Magic eDeveloper V10

新機能チュートリアル

MAGIC
Magic Software Japan K.K.

Enabling Business with Superior Technology

本書に記載の内容は、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。

本書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。

本書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、本書のいかなる部分も、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 の製品に対する互換性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。

本書において、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。

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。

初版　2006年12月20日

マジックソフトウェア・ジャパン株式会社

目次

目次 3
1. はじめに 11
1.1. 本書の目的 12
1.2. 前提条件 13
1.3. 本書で使う表記 15
1.4. サンプル実行環境の設定 16
2. プロジェクトを開く 18
2.1. Magic スタジオを起動する 19
2.2. プロジェクトのソースファイル 20
2.2.1. プロジェクトファイル 20
2.2.2. リポジトリソースファイル 20
2.3. 既存のプロジェクトを開く 21
2.4. アプリケーション特性 23
2.4.1. スタートアップ 23
2.4.2. 外部ファイル 24
2.5. プロジェクトを閉じる 26
2.6. 新しいプロジェクトを作成する 27
2.6.1. 新規プロジェクトの作成 27
2.6.2. V9Plus との比較 27
2.6.3. ソースファイルのディレクトリ名を変える 28
2.7. プロジェクトモジュールとナビゲーション 29
2.7.1. 新規プロジェクトをモジュールとして追加する 29
2.7.2. 既存のプロジェクトをモジュールとして追加する 29
2.7.3. プロジェクト中のモジュールを移動する 30
2.7.4. モジュールを名前変更・削除する 30
2.8. デフォルトプロジェクトを設定する 32
3. データリポジトリ 33
3.1. 新しいインターフェース 34
3.1.1. 上のペイン 34
3.1.2. 下のペイン 34
3.2. データソースを削除する 36
4. タスクエディタ 37
4.1. プログラムリポジトリ 38
4.2. データビュー エディタ 40
4.2.1. データビューでのコマンド 41
4.2.2. 特性 42
4.2.3. 複数一括選択 45

4.2.4. データビュー画面の表示 48
4.3. ロジックエディタ 50
4.3.1. ロジックエディタの構成 50
4.3.2. ハンドラの種類 51
4.3.3. コマンドの種類 51
4.3.4. ハンドラの順序 53
4.3.5. ハンドラの自動作成 54
4.4. フォームエディタ 55
4.5. タスクエディタの保存と終了 56
4.5.1. タスクエディタの終了 56
4.5.2. タスク修正内容の保存 56
4.6. タスク特性ダイアログボックス 57
4.6.1. V9Plus のタスク特性/タスク制御と、V10 でのタスク特性の対応 57
4.6.2. V10 で新たに追加されたタスク特性 61
4.7. 範囲ダイアログボックス 62
4.7.1. 範囲タブ 62
4.7.2. SQL Where と 式 タブ 63
4.8. データビュー／ロジックエディタの共通機能 64
4.8.1. クイック式入力 64
4.8.2. 表示の展開と縮小 64
4.8.3. クイックアクセスメニュー 68
4.8.4. ヘッダ行の削除 68
4.8.5. マルチマーキング 69
4.8.6. テキストの検索 69
4.9. 式エディタの強化 70
4.9.1. 関数の自動コンプリート 70
4.9.2. 式の展開 70
4.9.3. 選択リスト 71
4.9.4. コンテキスト依存ヘルプ 71
4.9.5. V9 よりの関数の変更 72
5. 項目変更ロジックユニット 74
5.1. 項目変更ハンドラの作成 75
5.2. 実行時の動作 77
6. ユーザ定義関数 78
6.1. ユーザ定義関数の定義 79
6.2. ユーザ定義関数のスコープ 80
6.3. ユーザ定義関数の参照 81
6.3.1. 式エディタ 81
6.3.2. 自動コンプリート 81
6.3.3. 関数一覧 81

6.3.4. シンタックスチェック 82
6.4. その他の注意事項 83
6.4.1. 既定義の関数と同じ関数名を使う場合 83
6.4.2. ユーザ定義関数を削除する場合 83
6.4.3. パラメータと戻り値 83
6.4.4. レベルとタスク世代 83
7. GUI の強化 84
7.1. TAB 順序 85
7.1.1. 自動／手動 TAB 順序の設定 85
7.1.2. 手動 TAB 順序 85
7.1.3. 自動 TAB 順序 86
7.1.4. TAB 順序の表示 86
7.2. GUI 特性の パーク カテゴリ 87
7.3. Windows コントロール 88
7.3.1. 前提条件 88
7.3.2. コントロール スタイルの設定 89
7.3.3. タブ コントロール 89
7.3.4. チェックボックス コントロール 93
7.3.5. ラジオボタン コントロール 96
7.4. サブフォーム 100
7.4.1. サブフォームをフォームに配置する 100
7.4.2. サブフォーム タスク実行ルール 101
7.4.3. サブフォーム タスク特性 102
7.4.4. サブフォーム タスク終了ルール 102
7.4.5. サブフォームの自動再表示 103
7.4.6. サブフォーム再表示イベント 103
7.4.7. サブフォーム関連の関数 105
7.5. プッシュボタン コントロール 106
7.5.1. 実行元 特性 106
7.6. メニュー 107
7.6.1. メニューリポジトリ 107
7.6.2. フォームにコンテキストメニューを設定する 108
7.6.3. コントロールにコンテキストメニューを設定する 109
7.6.4. コンテキストメニューの表示ルール 109
7.6.5. 実行時にプルダウンメニューにメニューを追加する 110
7.7. フォーム状態の保存と復元 111
7.7.1. フォーム状態 ID 特性 111
7.7.2. フォーム状態 ID の効果の確認 112
7.7.3. フォーム状態 ID 利用上の注意点 115
7.7.4. フォーム状態関連関数 115

8. 以前のバージョンから V10 への移行 116
8.1. 移行パス 117
8.1.1. バージョン別の移行パス 117
8.1.2. 移行のための旧バージョン 118
8.2. サポートされなくなった機能 119
8.2.1. サポートされなくなった関数 119
8.2.2. Web Online 119
8.2.3. HTML フォーム 119
8.2.4. エディット コマンド 119
8.2.5. 動作環境 119
8.3. V9Plus でのリポジトリ出力 120
8.3.1. リポジトリ出力時の注意事項 120
8.3.2. リポジトリ出力ファイルの確認 120
8.4. 移行ユーティリティの処理プロセス 122
8.4.1. 移行処理の内容 122
8.4.2. ユーザ定義イベントの強制終了パラメータ 123
8.4.3. パーキング関係の設定 124
8.4.4. 照会リンクの移行 125
8.5. V9Plus コンバータのパラメータ 127
8.6. ログファイル 128
8.7. GUI エンハンサ 130
8.7.1. GUI エンハンサとは？ 130
8.7.2. GUI エンハンサの利用方法 131
8.8. INI 変換ユーティリティ 133
8.8.1. INI 変換ユーティリティとは？ 133
8.8.2. INI 変換ユーティリティの使い方 134
8.9. 権利設定とセキュリティファイル 135
8.9.1. 権利設定の変更 135
8.9.2. セキュリティファイル 136
8.10. ビルダデータコンバータ 137
8.11. 作業ディレクトリ 138
8.11.1. 作業ディレクトリの変更 138
8.11.2. 既定義の論理名 138
8.11.3. コンポーネントの場合 138
8.12. RM 互換ロジックユニット 139
8.12.1. RM 互換ハンドラの作成 139
8.12.2. 参照 コマンド 139
8.12.3. データビューとの同期 140
8.12.4. TAB 順序 140
8.12.5. 行間の移動 140

8.12.6. コマンドの定義 141
8.12.7. RM 互換ロジックユニットの実行時動作 142
8.13. dbMAGIC V8.2 からの移行 143
8.13.1. V9Plus での色/フォントテーブル変換は不要 143
8.13.2. テーブルのタイトルが消える 143
8.14. ファントムタスク 145
8.14.1. Ver 9.40J SP3 での機能拡張とは？ 145
8.14.2. 機能拡張の波及と対応方法 145
8.15. トラブルシューティング 146
9. Datetime データ型のサポート 148
9.1. Datetime 型のカラムをデータソースリポジトリに定義 149
9.2. Datetime を構成する日付/時刻項目の変更 151
9.2.1. 変更時 151
9.2.2. 削除時 151
9.3. インデックス 152
9.3.1. インデックス定義 152
9.3.2. 外部キー 152
9.4. 定義取得 153
9.5. タスクでの利用 154
9.6. Datetime 関連の関数 155
9.6.1. DifDateTime 155
9.6.2. AddDateTime 155
10. ビュー事前読込 156
10.1. ビュー事前読込特性 157
10.2. ビュー事前読込関連の関数 158
10.3. ビュー事前読込機能のサンプル 159
10.4. ビュー事前読込利用時の注意事項と補足事項 160
11. XML サポート 161
11.1. 設定 162
11.1.1. DBMS リポジトリ 162
11.1.2. データベースリポジトリ 162
11.2. XML ファイルをデータリポジトリに定義する 164
11.2.1. XML データソースファイル定義から定義取得する 164
11.2.2. スキーマツリーの表示 170
11.2.3. データソース ビューを定義する 171
11.2.4. インデックス定義 174
11.2.5. 同じ XML ソースファイルから異なる XML ビューを定義する 174
11.3. XML ファイルの作成 176
11.4. XML 関数 177
11.4.1. XMLValidate 177

11.4.2. XMLValidationError 177
12. UNICODE 178
12.1. Unicode 型のモデル・項目を定義する 179
12.2. Unicode 型モデル・項目の特性 180
12.3. 変数間でのデータ変換 181
12.4. データ変換用関数 182
12.4.1. UnicodeFromANSI 182
12.4.2. UnicodeToANSI 182
12.4.3. UnicodeChr 182
12.4.4. UnicodeVal 183
12.5. Unicode サポート機能 184
12.5.1. I/O ファイル 184
12.5.2. フォーム・コントロール特性 185
12.5.3. 式 185
12.5.4. メニュー 185
12.5.5. ヘルプリポジトリ 185
12.6. 実行エンジンの動作 186
12.6.1. データの照合順序 186
12.6.2. 関数 186
12.6.3. その他の制限事項 188
13. 開発エンジンと実行エンジンの分離 189
13.1. 開発エンジンでの操作 190
13.1.1. 開発エンジンからプロジェクトを実行する 190
13.1.2. RunMode 関数 191
13.1.3. 実行中のアプリケーションを閉じる 191
13.1.4. 実行エンジンの異常終了時の動作 192
13.1.5. 開発エンジンからプログラムを実行する 192
13.1.6. 実行エンジンの強制停止 193
13.1.7. 実行エンジンをリセットする 193
13.2. アプリケーションの配備 194
13.2.1. キャビネットファイルを作成する 194
13.2.2. キャビネットファイルをオープンする 195
13.2.3. アプリケーションを閉じる 196
13.3. 動作環境の設定 197
14. 複数のタスクを同時に実行する 199
14.1. 並行実行プログラムのいろいろ 200
14.1.1. 並行実行の確認 201
14.2. 並行実行の設定 202
14.3. 単一インスタンス 203
14.4. プログラム再呼出 イベント 205

14.5. 並行実行プログラム間の相互作用 206
14.5.1. 航空会社プログラムの例 206
14.5.2. コンテキストのリフレッシュ イベントの作成 207
14.5.3. コンテキスト名の設定 208
14.5.4. コンテキストのリフレッシュ イベントの実行 208
14.5.5. コンテキストのリフレッシュ イベントの処理 209
14.6. コンテキスト ID 210
14.6.1. コンテキスト ID の生成、設定と取得 210
14.6.2. メインコンテキスト 210
14.6.3. 新規作成したコンテキスト ID の取得 210
14.6.4. コンテキスト ID を指定してのイベントの発行 212
14.7. ウィンドウメニュー 213
14.7.1. メニューウィンドウの表示内容 213
14.7.2. ウィンドウの順序 213
14.7.3. 新しいイベント 214
14.7.4. 関数 215
14.8. SDI ウィンドウ 216
14.8.1. ウィンドウタイプ特性 217
14.8.2. SDI ウィンドウの例 219
14.8.3. SDI ウィンドウのプルダウンメニュー 220
14.8.4. Magic のウィンドウを開かない SDI プログラム 221
14.9. コンテキスト関連のイベント 225
14.9.1. コンテキストを閉じる 225
14.9.2. アプリケーションクローズ 225
15. デバッガ 226
15.1. デバッガを開始する 227
15.2. アクティビティ モニタ 228
15.2.1. ロギング フィルタ 228
15.2.2. アクティビティ モニタ 229
15.2.3. Logging 関数 230
15.3. ブレイクポイント 232
15.3.1. ブレイクポイントを追加する 232
15.3.2. プログラムを実行する 233
15.3.3. 実行中に強制的にブレイクする 234
15.3.4. ブレイクポイントリポジトリ 234
15.4. 項目一覧 237
15.4.1. 項目一覧の表示 237
15.4.2. 項目一覧のオプション 237
15.5. ウォッチリスト 239
15.5.1. 項目をウォッチに追加する 239

15.5.2. ウォッチリストの表示 240
15.6. コールスタック 241
15.6.1. コールスタックの表示 241
15.6.2. コールスタックのオプション 241
15.7. 実行コンテキスト 243
15.7.1. 実行コンテキストの表示 243
15.7.2. コンテキスト切替え 243
付録 A. 利用可能なコードページの一覧 245

1. はじめに

Magic eDeveloper V10 をお使いいただきまして、ありがとうございます。Magic eDeveloper V10 は、高い生産性と保守性を誇る Magic パラダイムの歴史の上に、最新のテクノロジを融合して、マルチプラットフォーム、マルチDBMS、マルチパラダイムの高度な RADD (Rapid Application Development and Deployment)/複合アプリケーション (Composite Application)開発 ツールです。

