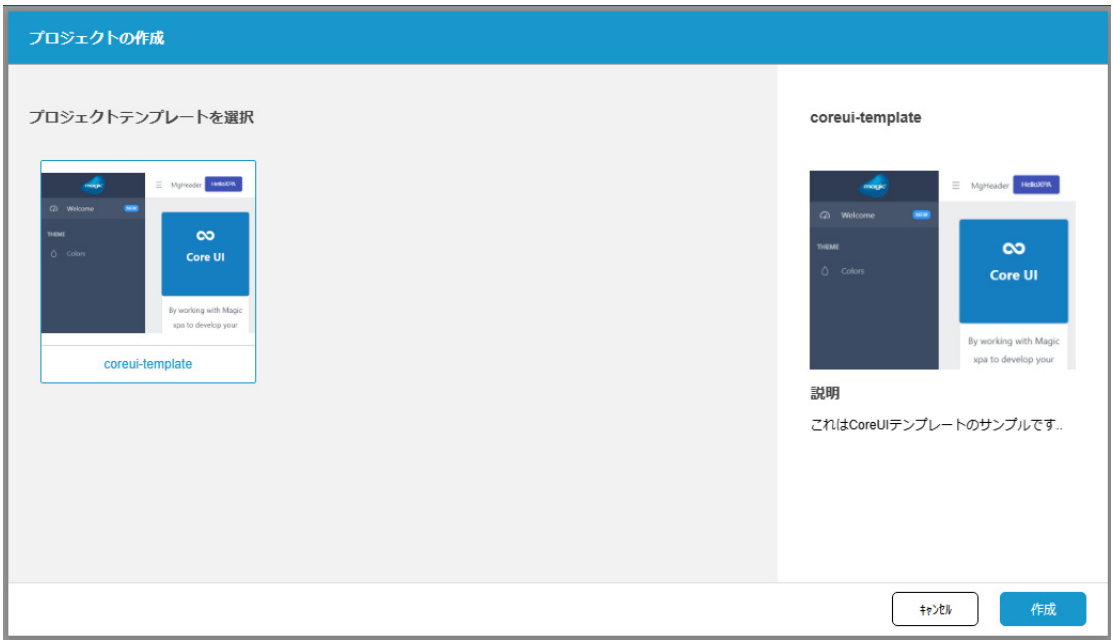
Web Client プロジェクトを新規作成する

Magic xpa で作成され、まだ生成されていない Web Client プログラムがあり、［Web］メニューの［SmartUX Studio で開く］をクリックします。

Input フォルダがまだ作成されていない場合、以下のメッセージが表示されます。



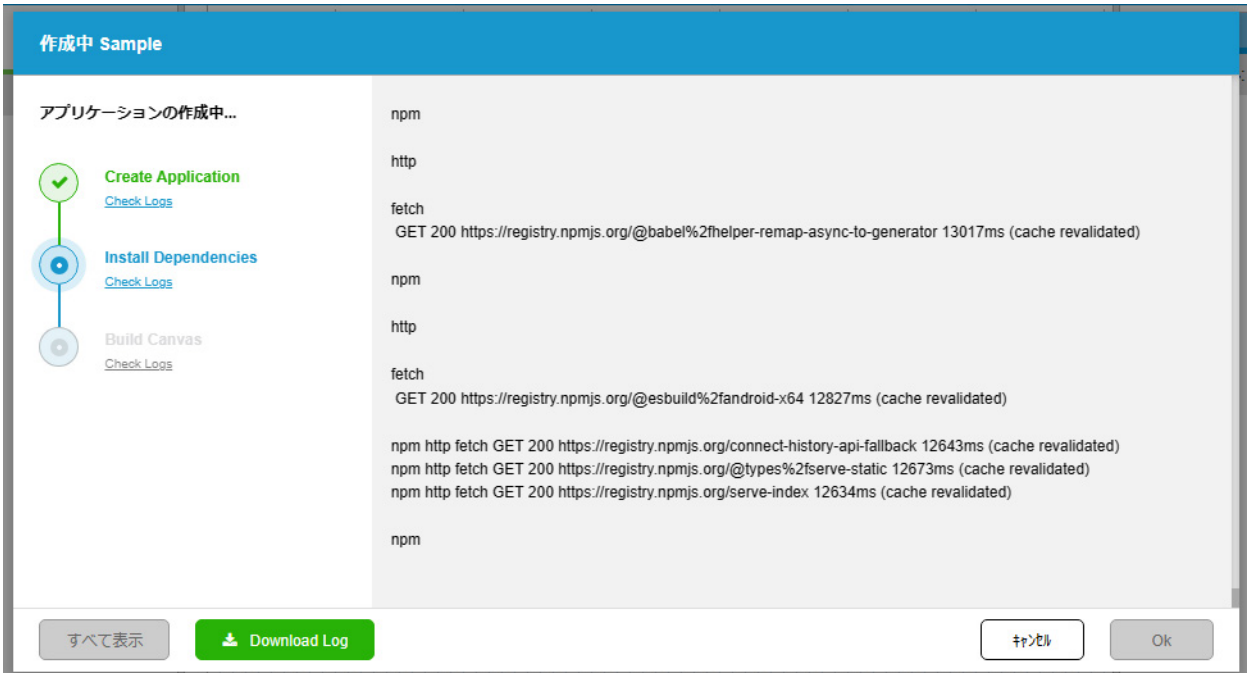
［はい］をクリックすると、Web ブラウザ上で SmartUX Studio が起動し以下のような［プロジェクト作成］ダイアログを表示します：



［プロジェクト作成］イアログには、用意されたテンプレートが表示されます。

1. 必要なテンプレートを選択します。
2. ［Create］をクリックします。

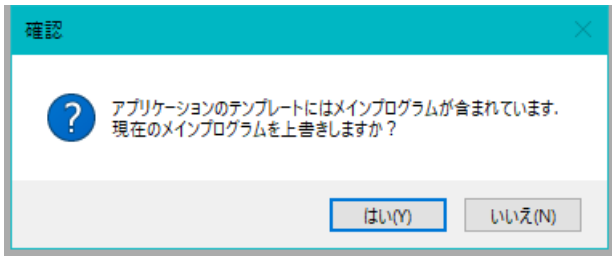
作成処理が実行します。



Magic xpa Studio 側では、プロジェクトが実行状態になり、メインプログラムを上書きするかどうかを確認するダイアログが表示されます。

ここで、「はい」をクリックします。

プログラムが追加され、プロジェクトが実行状態になります。



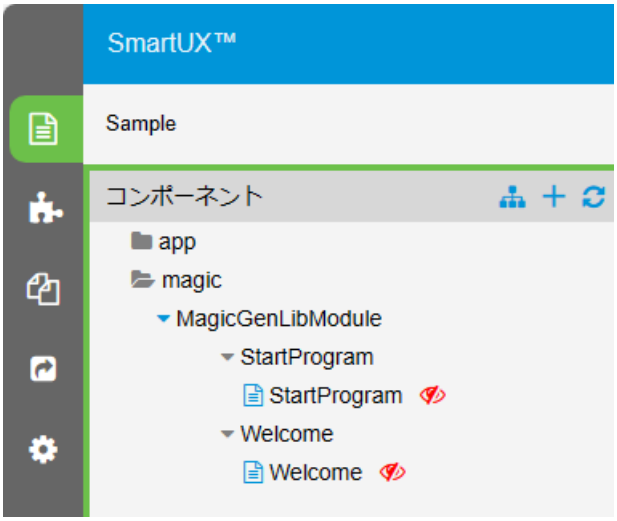
注： 上記のダイアログは、アプリケーションの作成中に、「はい」をクリックしてください。

作成処理が終了したら、［OK］ボタンが有効になります。ボタンをクリックしてウィンドウを閉じます。

これで、SmartUX Studio にプロジェクトが作成されます。

テンプレートとして "Corer-nabigation" を選択した場合、デフォルトで次の２つのプログラムが作成されます。

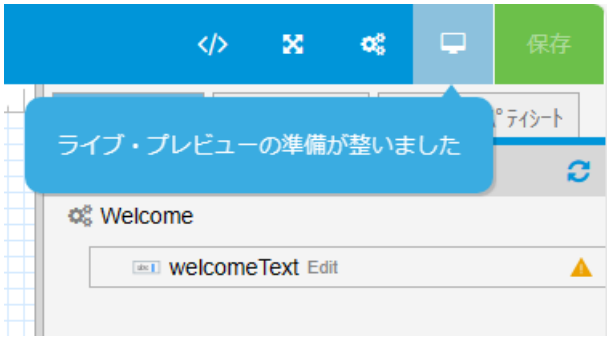
- StartProgram
- welcome



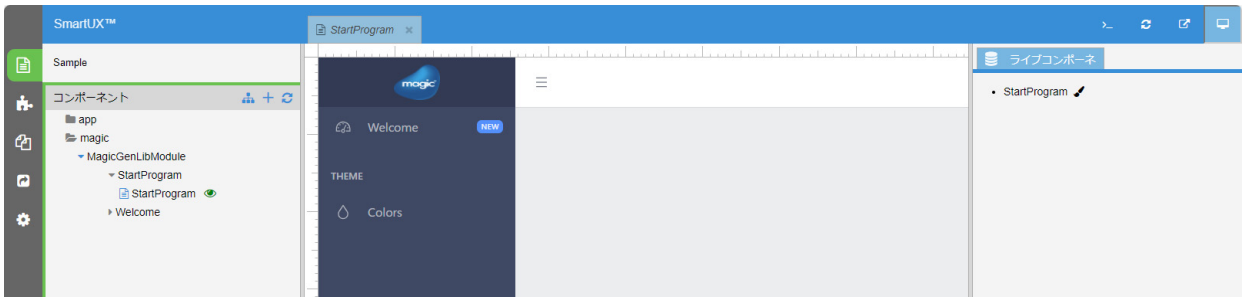
SmartUX の画面

Magic xpa Studio の［プログラム］リポジトリ

右上の［Live Preview is re4ady］をクリックします。

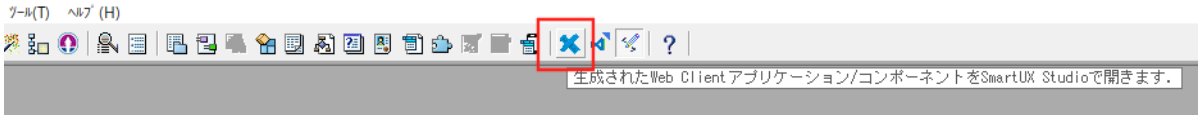


中央のペインに Web Client アプリケーションの実行結果が表示されます。

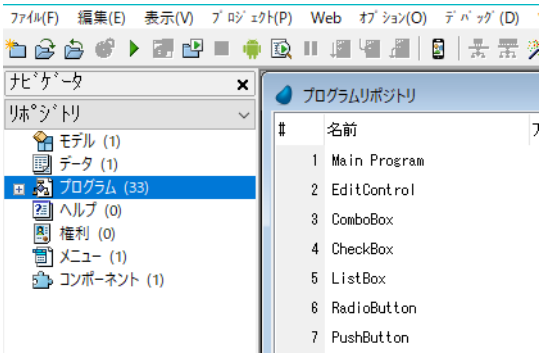


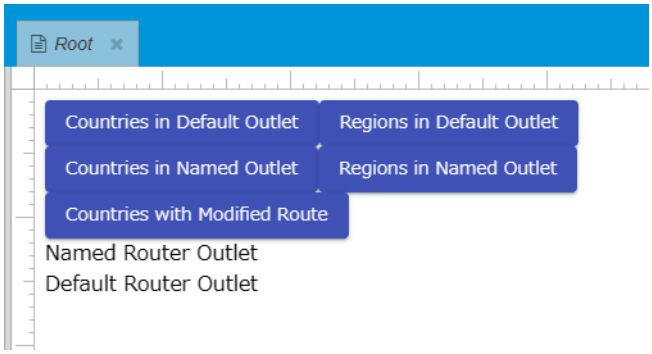
Magic xpa で作成された既存の Angular プロジェクトを SmartUX Studio で開く

Magic xpa で作成された Web Client プロジェクトがある場合、プロジェクトを開いた状態で［Web］メニューの［SmartUX で開く］(Ctrl+Shift+X) をクリックするか、以下のアイコンをクリックすることで、SmartUX Studio で開くことができます。



"Input" フォルダのファイルからインポートされます。



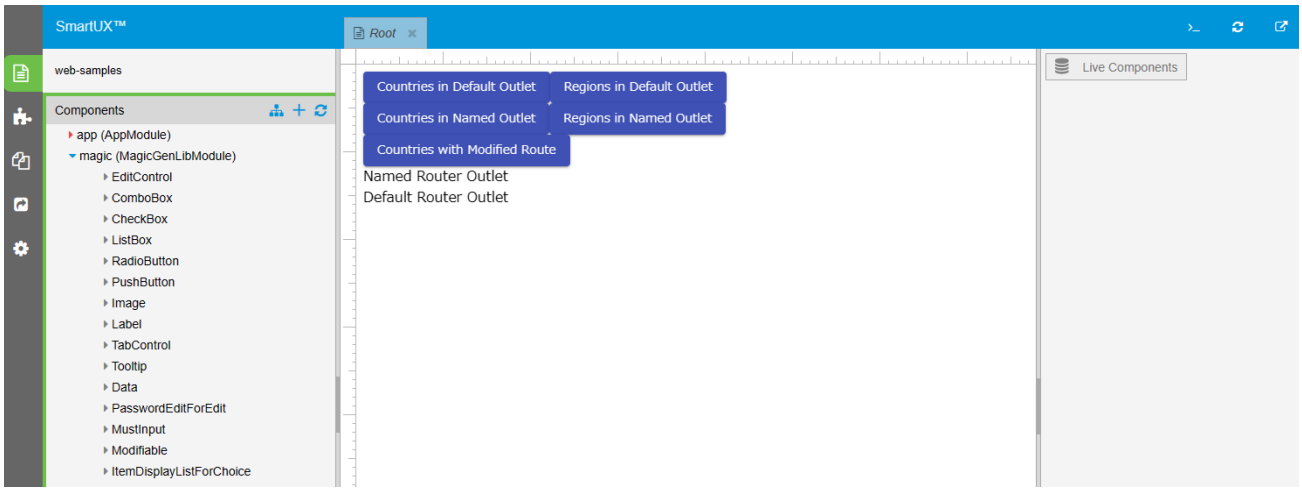


Magic xpa Studio の［プログラム］リポジトリ
SmartUX の画面

mg server での実行結果

SmartUX Studio を開くと、下図のように ［サイドバー］> コンポーネント に "Application Modules" と "Magic xpa Modules" が表示されます。

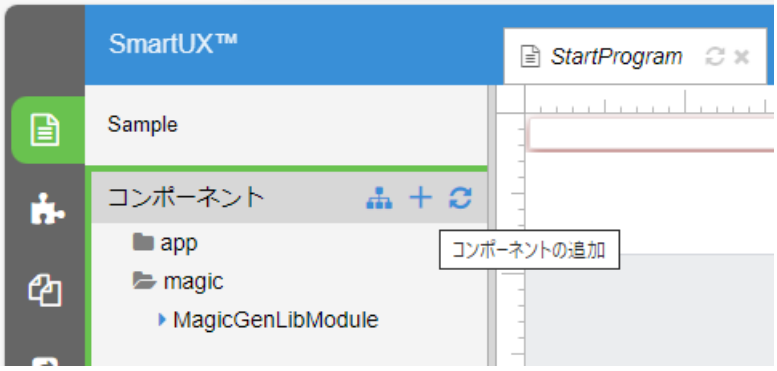
" ライブプレビュー " をクリックすることで動作を確認することができます。


SmartUX™
Root
web-samples
Components
app (AppModule)
magic (MagicGenLibModule)
EditControl
ComboBox
CheckBox
ListBox
RadioButton
PushButton
Image
Label
TabControl
Tooltip
Data
PasswordEditForEdit
MustInput
Modifiable
ItemDisplayListForChoice
Countries in Default Outlet
Regions in Default Outlet
Countries in Named Outlet
Regions in Named Outlet
Countries with Modified Route
Named Router Outlet
Default Router Outlet
Live Components

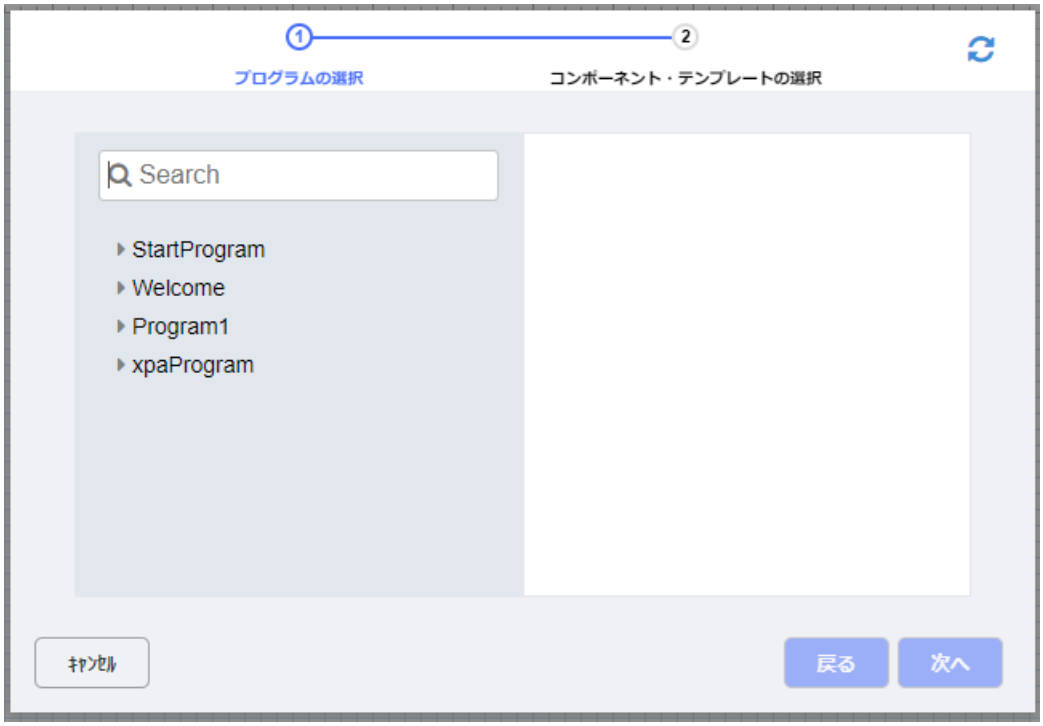
Magic xpa プログラムを SmartUX Studio に追加し、ライブプレビューで確認するには

Magic xpa で新しいプログラムを作成し、SmartUX に同じプログラムを追加したい場合は、以下のように行ってください。

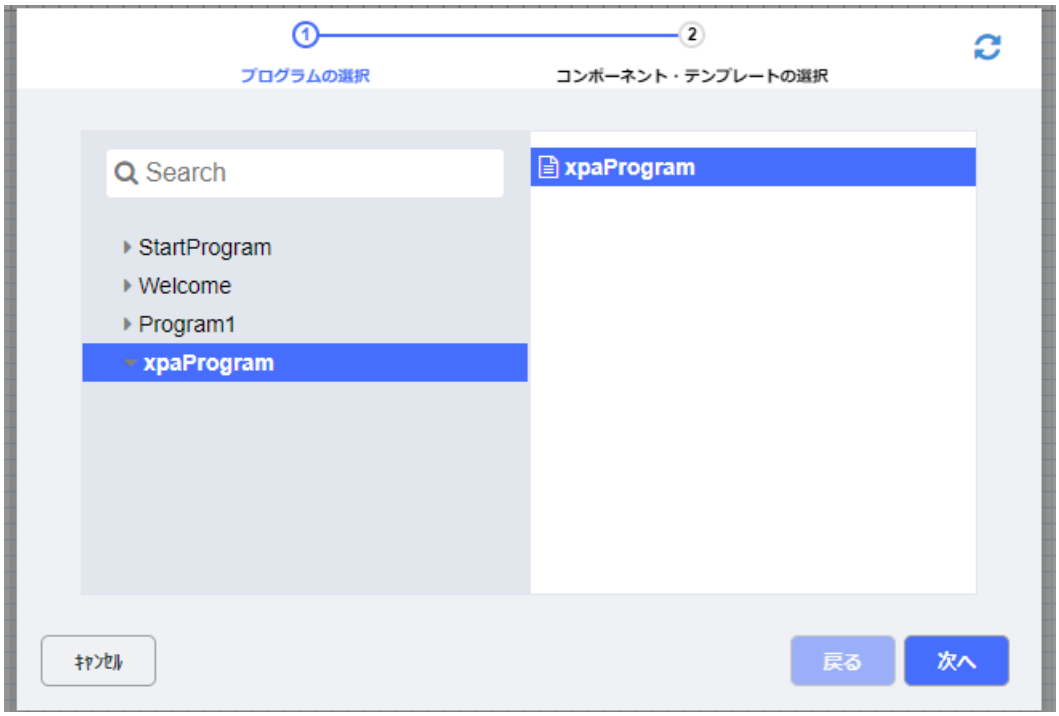
1. Magic xpa で "xpaProgram" という名前の新しい Web Client プログラムを作成します。
2. APG を使用して スクリーンモードで作成します。
3. プロジェクトを実行します (Ctrl+F7)。
4. SmartUX UI に切り替えます。
5. " コンポーネント " の隣に表示されている '+' (コンポーネントの追加) ボタンをクリックします。