Magic 製品により実現されるシステム形態は、スタンドアロンの小規模なものから大規模なクライアントサーバのオンラインシステム、およびサーバベースの Web アプリケーション、XML や Web サービス、J2EE 対応アプリケーション(サーバ、クライアント共)など、非常に広範囲にわたります。

本書では、すでに Magic eBusiness Platform V9Plus をご利用されている Magic 開発者を対象に、Magic eDeveloper V10 での新機能をフォーカスして解説するものです。

Magic eDeveloper V10 の新機能は多岐にわたるので、本書の限られた範囲では基本的な機能紹介だけを行っています。より詳細な機能については、リファレンスマニュアル、開発者ガイド、添付サンプルアプリケーション、弊社ホームページ (http://www.magicsoftware.co.jp) の技術情報などを参照されることをお勧めいたします。

1.1. 本書の目的

本書は、サンプルアプリケーション TravelAgency を例にとって、以下のような Magic eDeveloper V10 の新機能を学んで行きます。

1. プロジェクト パラダイム
2. 開発エンジンのインターフェース
3. 開発エンジンの機能強化
4. データリポジトリ
5. タスクエディタ
6. 式エディタ
7. イベントをベースにしたプログラミング パラダイムを実現するためのロジックユニット
8. ユーザ独自の関数の定義
9. 新しい GUI 機能を用いた、よりユーザフレンドリーなアプリケーションの設計
10. タブ順序と項目定義の順序との分離
11. サブフォーム コントロールを使った、親子タスクの表示
12. メニューリポジトリの強化
13. V9Plus アプリケーションの V10 への移行
14. Datetime 型のサポート
15. Magic のデータソースとして XML ファイルの利用
16. Unicode のサポート
17. 開発エンジンと実行エンジンの分離
18. 複数のプログラムの並行実行
19. 異なるコンテキスト間のイベントの受け渡し
20. MDI と SDI のウィンドウタイプ
21. SDI プログラムを実行エンジンの起動時に開始すること
22. Magic デバッガの強化

1.2. 前提条件

本書の読者は、次のような前提条件を満たしていることが必要です。

PC に関する基礎知識	1. テキストファイル、HTML ファイルを参照・作成するためのエディタの使い方を知っている。 2. インストーラを使ってプログラムをインストール・アンインストールすることができる。 3. Windows XP の環境についての基本的な知識を持っている。
Magic 開発に関する知識	1. Magic V9Plus を使って 6 ヶ月以上の経験がある。 2. 以下の話題について基本的な知識がある。 ● データベース ● Web サービス ● XML 3. 次の言葉を良く理解している。 ● DLL ● Unicode ● 関数 ● Windows コントロール ● 並行実行 ● コンテキスト ● イベント ● デバッガ ● ブレークポイント ● バージョン管理 ● Java
データベースに関する知識	次のような、データベースに関する基本的な知識がある。 ● データベース ● テーブル ● 行/レコード ● フィールド/カラム ● インデックス ● Datetime ● ストアド プロシージャ
Web サービスに関する知識	次のような Web サービスに関する言葉を理解している。 ● UDDI ● WSDL ● RPC/ドキュメント形式
XML に関する知識	次のような、XML に関する言葉を理解している。

	● XSD（スキーマ） ● 要素 ● 複合要素 ● 名前空間

1.3. 本書で使う表記

本書では、次のようなアイコンを使っています。

参考：補足的な説明を簡単に記述します。

操作：Magic を実際に操作する手順を説明します。

TIPS：知っておくと便利な豆知識を紹介します。

注意：おちいりやすい落とし穴、間違えやすい事柄などを示します。

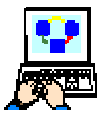
V9
Plus

V9Plus との比較：V9Plus と V10 とでの差異点を対照して説明します。

1.4. サンプル実行環境の設定

本書は、サンプルアプリケーション TravelAgency を例として説明しています。TravelAgency は Magic スタジオ製品にサンプルとしてとして添付されており、インストーラは環境を設定しますので、通常は特別な設定なくそのままで使うことができます。Magic スタジオの起動のしかたは、2.1 Magic スタジオを起動する を参照してください。

ここでは、参考のために、TravelAgency で必要な設定について説明します。何らかの理由で、再度設定しなおす必要がでてきた場合には、以下の要領でセットアップしてください。



サンプル TravelAgency を設定するには・・・

1. Projects というディレクトリの下に Accounting および TravelAgency というサブディレクトリがあるのを確認してください。もしなければ、Magic スタジオの修復インストールを行ってください。
2. MAGIC.INI の[MAGIC_DATABASES] セクションに、以下の定義を追加してください。

```
[MAGIC_DATABASES]
Travel Database = DBMS, 1, MAGIC, %Travel%Data¥, , , , , NoMagicRecordLock,+
 ChangeFileInToolkit, CheckDefinition, CheckKey, FileLocks, , , NoCheckExis+
t, 0, , NoAS400SrvrSort,
Travel XML Files = XML, 0, , %Travel%XML¥, , , , , NoMagicRecordLock, +
ChangeFileInToolkit, CheckDefinition, CheckKey, FileLocks, , , NoCheckExist+
, 0, , NoAS400SrvrSort, , Memory
```

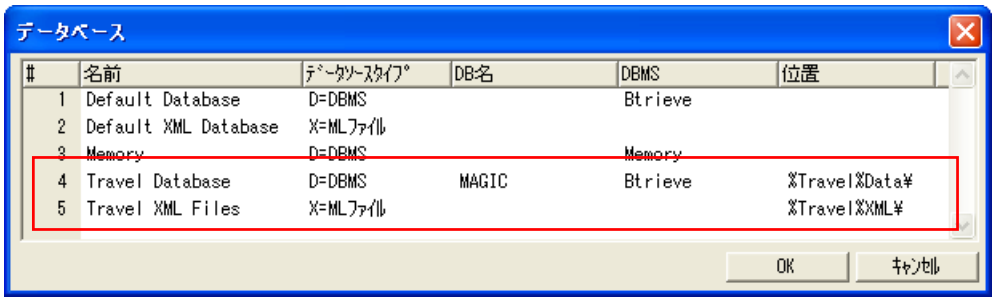
また、[MAGIC_LOGICAL_NAMES] セクションに、以下の定義を追加してください。

```
[MAGIC_LOGICAL_NAMES]
Travel = .¥
Travel_Pics = %Travel%Pictures¥
```

以上です。

上のステップ 3 は、以下のように Magic スタジオを起動して、データベーステーブルおよび論理名テーブルを編集しても同じです。

データベーステーブル:



以下の二つのデータベースを登録します。

- Travel Databse
- Travel XML Files

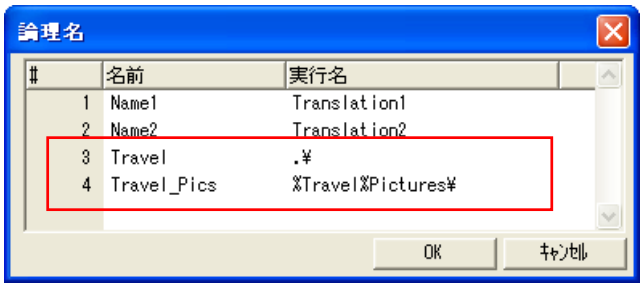
Travel Database は通常の Btrieve データベースで、**位置** として %Travel%Data% を指定します。

Travel XML Files は、V10 で新たに導入された XML View の機能を使うデータベースで、**データソースタイプ** 欄として、**X=ML ファイル** を指定します。このデータソースタイプは既定義のものなので、**データソースタイプ** 欄のコンボボックスから選んで設定します。

論理名テーブル

以下の論理名を右図のように指定してください。

- Travel
- Travel_Pics



2. プロジェクトを開く

Magic eDeveloper V10 では、「プロジェクト」をもとにした開発を行います。本章では、プロジェクトとそれに関連する設定について説明します。

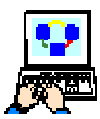
プロジェクトは、Magic の全リポジトリを含む、システム開発の単位です。V9Plus までは「アプリケーション」と呼んでいましたが、ひとつのまとまった仕事を行う「アプリケーション」は複数のコンポーネントから成ることがあるので、Magic の開発単位を「アプリケーション」という名称で呼ぶことは不適切であると考えられるようになりました。従って、V10 からはこれを「プロジェクト」と呼ぶようにし、「アプリケーション」という名前は、必要なコンポーネントまですべて含めた実行時の環境全体を指すようにしました。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

プロジェクトとアプリケーションの概念と取り扱い:	**プロジェクトとアプリケーション** の章
デフォルトプロジェクト:	**設定 → 動作環境 → [システム] タブ**

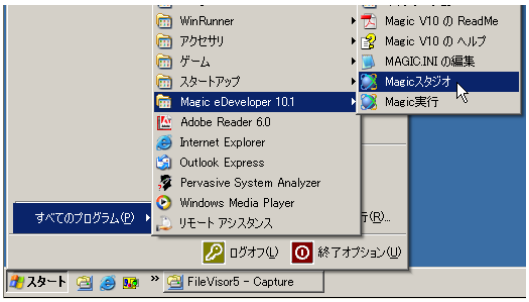
2.1. Magic スタジオを起動する

最初に、Magic スタジオを起動しましょう。



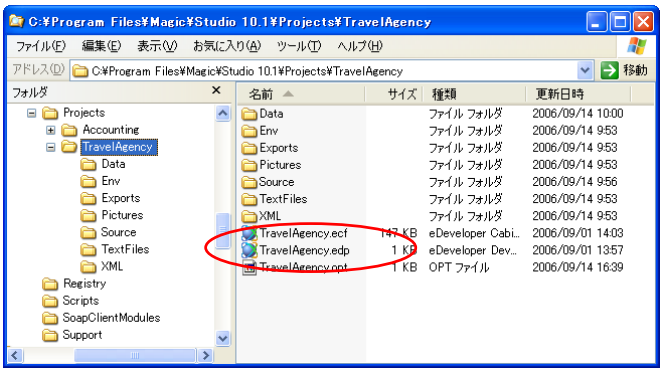
Magic スタジオを起動するには、いくつかの方法があります。

1. スタートメニューから
   すべてのプログラム(P)
   → **Magic eDeveloper 10.1**
   → **Magic スタジオ**
   を選びます。

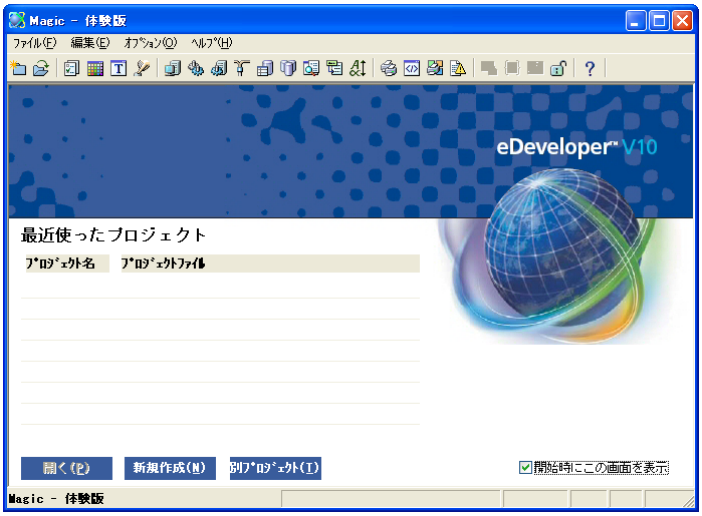


2. デスクトップ上の Magic スタジオ アイコンをダブルクリックしても起動できます。

3. Magic プロジェクトファイル (Projects¥TravelAgency¥TravelAgency.edp ファイル) をダブルクリックして開くこともできます。この方法については、2.3 既存のプロジェクトを開く で説明します。



Magic スタジオの起動初期画面は右図のようなものです。



2.2. プロジェクトのソースファイル

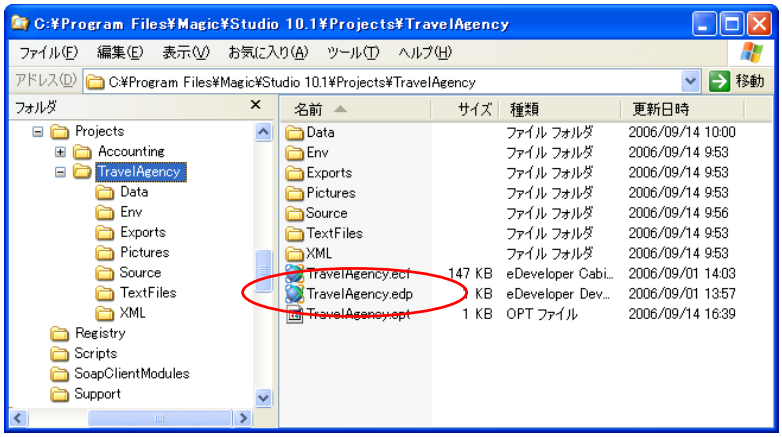
Magic V9Plus までは、プロジェクト(アプリケーション)を CTL ファイル(MCF ファイル)というバイナリの Pervasive ファイルとして格納していましたが、Magic スタジオ V10 は、プロジェクトの定義をソースファイルの形式で、リポジトリごと、プログラムごとに分けて管理します。

2.2.1. プロジェクトファイル

サンプルで具体的に見ていきましょう。前章で作成した、Projects¥TravelAgency ディレクトリの中身を見てください。ここには右図のようなファイルがあるはずです。

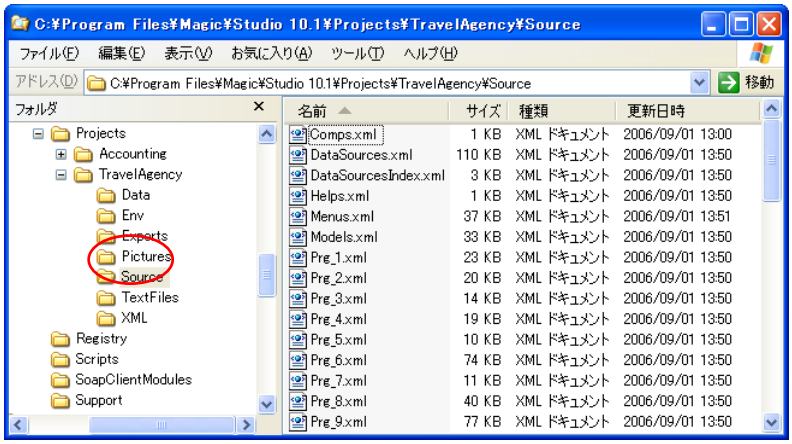
TravelAgency.edp というファイルが Magic V10 の **プロジェクトファイル** です。

プロジェクトファイル には、このプロジェクトに関する基本的な定義情報しか格納されていません。個々のリポジトリの定義情報は、Source サブディレクトリの下に格納されています。



2.2.2. リポジトリソースファイル

次に、Source サブディレクトリを開いてみてください。プロジェクトの個々のリポジトリ定義が、XML ファイルの形式で格納されています。右図に見るように、だいたいひとつのリポジトリに一つの XML ファイルが対応しています。ただし、プログラムリポジトリについては、一つのプログラムがひとつの XML ファイルに対応しています。

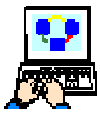


これらの XML ファイルはテキストファイルですが、テキストエディタなどで編集してはいけません。Magic のリポジトリ間の整合性が壊れてしまう可能性があります。

Source サブディレクトリに並んで、Exports サブディレクトリもありますが、これはリポジトリ出力を行う場合のデフォルトディレクトリで、初期状態では空です。

2.3. 既存のプロジェクトを開く

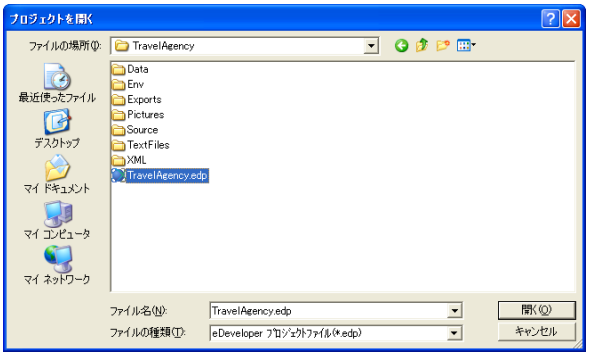
次に、TravelAgent プロジェクトを開いて、中身を見てみましょう。既存のプロジェクトを開きたい場合には、**ファイル → プロジェクトを開く** メニューで開きます。



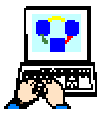
TravelAgent プロジェクトを開くには・・・

1. **ファイル** メニューから **プロジェクトを開く** を選択します。

2. **プロジェクトを開く** ダイアログが出るので、プロジェクトファイルを指定します。ここでは、Magic ディレクトリの下の Project¥TravelAgency¥TravelAgency.edp を開いてください。

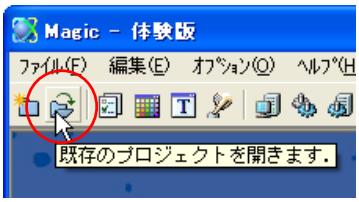


プロジェクトを開く メニューからプロジェクトファイルを選ぶのが基本ですが、その他にもプロジェクトの開き方があります。

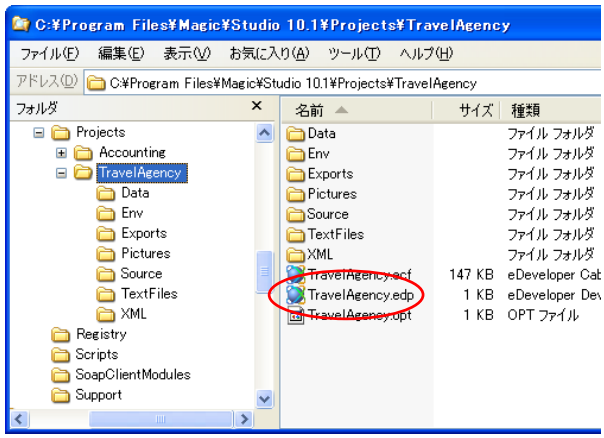


TravelAgent プロジェクトを開くには・・・（その他の方法）

- **ツールバーで開く: ファイル** メニューの代わりに、ツールバーからプロジェクトを開くこともできます。

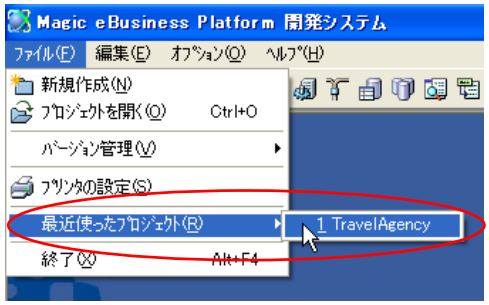


- **プロジェクトファイルのダブルクリックで開く**: インストーラは拡張子 .edp を Magic スタジオと関連付けるので、エクスプローラなどからプロジェクトファイル TravelAgency.edp をダブルクリックすると、Magic スタジオが起動し、TravelAgency プロジェクトが開かれます。



- **起動画面から開く**: Magic スタジオでは、最近使ったプロジェクトを記憶しています。一度 TravelAgency プロジェクトを開いたことがあれば、起動画面でこのプロジェクトが 最近使ったプロジェクト の一覧に表示されますので、それをダブルクリックして開くこともできます。
  ダブルクリックの代わりに、**Enter** キーを押したり、**開く(P)** ボタンを押しても、プロジェクトが開きます。

- **最近使ったプロジェクト メニューから開く**: 最近使ったプロジェクトは、**ファイル** メニュー → **最近使ったプロジェクト** にも登録されますので、ここから選択することもできます。



2.4. アプリケーション特性

アプリケーションを開いた状態で、アプリケーション特性を定義することができます。これはこのプロジェクトを実行する際の、全般的な設定に関わるパラメータを設定するものです。

アプリケーション特性は、**ファイル** → **アプリケーション特性メニュー** から開きます。

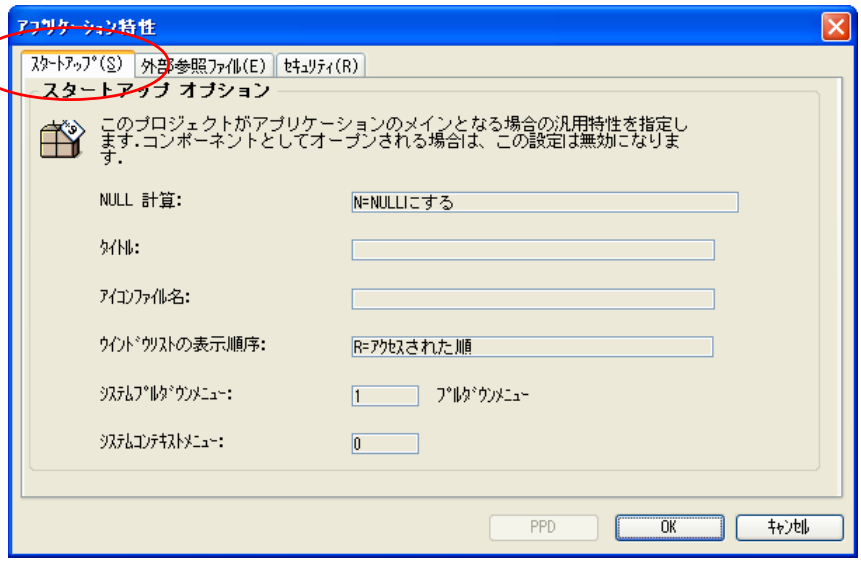
アプリケーション特性ダイアログには、

- **スタートアップ**
- **外部参照ファイル**
- **セキュリティ**

という 3 つのタブがあります。

2.4.1. スタートアップ

アプリケーション特性 ダイアログの **スタートアップ** タブは右図のようなものです。



このうち、**タイトル**、**アイコンファイル名**、**ウィンドウリストの表示順序**は V10 で新しく導入された特性で、**複数のタスクを同時に実行する**（第 14 章）で説明します。

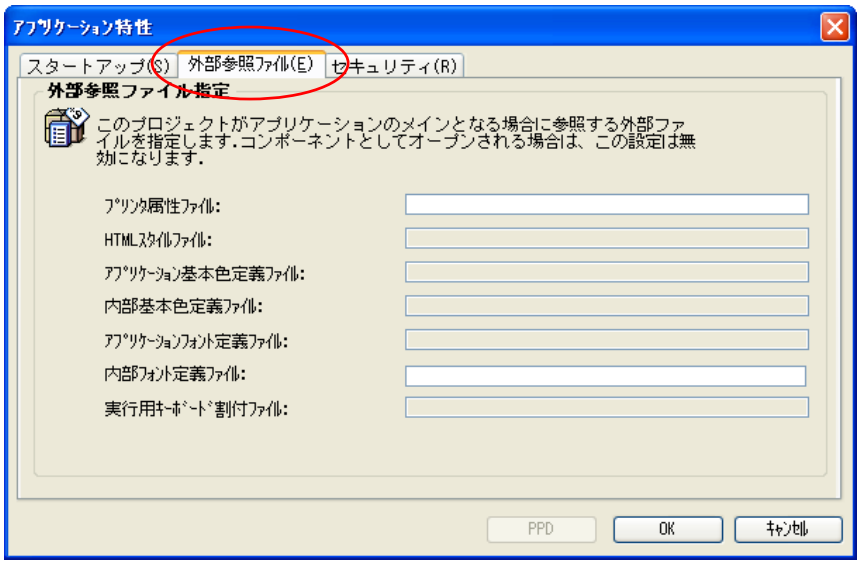
システムプルダウンメニュー と **システムコンテキストメニュー** も V10 で新しく追加された特性で、このプロジェクトが実行された場合のプルダウンメニューと、コンテキストメニュー（コンテキストメニュー)を指定します。

V9 Plus

V9Plus 以前では、プルダウンメニューとコンテキストメニューはメニューリポジトリの 1 番と 2 番とに固定されていましたが、V10 ではメニューリポジトリで定義されたメニューのいずれも利用することができます。

2.4.2. 外部ファイル

外部ファイル タブには、プロジェクトで参照する色、フォント、キーボード、その他の設定ファイルのパスを定義します。



色定義ファイルとフォント定義ファイル

V9Plus までは、色定義ファイルとフォント定義ファイルで定義された色とフォントは、開発画面、実行画面ともに共通のものでした。このため、アプリケーション実行時のために色やフォントの定義を変更すると開発画面での色やフォントも変わってしまう問題がありました。

V10 では色定義ファイルとフォント定義ファイルを目的によって次の 3 つに分けています。

	意味	色定義ファイル	フォント定義ファイル
アプリケーション	Magic アプリケーション実行時の画面用	clr_rnt.jpn	fnt_rnt.jpn
内部	Magic 実行版でのシステム画面用（アプリケーション画面以外の箇所）	clr_int.jpn	fnt_rnt.jpn
開発	Magic スタジオでの開発画面用	clr_std.jpn	fnt_std.jpn

このうち、アプリケーションと内部については、アプリケーション特性で設定することができます。

このように色定義ファイルとフォント定義ファイルを用途によって分けることにより、色やフォントのカスタマイズの自由度が上がりました。

V9 Plus

V10 での色定義ファイルとフォント定義ファイルは、V9Plus のものと形式が同じなので、そのまま利用することができます。

キーボード割り付けファイル

キーボード割付ファイルについては、V10 ではスタジオでの開発用と、実行用とに分けられました。

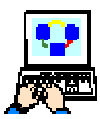
実行用のキーボード割付ファイルは、アプリケーション特性で指定することができます。

V10 でサポートされなくなったアプリケーション特性

- **EURO 関連機能**: V9Plus の EURO 関連機能は、V10 でサポートされなくなり、EURO 関連のプロパティ類も削除されています。アプリケーション特性では、**基準通貨** 指定および**通貨変換ファイル**などのプロパティが削除されています。
- **ワークグループ開発** タブ: チーム開発のために V9Plus では **ワークグループ開発** タブがありましたが、V10 ではバージョン管理システムを利用してチーム開発を行うため、**ワークグループ開発** タブがアプリケーション特性から削除されました。

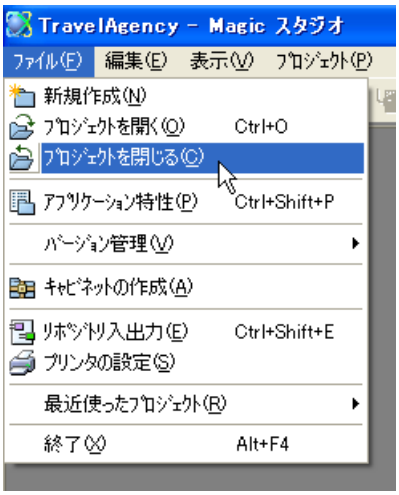
2.5. プロジェクトを閉じる

開発作業が終わったら、プロジェクトを閉じます。



開いたプロジェクトを閉じるには・・・

ファイル → **プロジェクトを閉じる** メニューを選びます。



ここでは、いったん TravelAgency プロジェクトを閉じて置いてください。プロジェクトに定義されている各リポジトリの内容は、次章以下で見ていきます。

2.6. 新しいプロジェクトを作成する

2.6.1. 新規プロジェクトの作成

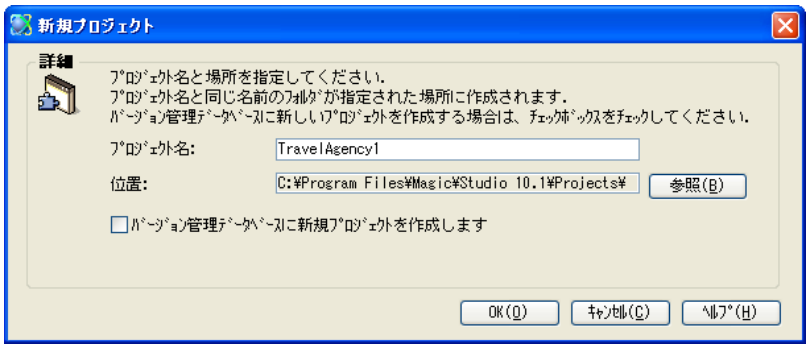
新規プロジェクトを作成するには、**新規プロジェクト** ダイアログを利用します。