下図のように、左ペインの "Select Program ム " のステップで「xpaProgram」が追加されていることを確認することができます。



新しく追加された "xpaProgram" をクリックします。右ペインに表示されます。



［次へ］をクリックします。

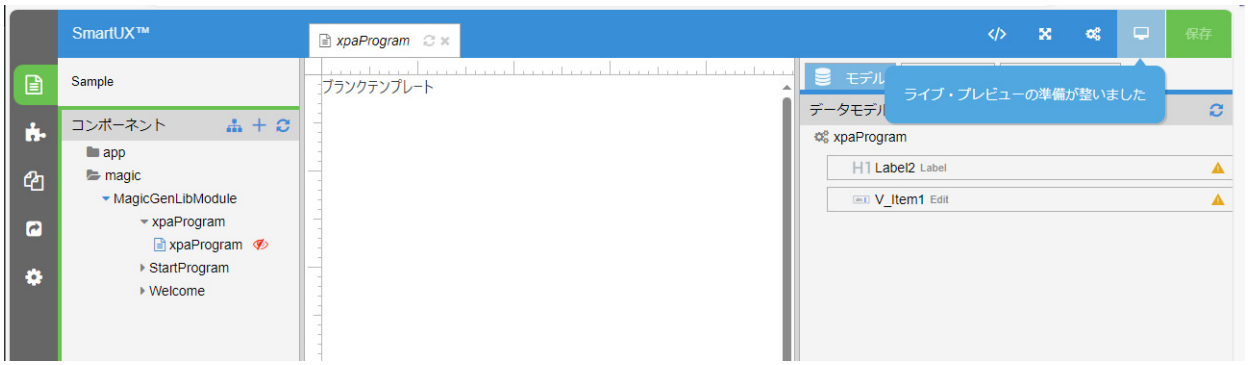
追加したプログラム（Angular コンポーネント）を選択します。SmartUX Studio では、コンポーネントのテンプレートを選択するためのダイアログが表示されます。

Magic xpa で生成されたコンポーネントを使用するか、様々なテンプレートからコンポーネントを選択することができます。

作成コンポーネント	コンポーネントテンプレート
Blank	Blank
Forms	CardView
	Registration
Login	CustomLogin
	VanillaLogin

テンプレートを選択したら、［終了］をクリックします。

コンポーネント > magic (MagicGenLibModule) の下にコンポーネントが追加されていることを確認します。



これで、このコンポーネントをライブプレビューで実行することができます。実行するには、以下の手順に従ってください：

1. "xpaProgram" を［Web アプリケーションの設定／スタートアッププログラム］として設定します。
2. SmartUX UI に切り替えます。
3. " ライブプレビューの準備が整いました " をクリックします。

コンポーネントが生成され、モックデータがコンポーネントで利用可能になっていることが確認することができます。

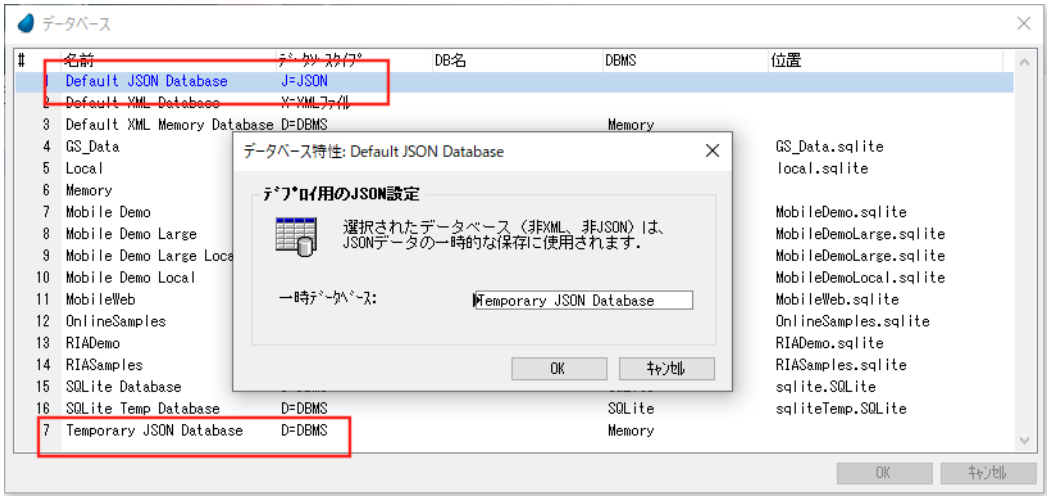
サポートバージョン：4.10

第 44 章 : JSON

JSON を処理するには

以下のようなプログラムを作成することで、JSON の構造体の読込、変更、更新を行うことができます。

1. ［データベース］テーブルに「Default JSON Database」という JSON タイプの項目がデフォルトで定義されています。JSON データベースのタイプを使用することで JSON データソースを作成することができます。

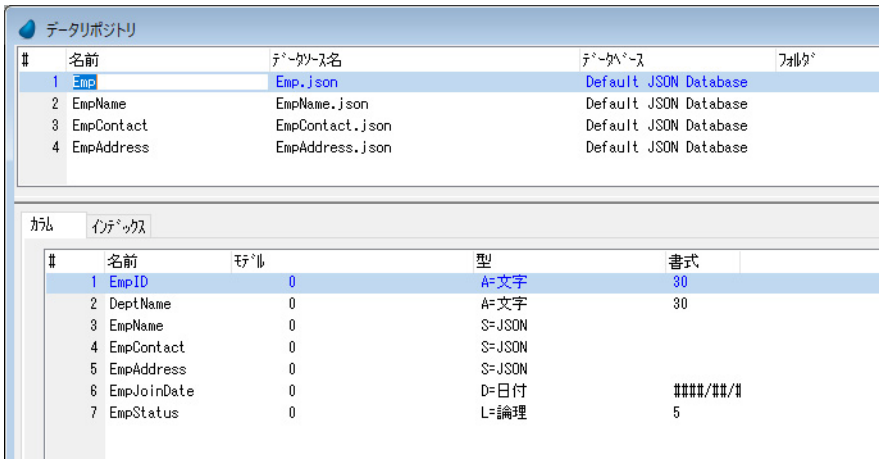


"Default JSON Database" の ［データベース］特性には 「Temporary JSON Database」というメモリデータベースが定義されています。

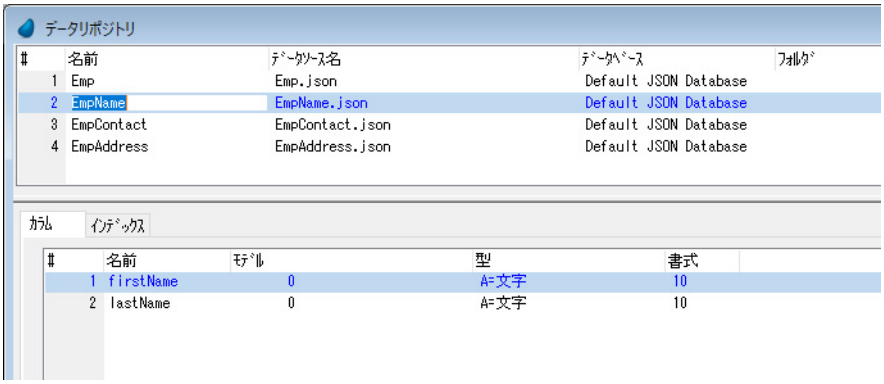
2. ［データソース］リポジトリに、以下のように "Emp"、"EmpName"、"EmpContact"、"EmpAddress" のデータソースを定義します。［データベース］には、「Default JSON Database」を設定します。

Emp データソース

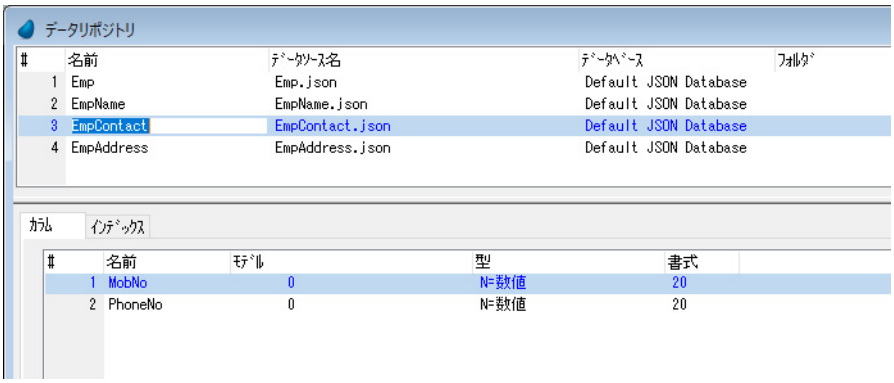
"Emp" データソースには、"EmpName"、"EmpContact"、"EmpAddress" のカラムを定義し、データ型に「S=JSON」を設定します。



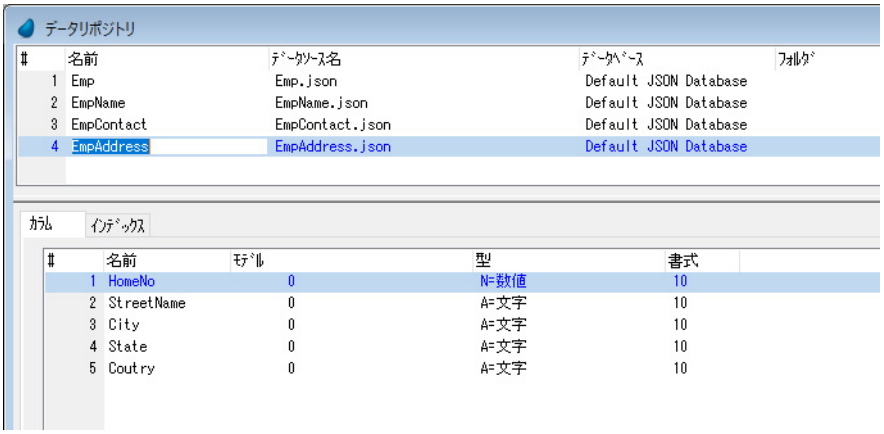
EmpName データ ソース



EmpContact データ ソース

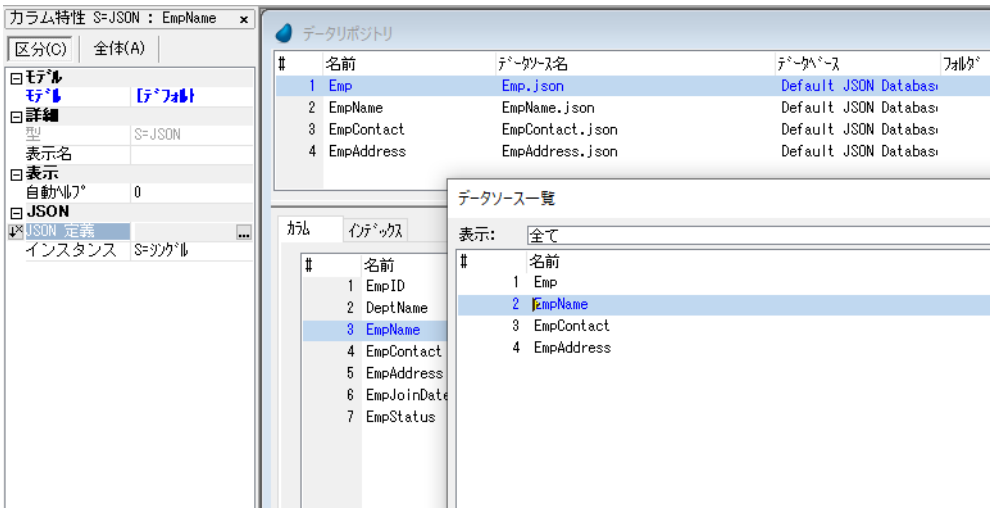


EmpAddress データ ソース

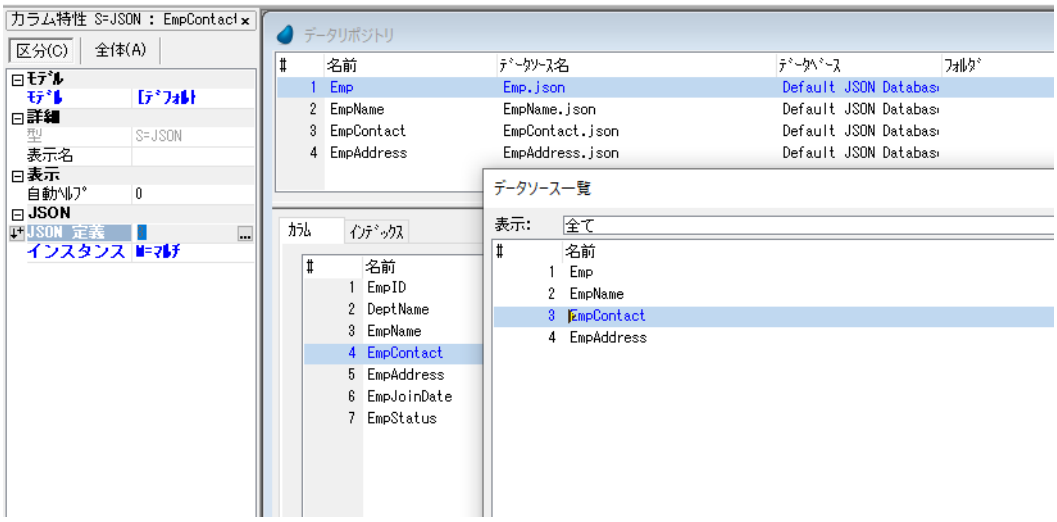


注： JSON データ ソースのの［データ ソース］特性は無効になっています。定義取得（オプション > 定義を取得）も無効です。

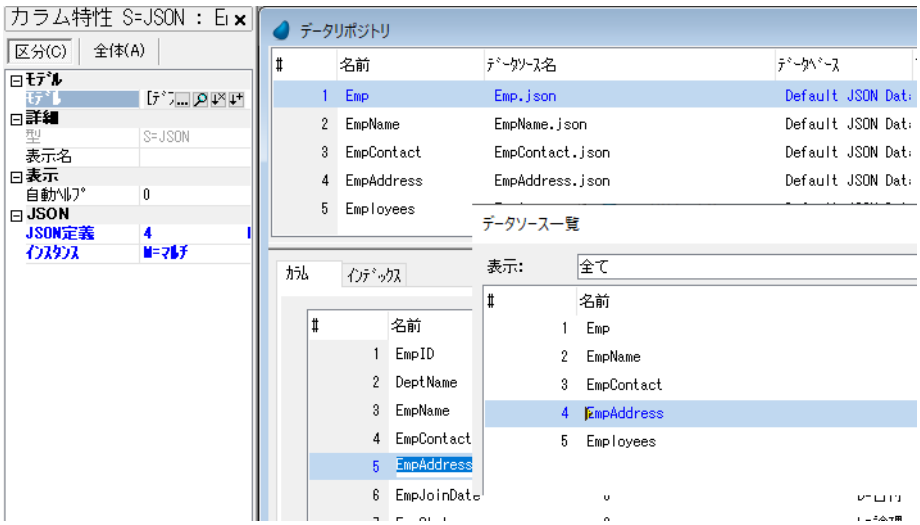
3. "Emp" データソースの "EmpName" カラムの［JSON 定義］特性に EmpName データソースを指定します。［インスタンス］特性には、「S= シングル」を設定します。



4. "Emp データソースの "EmpCotact" カラムの［JSON 定義］特性に EmpContact データソースを指定します。［インスタンス］特性には、「M= マルチ」を設定します。

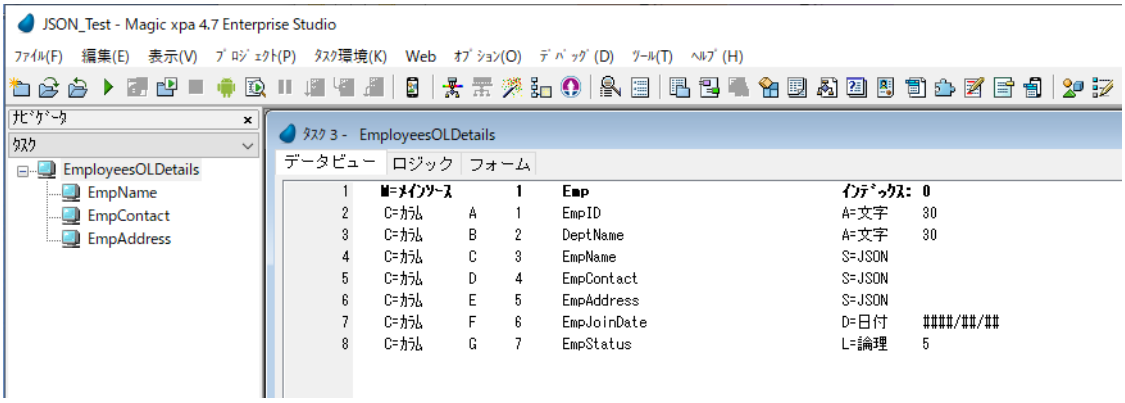


5. Emp データソースの "EmpAddress" カラムの［JSON 定義］特性に EmpAddress データソースを指定します。［インスタンス］特性には、「マルチ」を設定します。

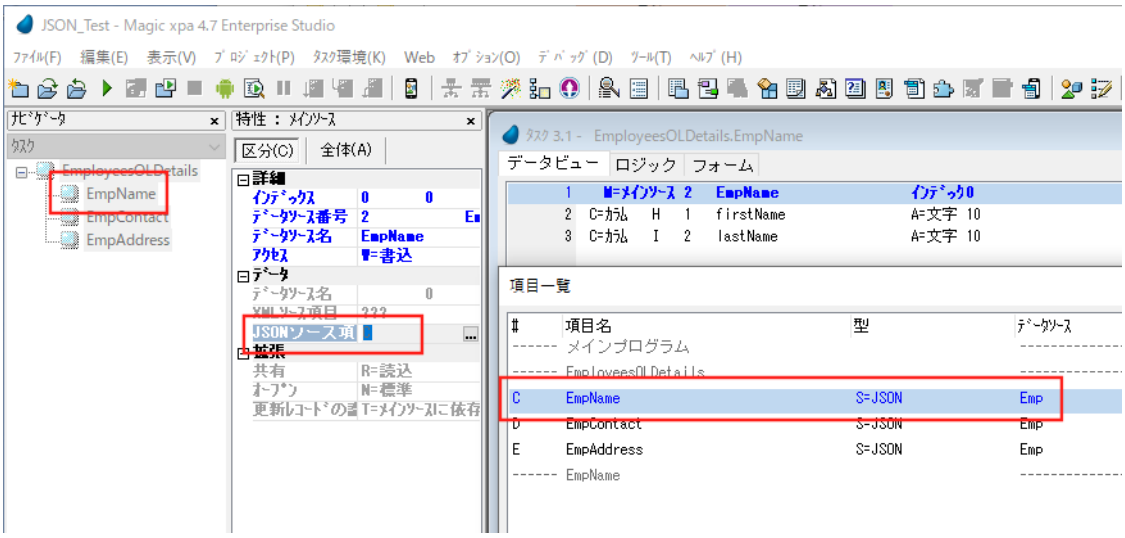


注： ［JSON 定義］特性で 4 ズームすると、［データ］リポジトリ内に定義された JSON データソースソースが一覧表示されます。

6. 対応する「シングル」と「マルチ」のインスタンステーブルを表示するプログラム "EmployeesOLDetails" を作成します。
   - メインソースには、"Emp" データソースを定義し、全てのカラムを［データビュー］ユニットに定義します。
   - "EmpName"、"EmpContact"、"EmpAddress" の各データソースをメインソースにしたサブタスクを作成します。