新しいプロジェクトを作成するには・・・

1. **ファイル**メニューから、**新規作成** を選択します。

ウェルカム画面の「新規作成(N)」をクリックして、新しいプロジェクトを作成することもできます。

2. 新規プロジェクトダイアログボックスが開きますので、**プロジェクト名**を指定します。
3. **位置**には、プロジェクトファイルを格納しておくフォルダ名を指定します。デフォルトでは、Magic インストールディレクトリの下の Projects サブディレクトリになります。



4. OK ボタンを押すと、プロジェクトファイルが作成され、開かれます。

指定したディレクトリが存在しない場合には、確認ダイアログが出ますので、はい(Y) を選んで作成してください。

2.6.2. V9Plus との比較

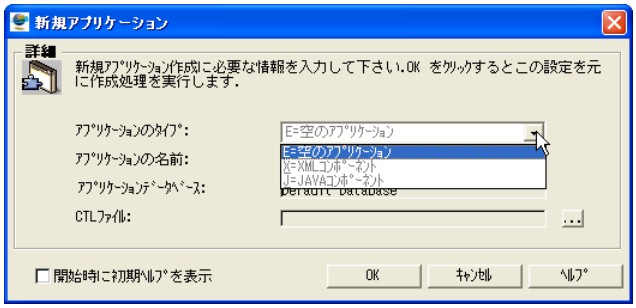
V9Plus での新規アプリケーション作成と、V10 での新規プロジェクト作成を比較してみると、次のような違いがあります。

アプリケーションタイプ

V9Plus では、**新規アプリケーション**ダイアログで、**アプリケーションタイプ**として、E=空のアプリケーション、X=XML コンポーネント、J=Java コンポーネント を選択することができました。

V10 では、アプリケーションタイプの選択はなくなり、つねに空のプロジェクトを作成します。

V9Plus の新規アプリケーションダイアログ

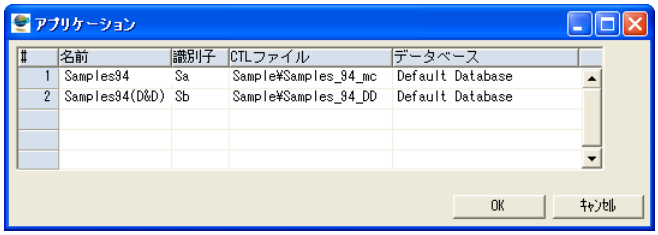


アプリケーションリポジトリ

V9Plus では、**アプリケーションリポジトリ**（**設定** メニュー → **アプリケーション**)があり、そこにアプリケーションを登録しておき、アプリケーションを開く場合にはその中から選択するようになっていました。

V10 では、プロジェクトをもとにして開発を行うため、アプリケーションテーブルはなくなりました。

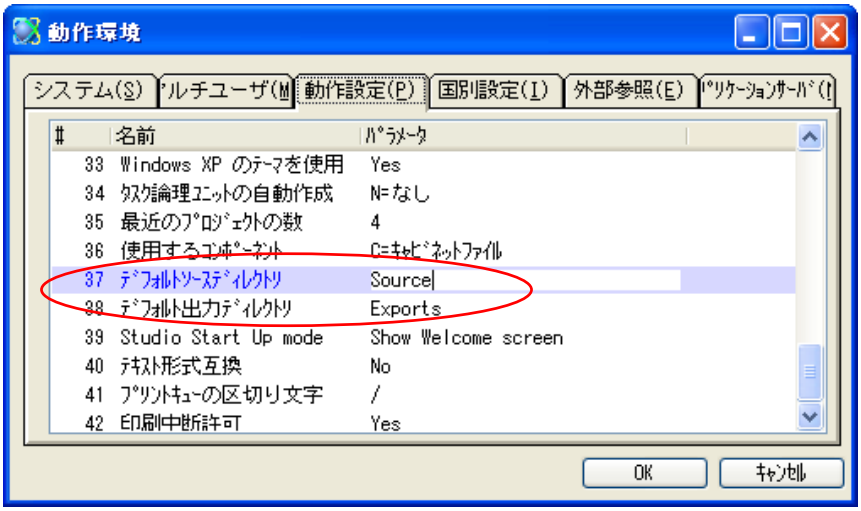
V9Plus のアプリケーションテーブル



2.6.3. ソースファイルのディレクトリ名を変える

新規プロジェクトを作成したときのソースディレクトリは、プロジェクトファイルの下の Source という名前のディレクトリですが、この名前は動作環境の設定により変更することが可能です。

動作環境ダイアログを開き、**動作設定(P)** タブの **デフォルトソースディレクトリ** で、作成されるサブディレクトリ名を指定します。デフォルトでは Source となっていますが、他の名前に変更することもできます。



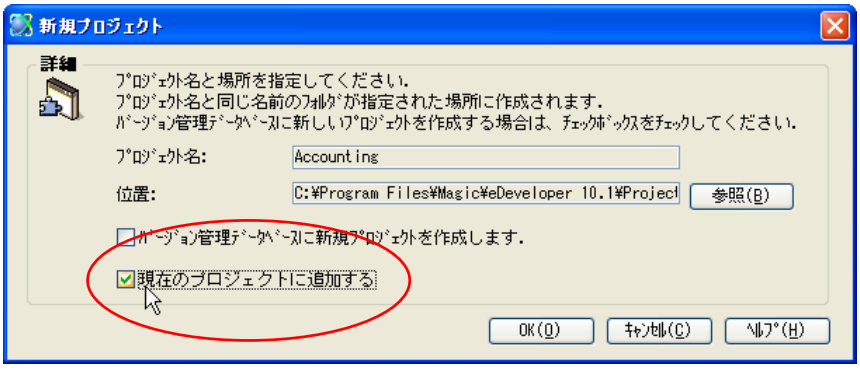
同様に、リポジトリ出力用ディレクトリのサブディレクトリ名も、**デフォルト出力ディレクトリ** の設定で変更することが可能です。

2.7. プロジェクトモジュールとナビゲーション

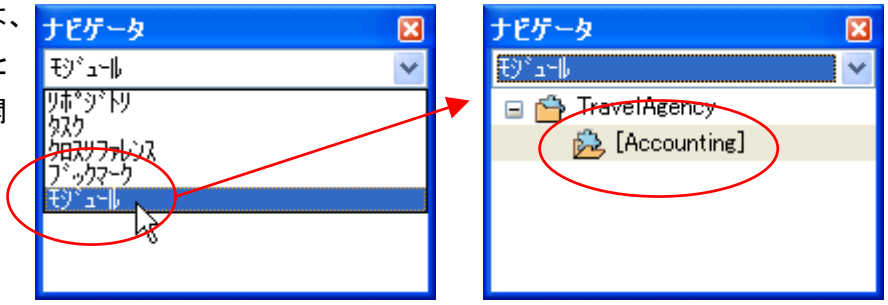
V10 では、**モジュール** という概念を導入しました。モジュールとは、あるプロジェクトに関連のあるプロジェクトを定義する機能で、コンポーネントや、相互に協調しながら動作する他のプロジェクトなどをまとめ、プロジェクトをすばやく開いたり、デバッグしたりすることができるようになります。

2.7.1. 新規プロジェクトをモジュールとして追加する

あるプロジェクトを開いている状態で、このプロジェクトに、新しいプロジェクトを作成して追加したい場合には、**ファイル** → **新規プロジェクト** ダイアログで、**現在のプロジェクトに追加する** にチェックして作成します。



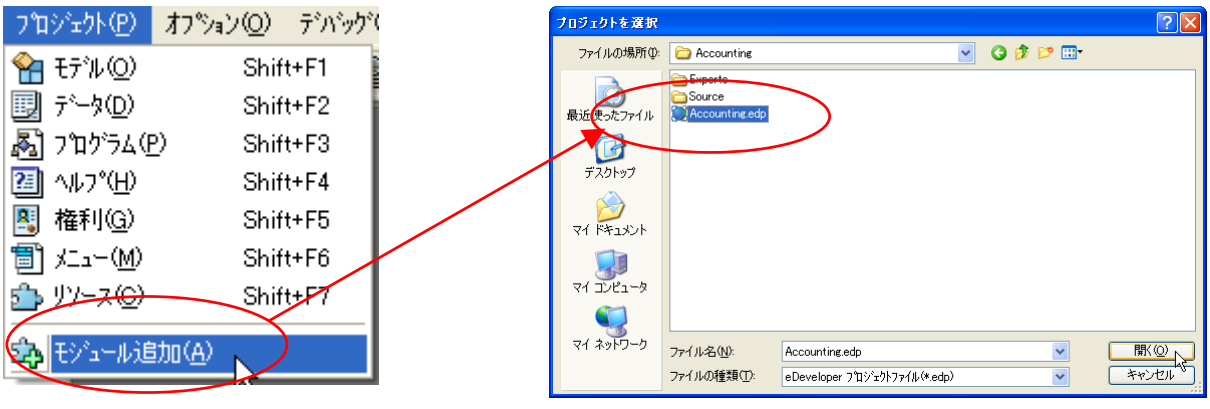
このようにして作成されたプロジェクトは、もとのプロジェクトに属するモジュールとして登録されます。モジュールの所属関係は、ナビゲータで **モジュール** を選択すると表示されます。



2.7.2. 既存のプロジェクトをモジュールとして追加する

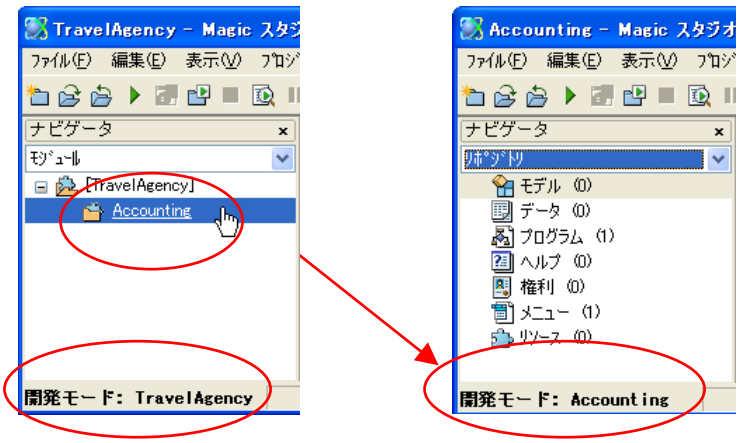
今開いているプロジェクトに、他の既存のプロジェクトをモジュールとして追加するには、**プロジェクト** → **モジュール追加** メニューで行います。

プロジェクトを選択 ダイアログが開くので、モジュールとして追加したプロジェクトファイルを選択します。



2.7.3. プロジェクト中のモジュールを移動する

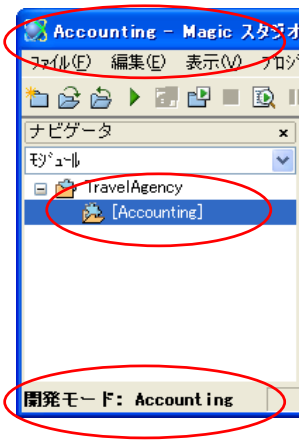
モジュールとして登録することの利点の一つは、プロジェクトを開く／閉じるが簡単に行えることです。
例えば、現在 TravelAgency プロジェクトを開いており、モジュールとして Accounting が登録されていたとします。このときに、Accounting プロジェクトを開いて編集したいと思う場合には、ナビゲータから Accounting をダブルクリックするだけで、開くことができます。
下図は、TravelAgency プロジェクトを開いているときに、モジュール一覧から Accounting をダブルクリックしたところです。TravelAgency が閉じて、Accounting プロジェクトが開いたことが、ステータスバーに表示されているプロジェクト名からわかります。



現在開いているプロジェクトは、

- ステータスバー
- タイトルバー
- ナビゲータ の **モジュール** 画面（かぎ括弧 [ ] により）

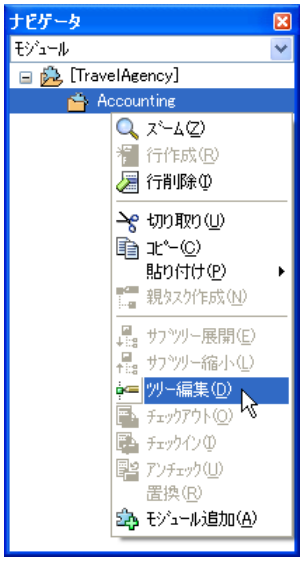
などに表示されます。

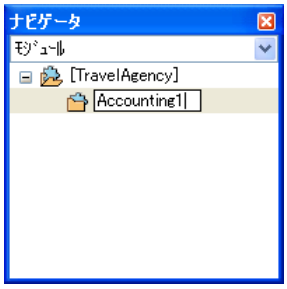


2.7.4. モジュールを名前変更・削除する

モジュールの名前を変更したり、登録してあるモジュールを削除したりすることもできます。
名前変更、削除の際には、現在開いているプロジェクトを対象にすることができません。例えば Accounting が開いている場合、Accounting を削除したり、名前を変更したりすることはできません。この場合には、いったん TravelAgency を開いて、その上で削除・変更するようにしてください。