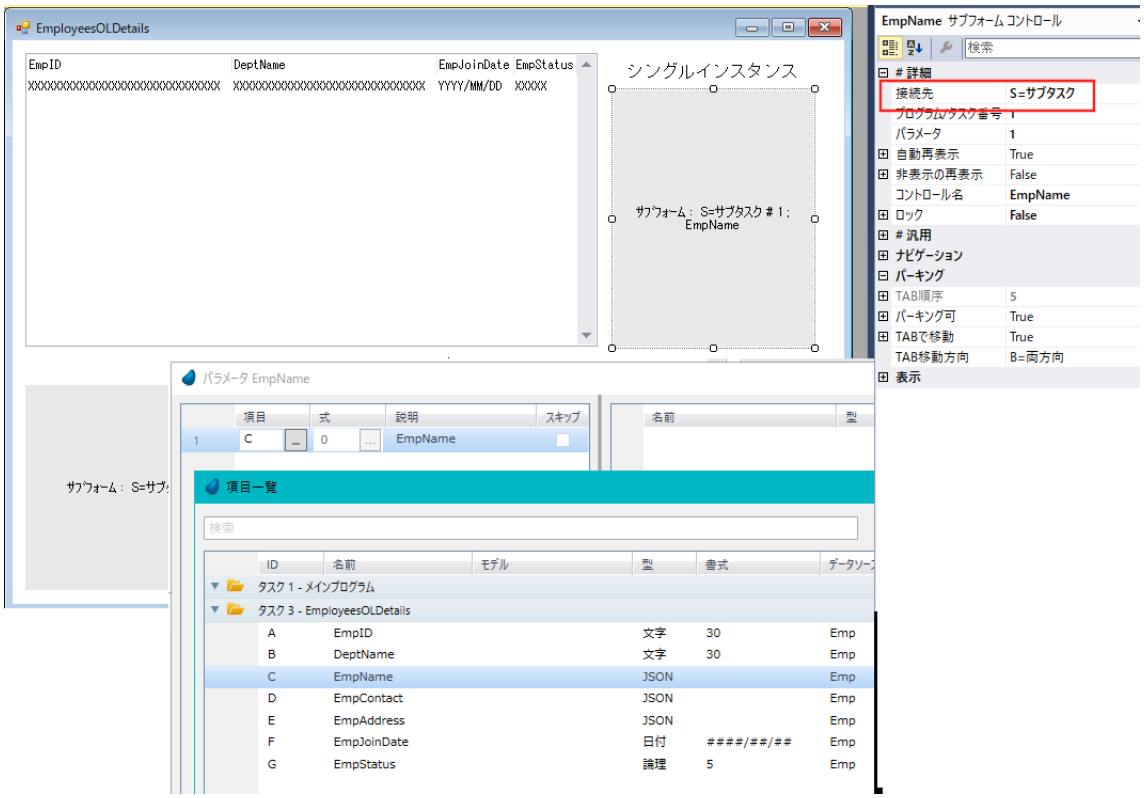


7. サブタスクにデータを提供するために、各サブタスクのメインソースの［JSON ソース項目］特性を定義します。

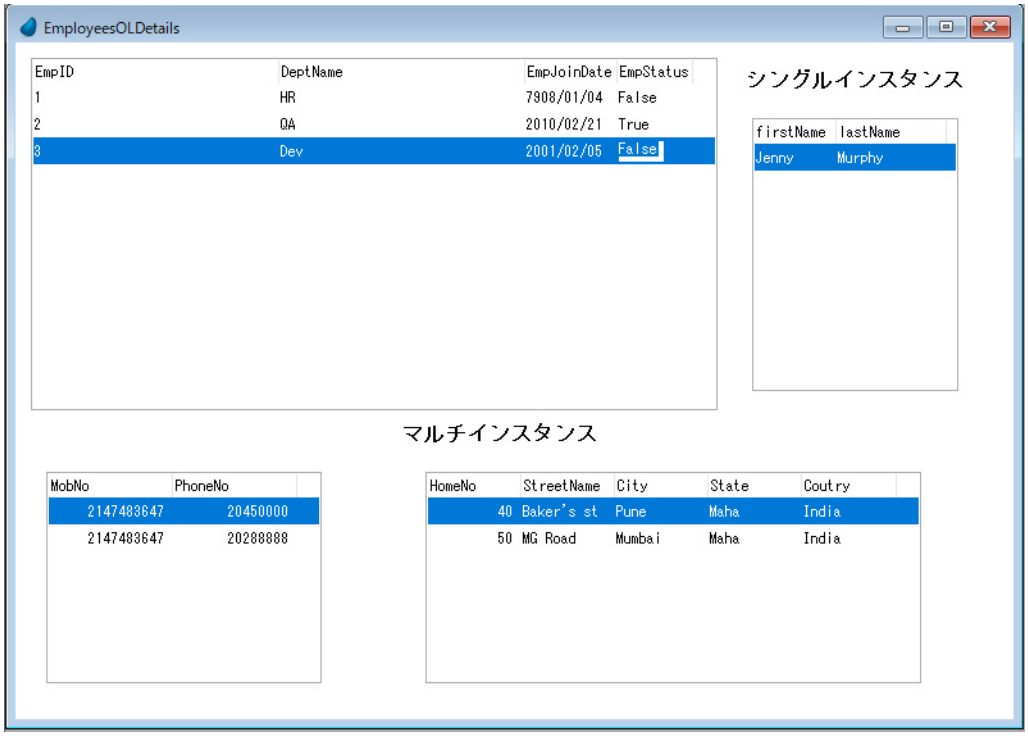


この特性には、"EmpName" サブタスクの場合、親タスクのデータビュー項目 C（"EmpNam"）を、サブタスク EmpName を呼び出す際のパラメータとして指定します。

8. メインフォームに、「シングル」および「マルチ」のインスタンス出力用のサブフォームを作成します。



9. プログラム "Emp" を実行すると、作業ディレクトリ上の emp.json が読み込まれ、プログラムの実行後に 同じファイル名で更新されます。Ctrl + 1 を押下した場合、以下のようなフォームが表示されます。



プログラムを終了すると、emp.json ファイルが更新されます。以下のような内容で出力されます。

```
{"Emp":[{"DeptName":"HR","EmpAddress":[{"City":"","Country":"","HomeNo":1,"State":"","Stre
etName":"1"},
{"City":"","Country":"","HomeNo":2,"State":"","StreetName":"2"},
{"City":"","Country":"","HomeNo":3,"State":"","StreetName":"3"}],
"EmpContact":[{"MobNo":1,"PhoneNo":1},{"MobNo":2,"PhoneNo":2},{"MobNo":3,"PhoneNo":3}],"Em
pID":"1","EmpJoinDate":"12/12/2000","EmpName":{"firstName":"aa","lastName":"aaa"},
```

"EmpStatus":false},{"DeptName":"QA","EmpID":"2","EmpJoinDate":"01/01/
2021","EmpStatus":true},
{"DeptName":"Dev","EmpAddress":[{"City":"Pune","Country":"India","HomeNo":40,"State":"Maha
","StreetName":"Baker'sst"},
{"City":"Mumbai","Country":"India","HomeNo":50,"State":"Maha","StreetName":"MG Road"}],
"EmpContact":[{"MobNo":2147483647,"PhoneNo":20450000},{"MobNo":2147483647,"PhoneNo":202888
88}],"EmpID":"3","EmpJoinDate":"10/10/2005",
"EmpName":{"firstName":"Jenny","lastName":"Murphy"},"EmpStatus":false},
{"DeptName":"sgahs","EmpID":"4","EmpJoinDate":"01/01/1901","EmpStatus":false}]}

サポートバージョン：4.8.1

第 45 章 : 日本語版での特有機能

ここでは、日本語版特有の機能を利用したプログラム例を説明します。インターナショナル版では利用できません。

IME 関係

IME を自動起動するには

Magic xpa では、文字型 /Unicode 型のデータの入力欄で自動的に IME を起動し、日本語入力を行うプログラムを作成することができます。その際、IME の起動モードは以下の方法で指定できます。

- コントロール特性の漢字入力で指定する。
- データ項目の書式で指定する。

ここでは、これらの設定方法について説明します。

コントロール特性で IME の起動モードを指定する

1. オンラインタスクを作成します。
2. GUI 表示フォームに文字型／ Unicode 型の変数項目をエディットコントロールとして配置します。
3. エディットコントロールの**漢字入力**特性に *0* から *9* までの数値を指定します。

データ項目の書式で IME の起動モードを指定する

タスクの変数項目に起動モードを設定する方法について説明します。データソースのカラムに定義する場合は、データリポジトリで行ってください。

1. オンラインタスクを作成します。
2. データビューエディタで文字型 /Unicode 型変数を定義し、書式の機能指示記号 : K と *0* から *9* までの数値を指定します。

特性値／書式と IME の関係は以下の通りです。

特性値 書式	モード	入力方法	IME パレット表示例	入力例（KANA と入力した場合）
0 K0	OFF			kana
1 K1	全 R かな	ローマ字		かな
2 K2	全かな	直接		のちみち
3 K3	全 R カナ	ローマ字		カナ
4 K4	全カナ	直接		ノチミチ
5 K5	半 R カナ	ローマ字		ｶﾅ

特性値 書式	モード	入力方法	IME パレット表示例	入力例（KANA と入力した場合）
6 K6	半カナ	直接		ﾉﾁﾐﾁ
7 K7	全英数	直接		ｋａｎａ
8 K8	半英数	直接		kana
9 K9	(変更しない)			

注： 上記の動作は、IME によって異なる場合があります。例えば以下のようになります。

特性値	MS-IME	ATOK	Google 日本語入力
0	OFF	OFF	OFF
1	全 R かな	全 R かな	全 R かな
2	全 R かな	全かな	全 R かな
3	全 R カナ	全 R カナ	全 R カナ
4	全 R カナ	全カナ	全 R カナ
5	半 R カナ	半 R カナ	半 R カナ
6	半 R カナ	半カナ	半 R カナ
7	全英数	全英数	全英数
8	半英数	半英数	半英数
9	-	-	-

注： キーボードの変換キーによって半角モードに切り替えられている場合、自動制御は動作しません。
IME のバージョンによって動作が異なる場合があります。ATOK2014/2015 では、「入力方法」が無効になります。

IME の起動モードを動的に指定する

IME の起動モードを実行時に動的に変更するには、コントロール特性の**漢字入力**か**書式**を式で指定します。**漢字入力**は数値で、**書式**は文字列で指定してください。

IME のモードを制御するには

Magic xpa は、動作環境で IME の実行方法を変更することもできます。この設定は、変更されると即時に有効になるため、**IniPut()** 関数を使用してプログラム内で動的に指定することもできます。

IME の入力方式を固定にする

IME の入力モードには、ローマ字入力とカナ入力の 2 通りがあります。通常は、IME のプロパティで設定しますが、Magic xpa の動作環境でこの設定を変更させることができます。

1. **動作環境**ダイアログを開きます（**オプション→設定→動作環境**）。
2. **国別設定**タブを開き、**IME の初期入力状態**にカーソルを移動します。この設定によって以下のように動作が変わります。

特性値	R= ローマ字	K= カナ	N= 無効	Y= 有効
0	OFF	OFF	IME のプロパティの設定にかかわらず、漢字入力の使用にもとづいて動作します。	IME のプロパティに従って動作します。
1	かな	のちみち		
2	かな	のちみち		
3	カナ	ノチミチ		
4	カナ	ノチミチ		
5	ｶﾅ	ﾉﾁﾐﾁ		
6	ｶﾅ	ﾉﾁﾐﾁ		
7	ｋａｎａ	ｋａｎａ		
8	kana	kana		
9	-			

注： **IniPut()** 関数で指定する場合のパラメータ名は、**NewIMECtrl** です。

コントロールから抜けた時点で IME の起動モードを OFF にする

漢字入力無効（特性値が「0」）に設定されたコントロールに移動した時に、前のモードを継承せずにオフの状態にする必要がある場合は、Magic xpa の動作環境で設定することができます。

1. **動作環境**ダイアログを開きます（**オプション→設定→動作環境**）。
2. **国別設定**タブを開きま、**IME を自動的にオフ**にカーソルを移動します。この設定を ***Yes*** に変更することで、コントロールを抜けた時点で漢字の入力モードがオフになります。