- 名前を変更するには、ナビゲータ の モジュール表示で、変更したいモジュールを選択して、右クリックによりコンテキストメニューを出し、**ツリー編集** を選んでください。
  ノードが編集可能になりますので、適当な名前に変更してください。





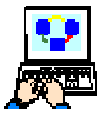
- モジュールを削除する場合には、同じく、ナビゲータでモジュールを選び、メニューで **行削除** を選んでください。

以下の説明では、Accounting プロジェクトは利用しないので、ここでモジュールから削除しておいてください。

モジュールを削除しても、登録情報が削除されるだけで、プロジェクトファイルそのものは削除されません。

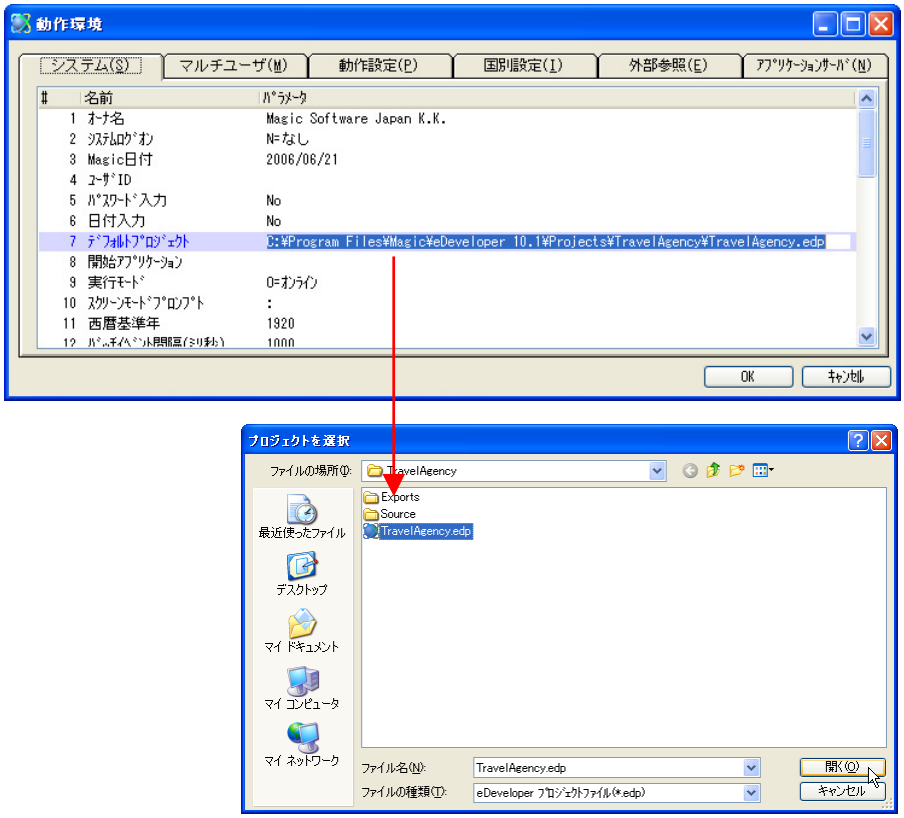
2.8. デフォルトプロジェクトを設定する

V9Plus での **デフォルトアプリケーション** 設定と同様に、Magic スタジオを開いたときに自動的に開くプロジェクトを指定することができます。V10 では、**デフォルトプロジェクト** と呼びます。



デフォルトプロジェクトを設定するには・・・

1. **オプション** → **設定** → **動作環境** ダイアログを開きます。
2. **システム** タブを開きます。
3. **デフォルトプロジェクト** パラメータからズームして、**プロジェクトを選択** ダイアログを開きます。
4. プロジェクトファイルを選択し、**開く** ボタンを押します。



- デフォルトプロジェクトには、論理名を使うことができます。
- デフォルトプロジェクトの設定は、Magic スタジオ(開発時)にだけ有効です。
- 実行版起動時に自動的に実行されるプロジェクトを指定する場合には、**開始アプリケーション** パラメータに設定します。

3. データリポジトリ

V9Plus でのテーブルリポジトリは、V10 では **データリポジトリ** と呼ばれるようになりました。これは、V10 で XML データソースを利用できるようになったため、ここで定義するデータがデータベースのテーブルに限らず、より一般化されたためです。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

モデルについて	**モデル** の章
データソースについて	**データソース** の章
データベース特性	**設定 → データベース**

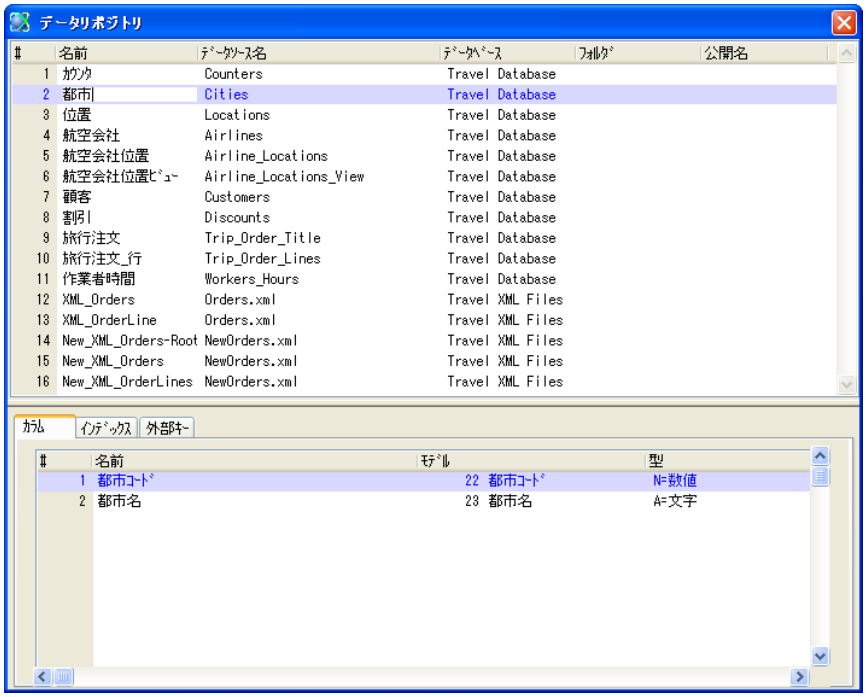
3.1. 新しいインターフェース

V10 では、データリポジトリのデザインが変更されています。ここで見るように、画面は上下二つの分割ウィンドウ(**ペイン**と呼びます)に分割されています。

- 上のペインにはテーブル／XML ファイルの一覧が表示されます。
- 下のペインには各テーブル／XML ファイルの詳細が表示されます。

下のペインには、カラム、インデックス、外部キーなどのタブがあります。

以下に、それぞれの部分について説明していきます。



3.1.1. 上のペイン

上のペインには、プロジェクトで利用するテーブルや XML ファイルの一覧が表示されます。

データベース としては、V9Plus と同じく、Pervasive、Memory、MS-SQL Server、Oracle、DB2 などの通常のデータベースのほかに、V10 で新たに XML ファイルをデータベースとして利用できるようになりました。

TravelAgency アプリケーションでは、11 番までが Pervasive 上に定義されたテーブルですが、12 番以降は XML ファイルです。XML ファイルのサポートについては、XML サポート(第 11 章)で詳しく説明します。

データソース名 には、Pervasive の場合には Pervasive ファイルのファイル名、SQL データベースの場合にはテーブル名、XML ファイルの場合には、XML のファイル名を指定します。

V10 では、データソース名を必ず指定しなければなりません。

3.1.2. 下のペイン

下のペインには、以下の 3 つのタブがあります。

- カラム
- インデックス
- 外部キー

これらのタブ間を移動するには、**Ctrl+Tab** キーを使います。

これらの内容は、V9Plus の場合とほとんど同じですが、カラムの定義で V10 で変更になった点について説明します。

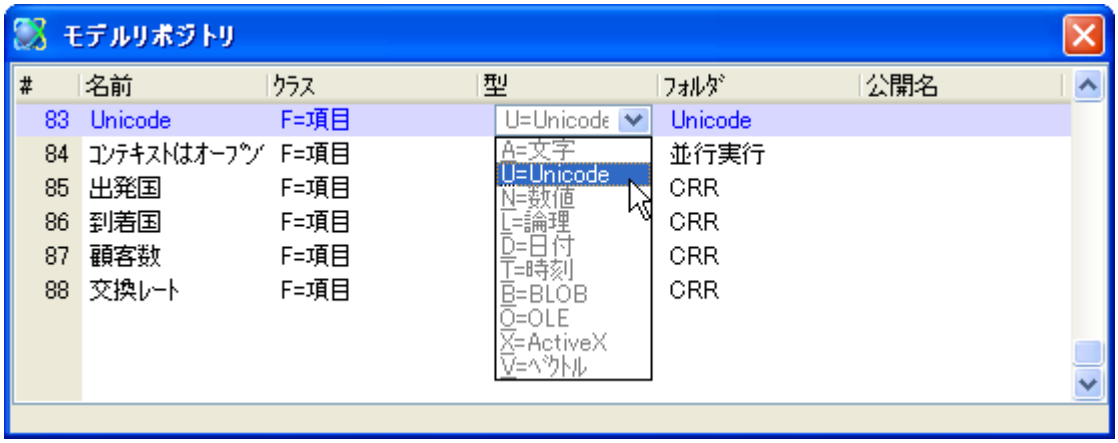
以下の説明は、データリポジトリの **カラム** 欄、モデルリポジトリ、およびプログラムリポジトリの変数やパラメータの定義においても同様です。

メモ属性

V10 では **メモ型** はなくなりました。V9Plus でメモ型として定義されていたカラムはすべて文字型となっています。ただし、V9Plus 以前で作成したデータと互換を保つため、**記憶形式** が String Memo あるいは Magic Memo となっています。

Unicode 属性

V10 で新たに **Unicode 型** がサポートされるようになりました。従来の文字型もそのままサポートされていますが、文字型は第二水準 Shift-JIS 規格で規定されている文字しか使えないのに対し、Unicode 型では、UNICODE 規格で規定されている文字が使えるようになります。

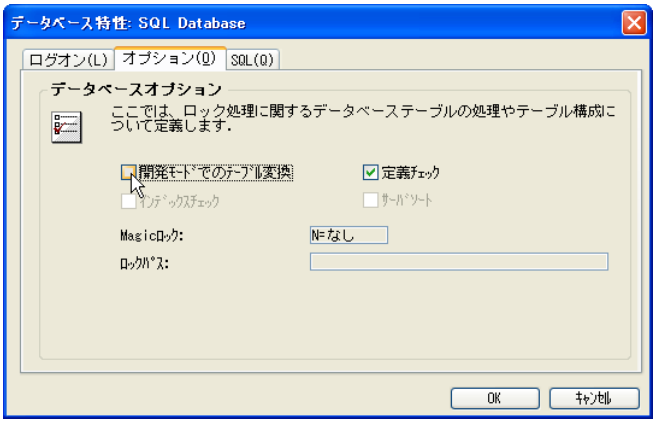


実際には、オペレーティングシステムがサポートしている UNICODE の規格により、Magic で使える文字が制限されます。

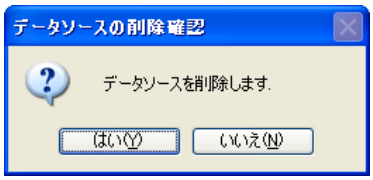
3.2. データソースを削除する

データリポジトリからデータソースを削除するには、V9Plus の場合と同じく、**編集** → **行削除** メニュー、あるい **F3** キー により行います。

このときに、DBMS 上に存在するテーブルも同時に削除するかどうかは、データベーステーブルの **開発モードでのテーブル変換** の指定と、確認ダイアログに対するユーザの返答により決まります。

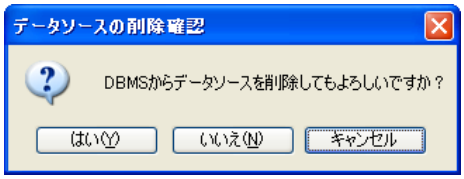


開発モードのテーブル変換 にチェックが入っていない場合には、Magic のデータリポジトリからデータソースが削除されても、DBMS 上のテーブルが削除されることはありません。データリポジトリから削除するかを確認するダイアログが表示されます。



- **はい** で答えると、Magic のデータリポジトリから削除されますが、DBMS 上のテーブルは残ります。
- **キャンセル** で答えると、何も起こりません。

開発モードのテーブル変換 にチェックが入っている場合には、Magic のデータリポジトリからデータソースが削除されるときに、V9Plus 以前では、DBMS 上のテーブルも自動的に削除されました。V10 では、自動的に削除するのではなく、削除するかどうかを聞くダイアログが表示されます。



- **はい** で答えると、Magic のデータリポジトリから削除されるとともに、DBMS 上のテーブルも同時に削除されます。
- **いいえ** で答えると、Magic のデータリポジトリから削除されますが、DBMS 上のテーブルは残ります。
- **キャンセル** で答えると、何も起こりません。

4. タスクエディタ

V10 では、タスクエディタのユーザインターフェースが大幅に改善されていて、入力が容易になり、表示も整理されてタスク全体の見通しがよくなっています。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

プログラムリポジトリ	**プログラム** の章
データビュー エディタ	**データビュー エディタ** の章
ロジック エディタ	**ロジック エディタ** の章
フォーム エディタ	**フォーム エディタ** の章
範囲 ダイアログボックス	**プログラム → 範囲ウィンドウ**
クイック式入力、 式エディタの強化	**式エディタ** の章

4.1. プログラムリポジトリ

まずは、TravelAgency プロジェクトのプログラムリポジトリを開いてください。

プログラムリポジトリを開くには・・・（V9Plus と同様です）

次のいずれかによって、プログラムリポジトリが開きます。

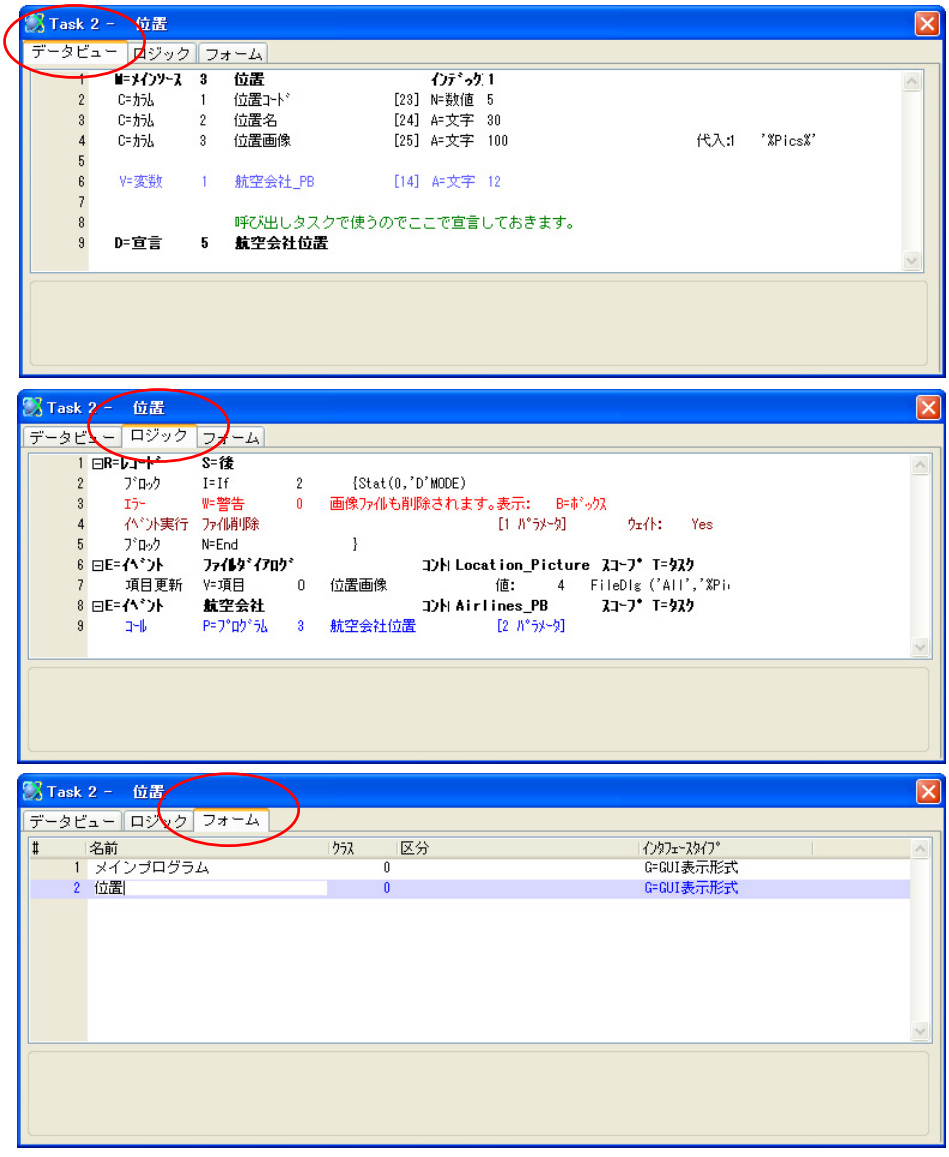
- ナビゲータから開く
- **プロジェクト** → **プログラム** メニューから開く
- **Shift＋F3** キーで開く
- ツールバーから開く

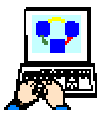
プログラムリポジトリから、プログラム 2 番 **位置** を開いてください。

ここで見るように、タスクエディタは

- データビュー
- ロジック
- フォーム

の三つのタブがあります。





これらのタブ間を移動するには、マウスでクリックするほかに、次のようなショートカット キーを使うこともできます。

- Ctrl+1: データビュー エディタを開きます
- Ctrl+2: ロジックエディタを開きます
- Ctrl+3: フォーム エディタを開きます

以下にこれらについて説明していきます。

4.2. データビュー エディタ

データビューは、V9Plus 以前のレコードメインに相当するもので、このタスクで利用するデータビューを定義します。例として、メインソースとリンクデータを含む、プログラム 6 番 **注文** を開いて、**データビュー** エディタを開いてみてください。

4.2.1. データビューでのコマンド

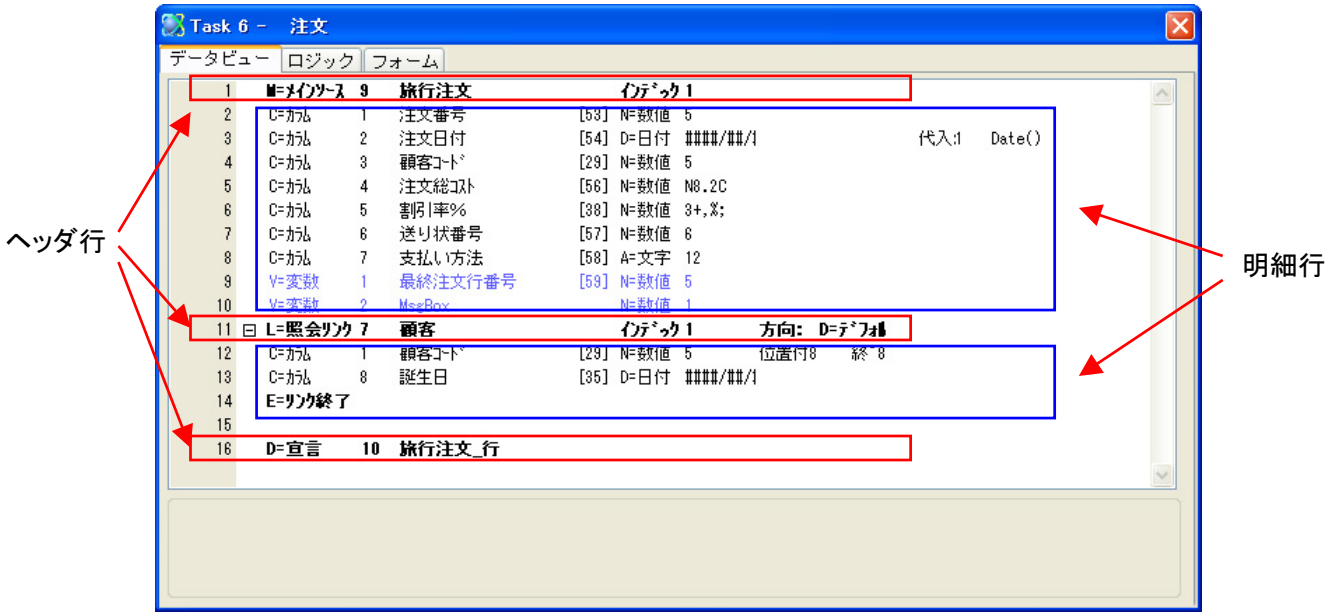
V9 Plus

データビューでのコマンドは、V9Plus と比較すると、次のように対応しています。

V10	V9Plus	レベル
メインソース	メインテーブル（タスク特性）	ヘッダ行
Q=SQL コマンド	埋め込み SQL	
L=照会リンク	リンク Q=照会 コマンド	
W=書込リンク	リンク W=書込 コマンド	
C=登録リンク	リンク C=登録 コマンド	
I=結合リンク	リンク J=結合 コマンド	
O=外部リンク	リンク O=外部結合 コマンド	
D=宣言	DB テーブルでの追加テーブル	
C=カラム	セレクト R=実データ コマンド	詳細行
V=変数	セレクト V=変数 コマンド	
P=パラメータ	セレクト P=パラメータ コマンド	

データビューで注意すべきことは、ここで定義される行は **ヘッダ** 行と、それに従属する **詳細** 行との階層的なグループを持っているということです。上の表で、**レベル** 欄に **ヘッダ行** と示したコマンドがヘッダ行となり、**詳細行** と示したコマンドが、それに従属する詳細行となることを示しています。

画面上では、ヘッダ行は太字で表示されます。



上の例で見てみると、1 行目は**メインソース**を設定する行で、ここではデータリポジトリの第 9 番目に定義されている

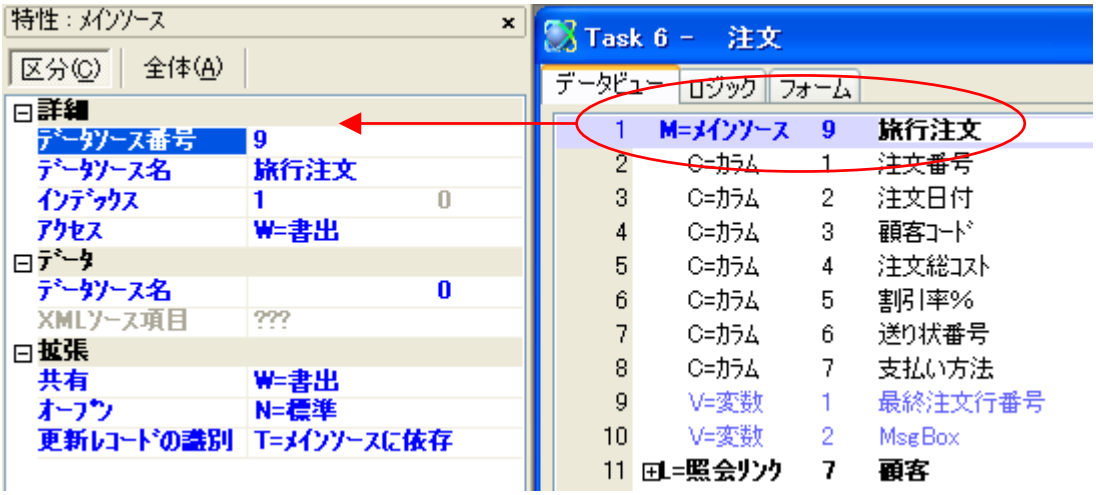
旅行注文 テーブルがメインソースとして設定されています。
それに続く **C=カラム** の行は、メインソース行に従属する詳細行として、メインソースの **旅行注文** テーブルのカラム(注文番号、注文日付、・・・) が選択されます。
第 11 行目は、**L=照会リンク** の行で、これはヘッダ行なので、新しいグループが始まります。このリンクコマンドでは、データリポジトリの第 7 番の **顧客** テーブルが選択されています。従って、この後に続く **C=カラム** の行は、これに従属する行として、**顧客** テーブルのカラム (顧客コード、誕生日)が選択されることになります。

4.2.2. 特性

データビュー各コマンドの詳細設定は、 **特性シート** で設定します。特性シートの内容は、コマンドにより変わります。

メインソース行

メインソース 行にカーソルがある場合、特性シートは下図のようになります。



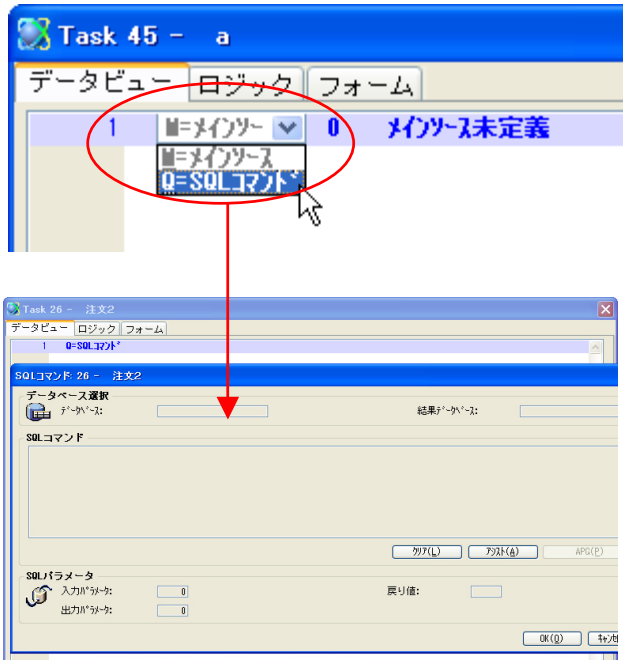
ここで設定する内容は、V9Plus では、**タスク特性**と **DB テーブル**で設定していた内容に対応します。

- V10 では **メインソース** 行、あるいは特性シート上で、**データソース名** を変更することができます。この機能は、同一のデータソースがメインソースやリンクで利用されていたとき、項目一覧上で区別するためなどに利用することができます。
- データソース名を変更しても、実際にアクセスされるファイルやテーブルが変わるわけではなく、表示上だけの変更となります。実際にアクセスされるファイルやテーブルの名前を指定したい場合には、 **データソース名** 特性で指定します。

Q=SQL コマンド

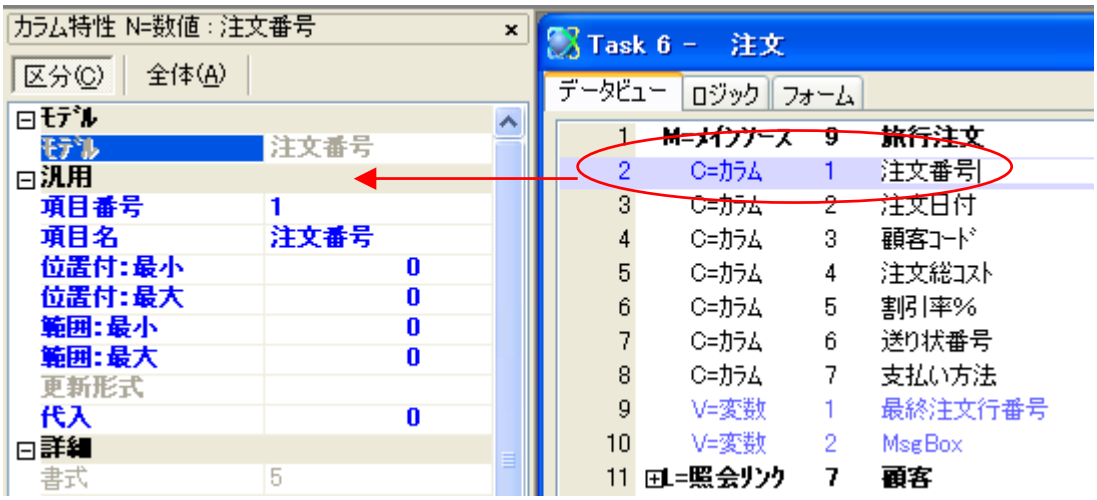
M=メインソース の行は、**Q=SQL コマンド** に変更することもできます。これは、V9Plus で埋め込み SQL タスクを定義することに相当します。