注： **IniPut()** 関数で指定する場合のパラメータ名は、**IMEAutoOff** です。

IME から入力された文字の読みを取得するには

ZIMERead() 関数を使用すると、IME を使用して入力した文字の読みを取得することができます。この機能を利用すると、例えば、顧客名を IME で漢字入力すると、その読み仮名を自動的に入力するができます。**ZIMERead()** 関数の構文は以下の通りです。

ZIMERead（*Number*）

パラメータ：

- *Number*：任意の数値

以下は設定例です。

```
ZIMERead(0)
```

文字が入力された後に、この関数が実行されると読み仮名（半角カタカナ）が返ります。

注： 返される読み仮名は、IME にキー入力した内容に依存します。例えば、「日本」という漢字を入力する場合、キーボードから 'NIPPONN' と入力すると 'ﾆｯﾎﾟﾝ' が返り、'NIHON' と入力すると 'ﾆﾎﾝ' が返ります。

文字操作

入力する文字を制限するには

Magic アプリケーションは、データ項目の型や書式を指定するだけで、入力する文字に制限をかけることができます。例えば、数値型の項目には、数値以外の文字を入力することができません。

入力する文字列にあらかじめ制限をかけることで、検索処理を簡素化することができます。

文字型項目の書式にて、以下のような位置指示記号を使用することでも制限することができます。

指示記号	機能
U	入力した英文字を大文字に変換します。
L	入力した英文字を小文字に変換します。
#	半角の数字やシンボル記号（. , - + * /）以外の文字を入力することができません。

日本語版の Magic xpa では、これに加え以下のような位置指示記号を使用することができます。

指示記号	機能
J	全角文字のみ入力可：2 文字単位で指定します。
T	全て全角か、全て半角で入力可：2 文字単位で指定します。
G	全角文字のみ入力可：2 文字単位で指定します。
S	半角のみ入力可

注：　この記号は、本来 iSeries/System i 用ですが、Windows システムでも利用できます。Magic xpa for IBM i で DBCS を使用する場合の詳細は、『Magic xpa for IBM i 開発ガイド』を参照してください。

位置指示記号を指定する

1. オンラインタスクを作成します。
2. データビューエディタで文字型 /Unicode 型変数を定義し、書式の機能指示記号を以下のように指定します。

```
JJJJJJJJJJ
```

プログラムを実行して文字を入力してみます。全角文字は入力できますが、半角文字を入力すると無視されます。

文字を半角に変換する

文字を入力時にチェックするのではなく、入力された後に変換する方法もあります。Magic xpa の **Han()** 関数を使用することで入力文字を半角に変換することができます。**Han()** 関数の構文は以下の通りです。

Han（*String*）

パラメータ：

- *String*：変換対象の文字列

以下は設定例です。

```
Han('アカサタナ')
```

入力された文字が半角に変換されて返ります。この例では、「ｱｶｻﾀﾅ」が返ります。

注：　変換される文字は、英数字と記号、カタカナだけです。

文字を全角に変換する

Zen() 関数を使用すると、入力文字を全角に変換することができます。**Zen()** 関数の構文は以下の通りです。

Zen（*String*）

パラメータ：

- *String*：変換対象の文字列

以下は設定例です。

```
Zen('ｱｶｻﾀﾅ')
```

入力された文字が全角に変換されて返ります。この例では、「アカサタナ」が返ります。

注： 変換される文字は、英数字と記号、カタカナだけです。半角スペースも、変換されません。

半角スペース含めて文字を全角に変換する

Zen() 関数は半角スペースを変換しません。半角スペースを含めて全角に変換する必要がある場合は、**Zens()** 関数を使用します。**ZenS()** 関数の構文は以下の通りです。

ZenS（*String,Mode*）

パラメータ：

- *String*：変換対象の文字列
- *Mode*：変換モード
  - 0……半角スペースは、変換されません。
  - 1……半角スペースは、全角スペースに変換されます。
  - 2……半角スペースは、2 個の半角スペースに変換されます。

以下は設定例です。

```
Zen('ｱｶ_ｻﾀﾅ')
```

入力された文字が全て全角に変換されて返ります。この例では、「アカ＿サタナ」が返ります（'＿'は、全角スペース）。

注： 変換される文字は、英数字と記号、カタカナだけです。

ひらがな／カタカナを変換する

ZKana（）関数を使用すると、全角のひらがな／カタカナを相互に変換することができます。**ZKana**（）関数の構文は以下の通りです。

ZKana（*String,Mode*）

パラメータ：

- *String*：変換対象の文字列
- *Mode*：変換モード
  - 0……「ひらがな」を「カタカナ」に変換します。
  - 1……「カタカナ」を「ひらがな」に変換します。

以下は設定例です。

```
ZKana('アカサタナ',1)
```

入力された文字がひらがなに変換されて返ります。この例では、「あかさたな」が返ります。

注： 全角のひらがな／カタカナ以外は、変換されません。ZKana（'あか漢字さたなﾊﾏﾔﾗﾜ',0）は、「アカ漢字サタナﾊﾏﾔﾗﾜ」が返ります。

全角文字の文字コードを取得するには

入力された文字の妥当性を判断する方法の 1 つとして文字コードを使用する場合があります。

半角文字の文字コードを取得する場合は、**ASCIIVal** 関数が使用できますが、全角文字（2 バイトコード）の場合は、1 バイト目のコードしか取得できません。このような場合は、**MIDV** 関数を使用して全角文字の 2 バイト目を取り出すことで文字コードを取得することができます。

全角文字の 2 バイト目の文字コードを取得する

MIDV 関数の構文は以下の通りです。

MIDV（*String*、*StartPosition*、*CharcterNum*）

パラメータ：

- *String*：変換対象の文字列
- *StartPosition*：文字列内の部分文字列の開始位置を表す数
- *CharcterNum*：抜き出す文字の数（部分文字列の長さ）

全角文字の 2 バイト目を取り出す場合は以下のように設定します。

```
MIDV('あ',2,1)
```

全角文字の 2 バイト目の文字コードは以下の式で取得できます。

```
ASCIIVal (MIDV ('あ',2,1))
```

文字コードを 16 進で取得する場合は、**HStr** 関数を使用して変換できます。

```
Hstr (ASCIIVal (MIDV ('あ',2,1)))
```

これにより、「あ」の 1 バイト名は、130（x'82'）、2 バイト名は、160（x'A0'）が返ります。

注： 上記の方法は、文字形項目の場合のみ有効です。Unicode 型項目の場合は、**UnicodeVal()** 関数で取得できます。

MID 関数を使用した場合、全角文字の 2 バイト目は空白が返るため使用できません。

文字データの長さを利用するには

文字データの長さを取得する

文字データの長さは、**Len()** 関数で取得できますが、データ項目が文字型の場合と Unicode 型の場合、全角文字の場合と半角文字の場合で結果が異なります。

- 文字型の場合……バイト長が返るため。全角文字の場合の実際の文字長は、この半分になります。
- Unicode 型の場合……文字長が返るため。全角 / 半角に関係なく実際の文字長になります。

以下の表は、入力する文字と文字型 /Unicode 型での **Len()** 関数の戻り値の例です。

入力文字列	文字型	Unicode 型
ABCDEFG （半角）	7	7
あいうえお	10	5
ABC うえお	9	6

関数のパラメータとして文字長を指定する

文字列を処理する関数では、文字数や位置をパラメータで指定する必要がありますが、文字型と Unicode 型で扱いが異なります。文字型項目に全角文字が入力されている状態で処理の開始位置を 2 バイト目からに設定した場合、1 バイト名が空白になります。

Ins 関数を使用して文字を挿入する例

文字列が文字型の場合

挿入先文字列（A）	挿入する文字列（A）	位置	文字数	結果
ABCDEFG	XX	2	2	A**XX**BCDEFG
あいうえお	XX	2	2	_**XX** _ いうえお
ABC うえお	XX	2	2	A**XX** うえお
ABCDEFG	漢字	2	2	A **漢** BCDEFG
あいうえお	漢字	2	2	_ **漢** _ いうえお
ABC うえお	漢字	2	2	A **漢** BC うえお

文字列が Unicode 型の場合

挿入先文字列（U）	挿入する文字列（U）	位置	文字数	結果
ABCDEFG	XX	2	2	A**XX**BCDEFG
あいうえお	XX	2	2	_ あ **XX** いうえお
ABC うえお	XX	2	2	A**XX**BC うえお
ABCDEFG	漢字	2	2	A **漢字** BCDEFG
あいうえお	漢字	2	2	あ**漢字**いうえお
ABC うえお	漢字	2	2	A **漢字** BC うえお

_：半角スペース

太字：挿入された文字

Del 関数を使用して文字を削除する例

文字列が文字型の場合

文字列（A）	開始位置	長さ	結果
ABCDEFG	2	3	AEFG
あいうえお	2	3	_うえお
ABC うえお	2	3	A_えお

文字列が Unicode 型の場合

文字列（U）	開始位置	長さ	結果
ABCDEFG	2	3	AEFG
あいうえお	2	3	あお
ABC うえお	2	3	A えお

_：半角スペース

Rep/RepV 関数を使用した文字の置き換えの例

Rep 関数は、文字型項目の全角文字列を二分すると半角スペースに置き換えますが、**RepV** 関数では、置き換え処理を行いません。このため、文字化けになる場合があります。

文字列が文字型の場合

文字列（A）	置換文字 (A)	開始位置	長さ	Rep 関数	RepV 関数
ABCDEFG	abc	2	3	A**abc**EFG	A**abc**EFG
あいうえお	abc	2	3	**_abc** うえお	**B**bc うえお
ABC うえお	abc	2	3	A**abc_** えお	A**abc、** えお
ABCDEFG	漢字	2	3	A **漢** _EFG	A 漢**殺** FG
あいうえお	漢字	2	3	_ **漢** _ うえお	**ｊ ﾘ師、** えお
ABC うえお	漢字	2	3	A **漢** _ えお	A **漢痔**えお

文字列が Unicode 型の場合

文字列（U）	置換文字 (U)	開始位置	長さ	Rep 関数	RepV 関数
ABCDEFG	abc	2	3	A**abc**EFG	A**abc**EFG
あいうえお	abc	2	3	あ **abc** お	あ **abc** お
ABC うえお	abc	2	3	A**abc** えお	A**abc** えお
ABCDEFG	漢字	2	3	A **漢字** _EFG	A **漢字** _EFG
あいうえお	漢字	2	3	あ**漢字** _ お	あ**漢字** _ お
ABC うえお	漢字	2	3	A **漢字** _ えお	A **漢字** _ えお

_：半角スペース

太字：置き換えられた文字

赤字：文字化け

日付

日付を元号で表示させるには

日付型データを元号表記で表示させるには、書式を指定するだけで実現できます。

日付データの書式を指定する

2010 年 2 月 4 日を元号表記させる場合、指定した書式に応じて以下のように表示されます。

書式	表示内容
YYYY/MM/DD	2010/02/04
JYY/MM/DD	H22/02/04
JJYY/MM/DD	平 22/02/04
JJJYY/MM/DD	平成 22/02/04
JJJYY 年 MM 月 DD 日	平成 22 年 02 月 04 日

注：　上記の書式が指定された入力項目に日付を入力する場合も、表示内容と同じ書式で日付を入力する必要があります。

文字データとして取得する

日付型データを文字データに変換するには、**DStr()** 関数を使用します。関数の構文は、12 章の「文字列で格納された日付を日付データに変換するには」を参照してください。

上記の例と同じ内容で日付データを文字列に変換すには、以下のような式を定義します。

```
DStr(A,'JJJJYY 年 MM 月 DD 日')
```

この場合、文字列「平成 22 年 02 月 04 日」が返ります。

元号名のみを取得する

日付型データから元号名のみ取得するには、**JGengo()** 関数を使用します。**JGengo** 関数の構文は、以下の通りです。

JGengo（*Date*、*StringLength*）

パラメータ：

- *Date*：変換対象の日付型データ
- *StringLength*：元号名の長さを指定します。以下の値が指定できます。
  - 1……M/T/S/H の形式になります。
  - 2……明 / 大 / 昭 / 平 の形式になります。
  - 4……明治 / 大正 / 昭和 / 平成 の形式になります。

上記の日付例で元号名を取り出す場合は、以下のように設定します。

```
JGengo('2010/02/04'DATE',4)
```

この場合、文字列「平成」が返ります。

元号年を取得する

日付型データから元号年を取得するには、**JYear** 関数を使用します。**JYear** 関数の構文は、以下の通りです。

JYear（*Date*）

パラメータ：

- *Date*：変換対象の日付型データ

上記の日付例で元号名を取り出す場合は、以下のように設定します。

```
JYear('2010/02/04'DATE')
```

この場合、数字の「22」（平成 22 年）が返ります。

日付データに 0 を表示させないようにする

日付データを表示するには書式を指定するだけで対応できますが、年月日の各数字が 1 桁の場合、10 の位が 0 で表示されます。0 を表示させないようにするには、年月日を一旦数値に置き換えて文字列にすることで実現できます。その際、日本語版用の関数を組み合わせて表示させることもできます。例えば、以下のような式を定義します。

```
JGengo(日付,4)&Str(JYear(日付),'2')&'年'&JMonth(日付)&Str(Day(日付),'2')&'日'
```

前記の日付例（2010/02/04）で実行すると、文字列「平成 22 年 2 月 4 日」が返ります。

曜日を日本語で表示させるには

日付型データをもとに、曜日を表示させることもできます。英語表記の場合は、書式に WWW... を 3 ～ 10 桁で指定すると表示されますが、日本語表記の場合は、SSS... と 2 または、4、6 桁で指定します。

日付データの書式を指定する

2010 年 2 月 4 日を表記させる場合、指定した書式に応じて以下のように表示されます。

書式	表示内容
JJJJYY 年 MM 月 DD 日 SS	平成 22 年 02 月 04 月木
JJJJYY 年 MM 月 DD 日 SSSS	平成 22 年 02 月 04 月木曜
JJJJYY 年 MM 月 DD 日 SSSSSS	平成 22 年 02 月 04 月木曜日
SSSSSS	木曜日

ヒント : *日付を変更することで、曜日の表示も自動的に変更されます。*

文字データとして取得する

日付型データから曜日名を取得するには、**JCDOW()** 関数を使用します。関数の構文は、12 章の「指定された日付に対応する曜日名を取得するには」を参照してください。

上記の例と同じ内容で日付データを文字列に変換すには、以下のような式を定義します。

```
JCDOW(A)
```

上記の日付例で曜日名を取り出す場合は、以下のように設定します。

```
JCDOW('2010/02/04'DATE')
```

この場合、文字列「木曜日」が返ります。

年の入力を省略するには

日付型データの書式は、通常（####/##/##）となり、年数を入力する必要があります。ただし、書式で年数を省略した場合（##/## または、MM/DD）、入力した日付の年数として解釈されます。

例：2010 年にデータ入力を行った場合、06/12 は、2010/06/12 というデータで格納されます。

締め日の計算を行うには

販売管理などのアプリケーションでは、締め日の計算を行う必要があります。締め日や支払日のを算出する場合は、Magic の日付関数を組み合わせる必要があります。

日付を基に締め日を算出する

当月の日付を入力した場合の締め日は、以下の式で算出できます。

- 前月の締め日以降から当月の締め日まで（Day（日付）<= 締め日）
  AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）, 締め日）
- 当月の締め日以降（Day（日付）> 締め日）
  AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）+1, 締め日）

また、締め日が末日の場合は、**EOM** 関数で算出できます。

EOM（日付）

日付を基に支払日を算出する

当月の日付を入力した場合の支払日は、以下の式で算出できます。

- 前月の締め日以降から当月の締め日まで（Day（日付）<= 締め日）
  AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）+1, 支払日）
- 当月の締め日以降（Day（日付）> 締め日）
  AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）+2, 支払日）

例えば、以下のように算出できます。

締め日：15 日／支払日：翌月 10 日

- 前月 16 日から当月 15 日まで（Day（日付）<= 15）
  - 締め日
    AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）,15）
  - 支払日
    AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）+1,10）
- 当月 16 日以降（Day（日付）> 15）
  - 締め日
    AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）+1,15）
  - 支払日
    AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）+2,10））

締め日：末日／支払日：翌月 26 日

- 前月 1 日から当月末日まで
  - 締め日
    AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）,EOM（日付））
  - 支払日
    AddDate（'0000/00/00'DATE,Year（日付）,Month（日付）+1,26）

土日の場合に翌日／翌々日に変更する

支払日が土日により翌日（または翌々日）に変更する必要がある場合、**DOW** 関数を使用することで判定できます。

- 土曜日の場合（翌々日に変更する）
  IF DOW（日付）= 7
  日付 + 2
- 日曜日の場合（翌日に変更する）
  IF DOW（日付）= 1
  日付 + 1

祝日の場合に日付を変更する

祝日により日付を変更する必要がある場合は、祝日用のテーブルを別途用意して、これを基に判定するようにしてください。

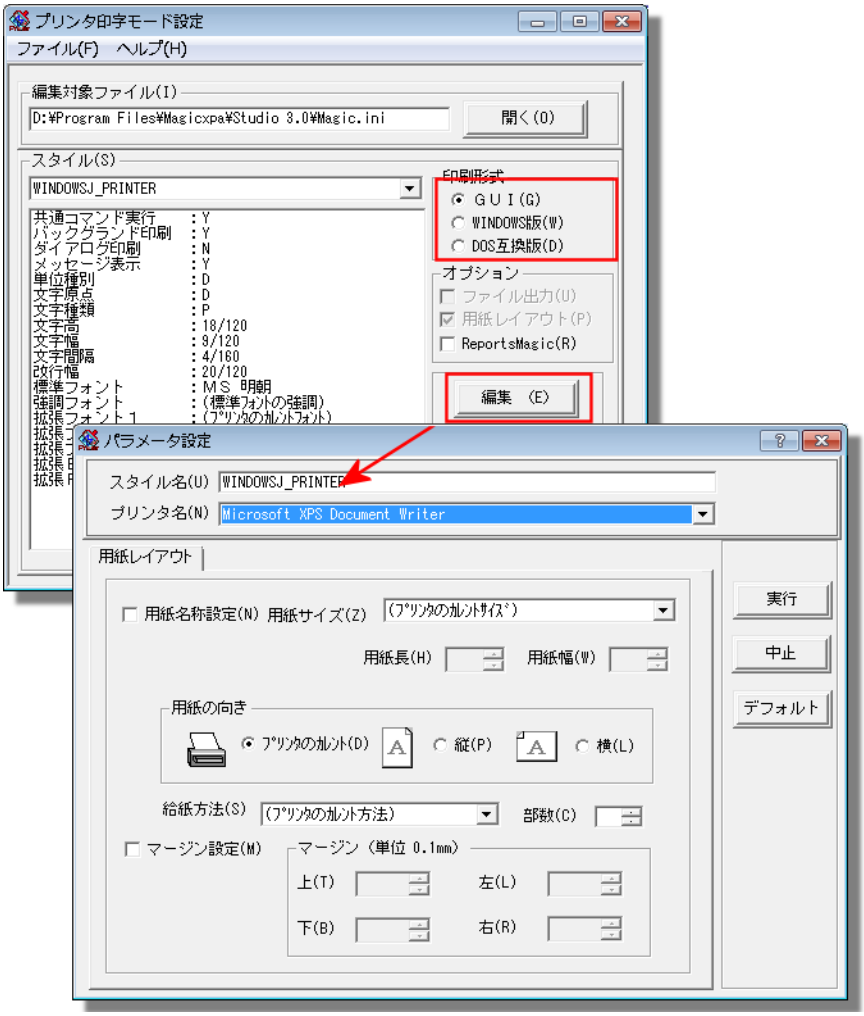
印刷

GUI 印刷を行う場合、入出力特性を利用して出力方法（用紙サイズや向きなど）を指定することができますが、これ以外に日本版の Magic xpa では、印刷用ユーティリティを使用することで様々な設定を行うことができます。

注： 印刷ユーティリティは、テキスト形式の印刷にも対応していますが、ここでは、GUI 印刷を行う場合の設定例のみ説明しています。

印刷モードを設定するには

プリンタ設定ユーティリティを使用する



印刷モード設定（MGPRN.EXE）ユーティリティで印刷設定を指定することができます。

1. Magic xpa を一旦終了させます。
2. Magic xpa がインストールされているフォルダ内の MGPRN.EXE を実行します（**スタート→プログラム→ Magic xpa Studio V2 →テキスト印刷ユーティリティ→プリンタ設定ユーティリティ**でも実行できます）。
3. **編集対象ファイル**の開くボタンを**クリック**して、使用している MAGIC.INI ファイルを選択します。
4. **印刷形式**で ***GUI*** を選択します。
5. 編集ボタンを**クリック**して**パラメータ設定**ウィンドウを開きます。
6. ここでは、以下の設定が可能です。
   - 出力するプリンタ名（OS に登録されているプリンタから選択できます。）
   - 用紙のサイズ
   - 用紙の向き
   - 給紙方法

- 部数
- 印刷マージン

7. 設定が終了したら、実行ボタンをクリックしてパラメータ設定ウィンドウを閉じます。
8. 終了ボタンをクリックしてユーティリティを終了させます。

設定内容は、MAGIC.INI ファイルの［WINDOWSJ_PRINTER］セクションに反映されます。Magic uniPaS を起動させて、印刷プログラムを実行するとユーティリティによって指定された内容で出力されます。