M=メインソース を **Q=SQL コマンド** に変更すると、SQL コマンド ダイアログが表示されます。SQLコマンド ダイアログは、V9Plus と同様です。



C=カラム 行

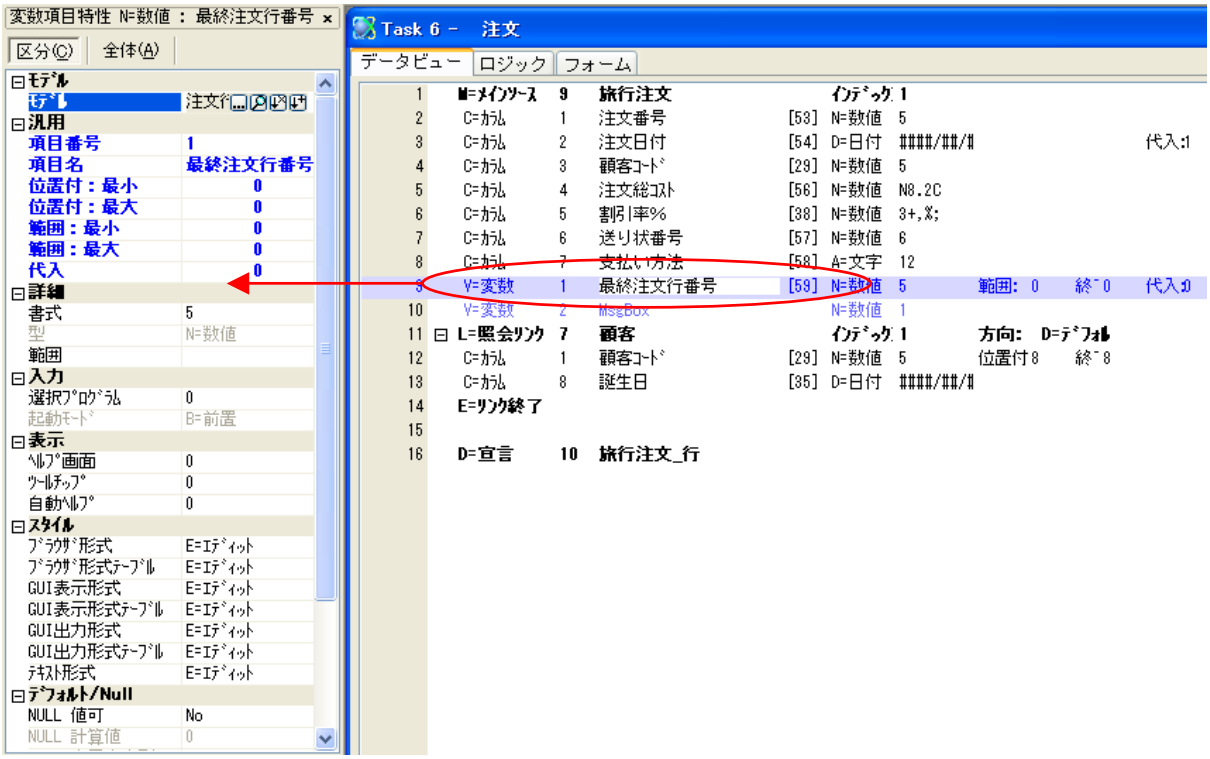
C=カラム 行にカーソルがある場合、特性シートは下図のようになります。ここで指定できる設定は、V9Plus では **セレクト** コマンドおよび **セレクト特性** ダイアログで設定していた内容となります。



C=カラム で選択したカラムの名前も、ここで変更することができます。これも同じ名前のカラムがある場合に区別するため、あるいはフォーム上でのラベルを変えたい場合などに利用することができます。

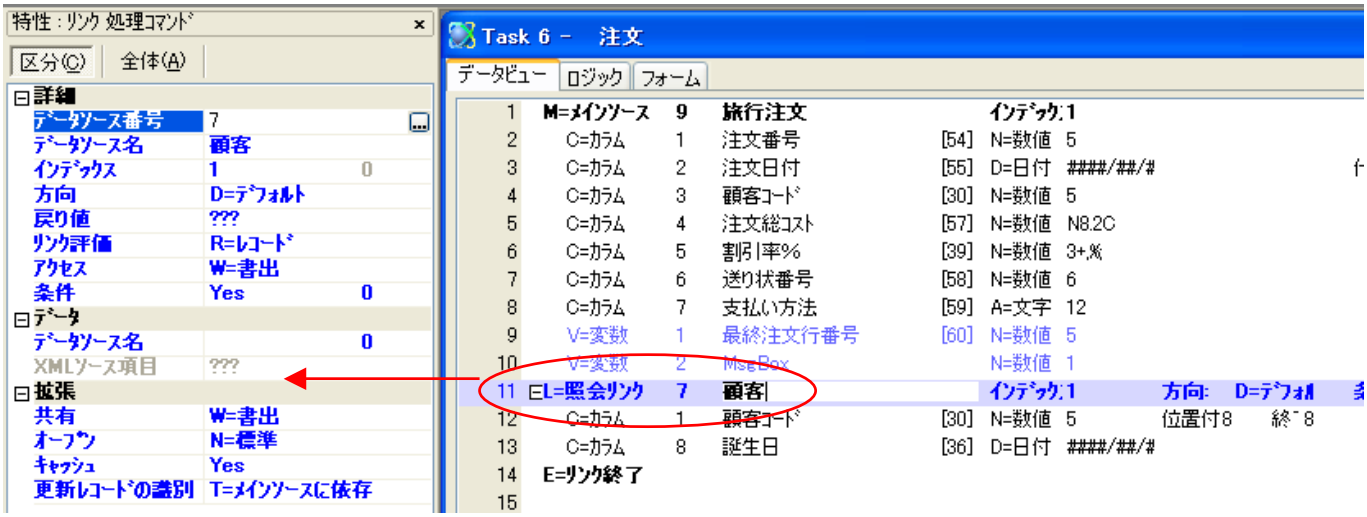
V=変数 および P=パラメータ 行

カーソルが **V=変数** 行、あるいは **P=パラメータ** 行にある場合の特性シートは以下のようなものです。実際の内容は、変数項目のデータ型により異なります。ここで指定できる内容は、V9Plus では、**セレクト** コマンド、あるいは変数テーブルで指定していた内容となります。



リンク行

カーソルが **L=照会リンク**、**W=書き込みリンク**、**C=登録リンク**、**I=結合リンク**、**O=外部リンク** などの **リンク** コマンド上にある場合には、特性シートは次のようになります。

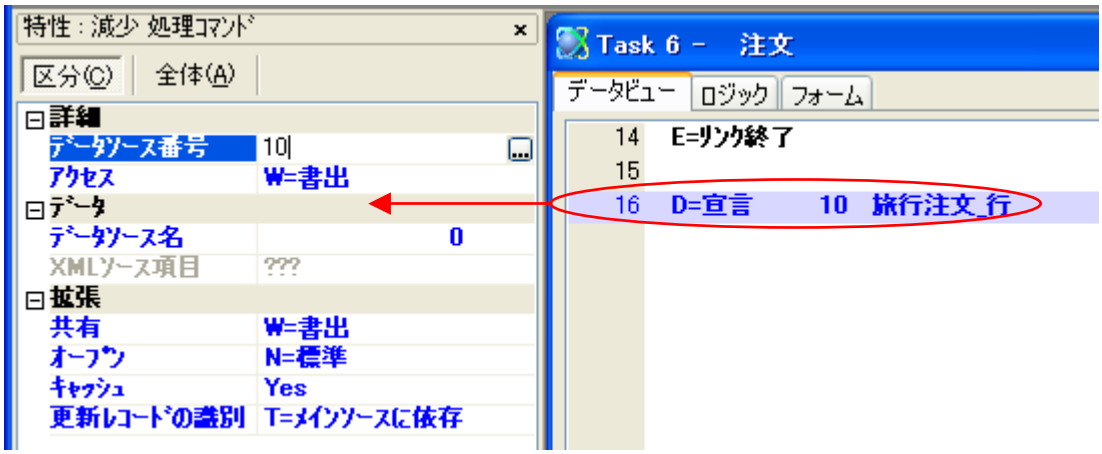


V9 Plus

- ここで指定する内容は、V9Plus では **リンク** コマンド、**リンク特性**、あるいは **DB テーブル**で指定していた内容に対応します。
- **方向** パラメータは、V9Plus では **A=昇順** および **D=降順** でしたが、V10 では **D=デフォルト** および **R=逆方向** となっています。V9Plus のとき、インデックスの順番が **D=降順** に設定されていて、リンクの **方向** が **D=降順** に設定されていた場合には、実際には昇順でリンクレコードが検索されていましたので、名前と実行結果とが異なる結果となっていました。V10 ではこれを正すために、**D=デフォルト** と **R=逆方向** としました。

D=宣言

V9Plus にはなかったコマンドとして、**D=宣言** というものがあります。これは、このタスクでは利用しないが、サブタスクなどで利用することがわかっているテーブルについて、予めここでオープンしておく場合に利用します。



V9 Plus

D=宣言 の特性は、V9Plus で DB テーブルで設定していた内容と同じです。

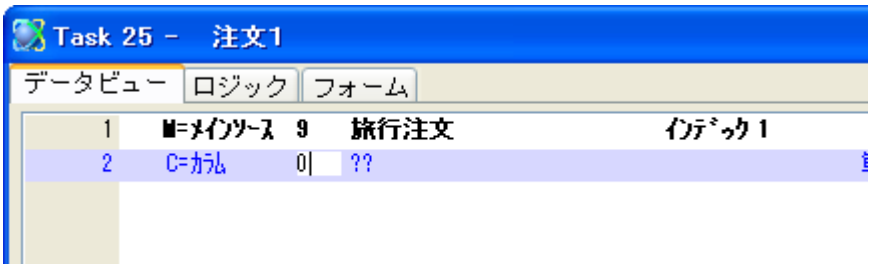
4.2.3. 複数一括選択

カラムの一括選択

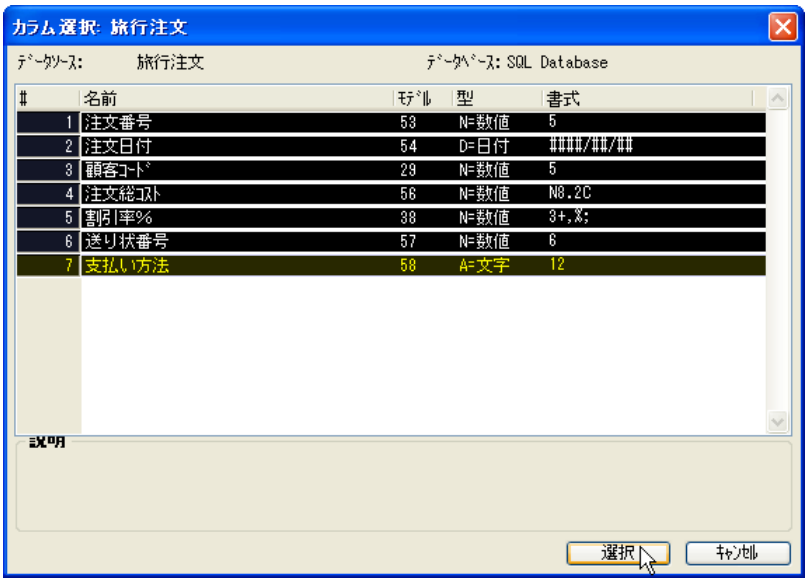
効率向上のため、V10 のデータビュー画面では、カラムを一括して複数選択することができるようになりました。実際に操作してみてください。

カラムを一括選択する方法（旅行注文テーブルの全カラムを選択する例）

1. プログラムリポジトリで新規プログラムを作成して、開いてください。名前は「注文1」などとしてください。
2. データビューを開き、メインソースとして 9 番（旅行注文)を選びます。
3. **F4** で 1 行作成し、C=カラム を選んでください。



4. Tab で次の欄に進み、ズーム します。カラム一覧が表示されます。
5. ここで、全カラムを選択します。複数カラムの選択は、Shift キーを押しながらマウスクリックします。



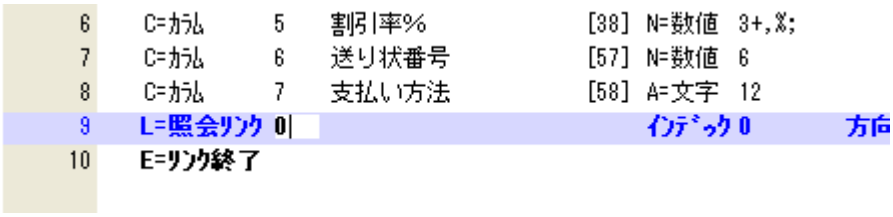
6. **選択** ボタンを押すと、全カラムについて **C=カラム** コマンドが作成されています。