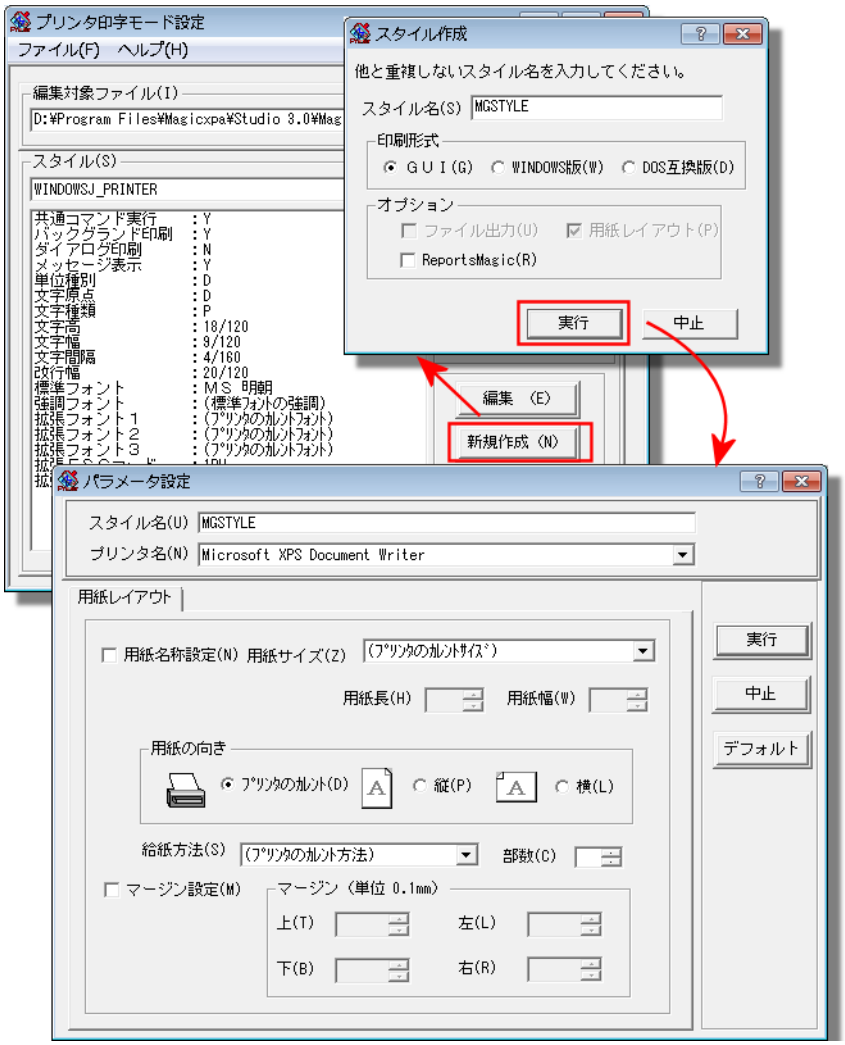
注： この設定は、出力特性より優先されます。出力特性の内容で印刷させる場合は、全てのパラメータをデフォルト（プリンタのカレント）に設定する必要があります。

設定内容は、Magic xpa を再起動するまで反映されません。

スタイル定義を利用するには

印刷モード設定（MGPRN.EXE）ユーティリティによる印刷設定は、デフォルトでは［WINDOWSJ_PRINTER］セクションに登録されます。ただし、別の名前で複数のスタイル情報を登録して使い分けることもできます。

この場合は、スタイル設定（SETSTYLE.EXE）ユーティリティで Magic xpa のプリンタテーブルのキューカラムにスタイル情報を割り当てる処理が必要になります。



スタイルを定義する

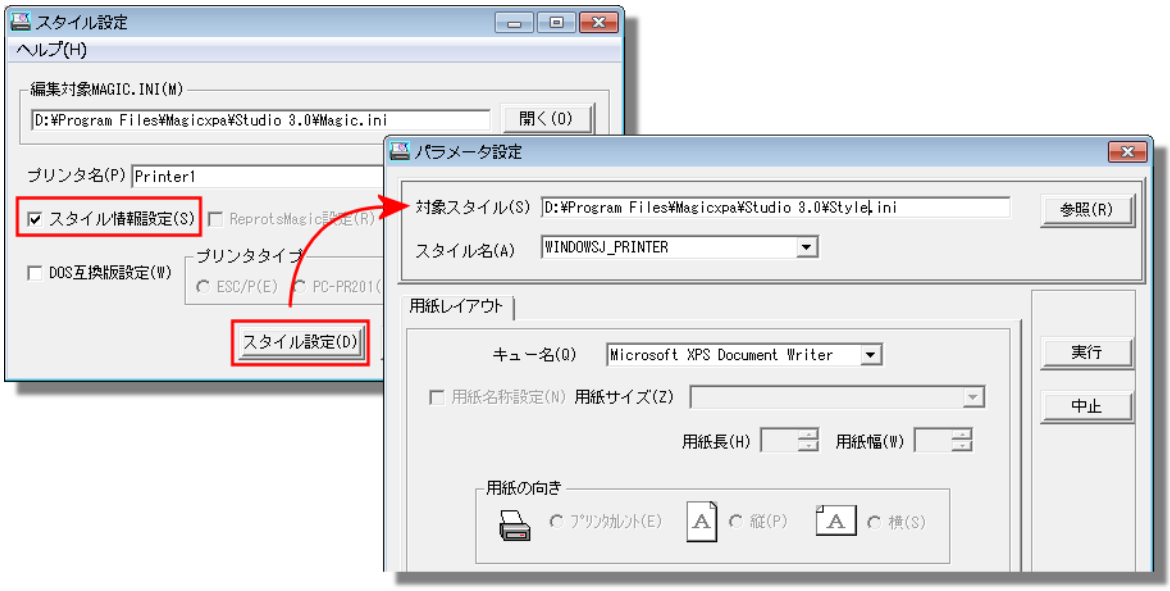
1. Magic xpa を一旦終了させます。
2. スタイルファイルとなるテキストファイルを作成します（例：STYLE.INI）。中身は、空のままにします。
3. Magic xpa がインストールされているフォルダ内の MGPRN.EXE を実行します。
4. 編集対象ファイルの開くボタンをクリックして、スタイルファイル（STYLE.INI）を選択します。
5. 新規作成ボタンをクリックしてスタイル作成ウィンドウを開きます。
6. スタイル名にユニークな名前（例：MG_STYLE1）を入力します。
7. 印刷形式で *GUI* を選択して、実行ボタンをクリックします。パラメータ設定ウィンドウを開きます
8. 前述と同じように設定します。
9. 設定し終わったら、実行ボタンをクリックしてパラメータ設定ウィンドウを閉じます。
10. 終了ボタンをクリックしてユーティリティを終了させます。

上記の例では、設定内容が STYLE.INI ファイルの［MG_STYLE1］セクションに反映されます。次にこのスタイル定義を有効にする処理を行います。

スタイルをキューに割り当てる

1. Magic xpa がインストールされているフォルダ内の SETSTYLE.EXE を実行します（スタート→プログラム→ Magic xpa Studio V2 →テキスト印刷ユーティリティ→スタイル設定ユーティリティでも実行できます）。
2. 編集対象ファイルの開くボタンをクリックして、使用している MAGIC.INI ファイルを選択します。

3. **プリンタ名**では、MAGIC.INI に定義されている論理プリンタ名から選択します。このプリンタ名にスタイルが割り当てられます。
4. **スタイル情報設定**をチェックします。
5. スタイル設定ボタンを**クリック**して、**スタイル設定**ウィンドウを開きます。
6. **対象スタイル**で、参照ボタンをクリックして**スタイルファイル**（例：STYLE.INI）を選択します。
7. **スタイル名**には、**印刷モード設定**ユーティリティによって定義されたスタイル名が選択できるようになります。この例では、MG_STYLE1 を選択しています。
8. **プリンタ設定**では、**キュー名**のみ変更できます。ここでは、OS に定義されているプリンタ名から選択します。
9. 実行をクリックして、ウィンドウを閉じます。
10. 登録をクリックして、ユーティリティを終了します。

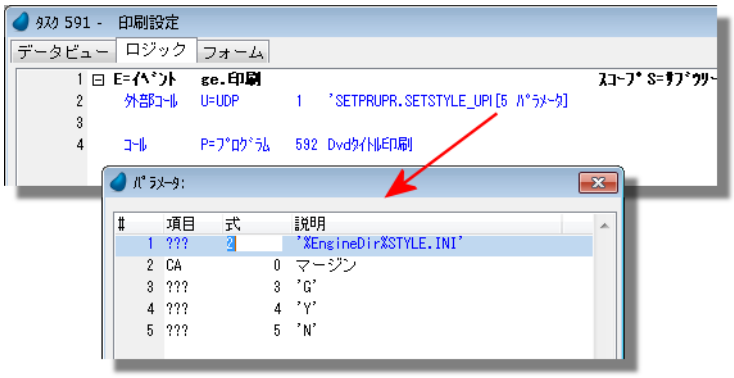


設定内容は、MAGIC.INI ファイルの［MAGIC_PRINTER］セクションに反映されます。Magic xpa を起動させて、印刷プログラムがスタイルを割り当てたプリンタを使用するように定義することで、スタイルが反映されます。

注： ユーティリティではスタイルファイルを新規作成することはできないため、あらかじめテキストファイルを用意しておく必要があります。

実行時にスタイルを切り替える

スタイル設定ユーティリティを使用してスタイルを割り当てた場合、MAGIC.INI ファイルに設定されるため、Magic xpa を一旦終了させてから行う必要があります。アプリケーションの実行時に動的に切り替えたい場合は、**スタイル変更**ユーティリティを使用します。



1. 印刷プログラムの**タスク前**、または上位プログラムにて、**コール UDP** 処理コマンドを定義します。
2. UDP 表示の次のカラムに移動して**式**エディタに**ズーム**し、以下の式を入力します。
   'SETPRUPR.SETSTYLE_UPR'
3. **パラメータ**特性で**ズーム**して、以下の 5 つのパラメータを定義します。
   - スタイルファイル（上記の例では、STYLE.INI）
   - スタイル名（上記の例では、MG_STYLE1）
   - 印刷形式（GUI 印刷の場合：*G*）

- 用紙レイアウト（GUI 印刷の場合：*Y* 固定）
- ダイアログ表示（表示しない場合は、*N*）

このプログラムを実行すると、指定されたスタイルの設定内容に基づいて印刷されます。

注： プリンタテーブルのキューカラム（MAGIC.INI ファイルの［MAGIC_PRINTER］セクション）にスタイル情報が定義されている場合、スタイル変更ユーティリティによって切り替えることはできません。あらかじめ、スタイル設定ユーティリティでスタイル情報設定のチェックを外した状態に設定しておいてください。

Magic xpa 逆引き辞典

発行 〒 169-0074 東京都新宿区北新宿二丁目二十一番地一号
新宿フロントタワー 24 階

Magic Software Japan K.K.