リンクするデータソースの一括選択

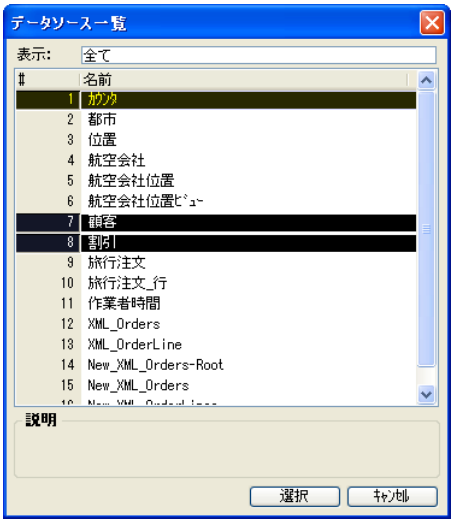
複数のテーブルをリンクする場合、リンクコマンドのデータソース欄から複数データソースを選択することもできます。

複数のデータソースをリンクする方法（カウンタ、顧客、割引 のデータソースをリンクする例）

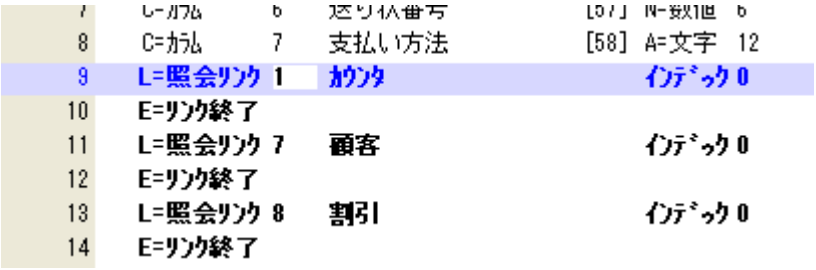
1. データビューに新しいヘッダ行を作成します。ヘッダ行は、Ctrl+H で作成します。



2. Tab キーで次の欄に移り、ズームすると、**データソース一覧**が表示されます。
3. 一覧から、**カウンタ**、**顧客**、**割引** を選択します。個々に複数選択する場合には、Ctrl キーを押しながらマウスクリックします。



4. 選択 ボタンを押します。データビューに、3つの L=照会リンク コマンドが作成され、それぞれに選択したデータソースが設定されます。



ここで、カラムの一括選択(45 ページ)に書いた手順で、各リンクコマンドに対し、全カラムを **C=カラム** コマンドで選択してください。

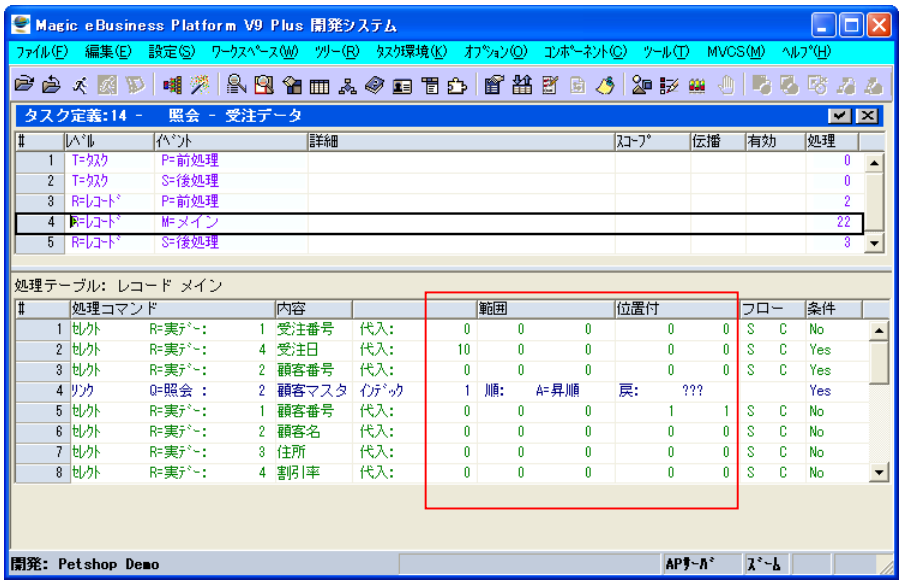
4.2.4. データビュー画面の表示

デフォルト値は非表示

V10 のデータビュー画面は、V9Plus のレコードメインとは異なり、設定されていない(デフォルトのままの)欄の値は表示しません。これにより、画面がすっきりとして、読みやすくなりました。

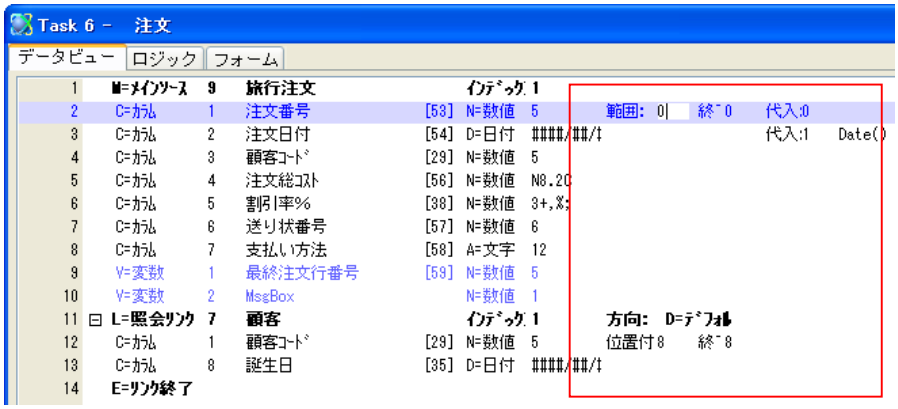
V9Plus のタスクエディタ

例えば、V9Plus の場合、実項目を定義する **セレクト R=実項目** コマンドでは、代入式、範囲小・大、位置付小・大、および条件式が常に表示されていました。しかし、実際には多くの場合それらの値は 0（未指定)でした。



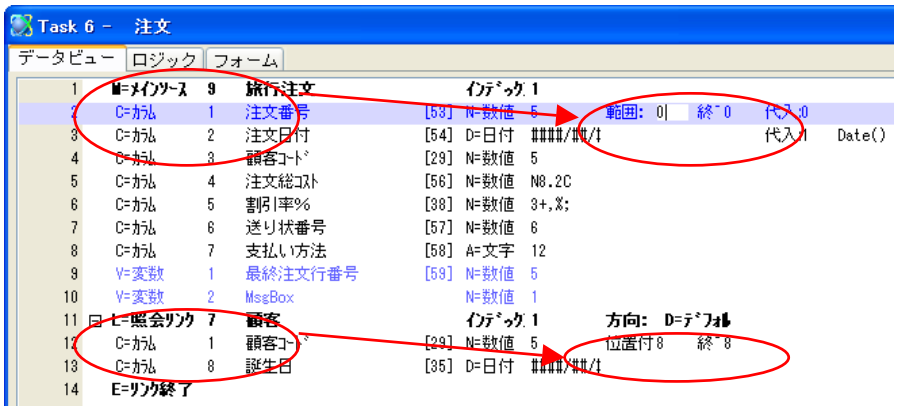
V10 のタスクエディタ

V10 では、実項目を定義する **C=カラム** 行では、未指定の欄は表示されません。ただし、下図の 2 行目にあるように、現在カーソルのある行については、未設定であっても全項目が表示されます。



範囲と位置付指定

一般的に言って、Magic のプログラムでは、メインデータソースには範囲付けでデータを絞り込み、リンクデータソースは位置付でレコードを特定します。このため、データビューエディタの表示上でも、メインデータソースのカラム定義には範囲設定を、リンクデータソースのカラム定義には位置付を表示します。

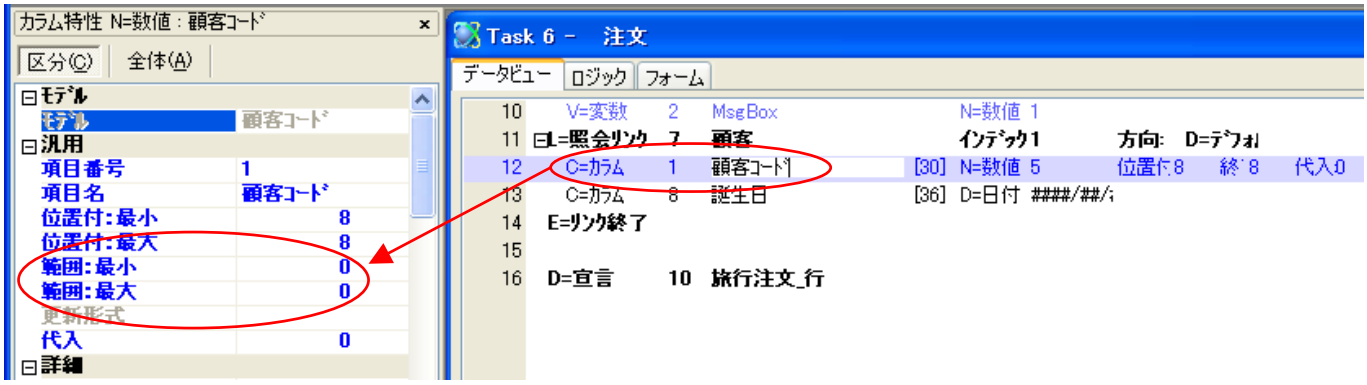


例として、右図(注文タスクのデータビュー)の 2 行目と 13 行目を見てください。

2 行目は、メインソース(旅行注文)のカラム **注文番号** を定義するものです。ここには、**範囲**（範囲開始）と **終了** とが表示されています。

12行名は、同じく **C=カラム** ですが、リンクデータソース **顧客コード** のカラムを定義しているものです。ここでは、**位置付**（位置付開始）と **終了** とが表示されています。

もし、メインデータソースで位置付を定義したい場合には、特性シート上で定義します。また、リンクデータソースのカラムに範囲付け指定を指定したい場合にも、特性シート上で定義します。



4.3. ロジックエディタ

ロジックエディタは、データなどに対する操作を行う **ハンドラ** を定義します。下図はプログラム6 **注文** のロジックエディタです。

4.3.1. ロジックエディタの構成

ロジックエディタの画面も、**ヘッダ行** と **詳細行** という階層構造をしています。ヘッダ行はハンドラの種類を定義し、詳細行はそのハンドラの中で実行するコマンドを定義します。

行の種類	用途	作成方法（ショートカットキー）
ヘッダ行	ハンドラの先頭を定義します。	Ctrl+H
詳細行	各ハンドラで実行すべきコマンドを定義します。	F4

4.3.2. ハンドラの種類

ロジックエディタに定義するハンドラには次の種類があります。

レベル	ハンドラ	備考
タスク	前処理	
	後処理	
レコード	前処理	
	後処理	
グループ	前処理	バッチタスクのみ
	後処理	
コントロール	前処理	オンラインタスクのみ
	後処理	
	検証	
イベント		
項目	変更	オンラインタスクのみ
関数		

ロジックエディタでは、V9Plus と比べると、次の点が異なります。

- **レコード メイン** はありません。**データビュー** がレコード メインの代わりとなっています。
- **コントロール** レベルの **変更** ハンドラがなくなり、代わりに **項目変更** という、より一般化したハンドラができました。
- **関数** という、新しい種類のハンドラができました。これはユーザ独自の関数を定義します。

4.3.3. コマンドの種類

V10 のコマンドは、V9Plus と基本的に同じですが、概念的に整理されています。

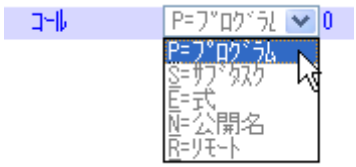
V9Plus 以前のコマンド	V10 のコマンド	補足
コメント	コメント	変更なし
項目更新	項目更新	変更なし
コール（プログラム、サブタスク、式、公開名、リモート）	コール	V10 のコールコマンドは、Magic プログラムを呼び出す場合に使う
コール（UDP, Web サービス、COM）、OS コマンド	外部コール	Magic 以外のものを呼び出す場合に使う。
アクション	アクション	変更なし
ブロック、 ブロック終了	ブロック	V9Plus 以前の **ブロック終了** は、**ブロック** の **End** タイプとなった。

エラー	エラー	変更なし
データ入力、データ出力	フォーム	フォーム コマンドのタイプとして、**入力**、**出力**を指定するようになった。
イベント実行	イベント実行	変更なし
セレクト 変数	項目	イベントハンドラ、項目変更ハンドラで、パラメータやローカル変数を定義する場合に用いる。
セレクト（カラム、変数、パラメータ）	（データビュー）	データビュー エディタで、カラム、変数、パラメータとして定義する。
リンク	（データビュー）	データビュー エディタで、リンクコマンドとして定義する。
エディット	（対応なし）	V10 で廃止

コール と 外部コール コマンド

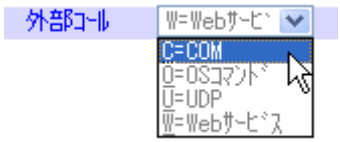
V10 では、**コール** と **外部コール** という 2 種類のコマンドがあります。

- **コール** コマンドは、他の Magic プログラムを呼び出す場合に使います。**コール** コマンドの**タイプ**としては、以下のものがあります。



P=プログラム	同一プロジェクト、あるいはコンポーネント中の他のプログラムを呼び出します。
S=サブタスク	自分の下に定義されているサブタスクを呼び出します。
E=式	プログラム番号を数値型で指定してプログラムを呼び出します。
N=公開名	コンポーネントリポジトリに定義されていないコンポーネントを、プロジェクトファイル名とプログラムの公開名を直接指定することにより、呼び出します。
R=リモート	リモートマシン上で動作している、Magic アプリケーションサーバ上のプログラムを、MRB のサービス名とプログラムの公開名を指定することにより呼び出します。

- **外部コール** コマンドは、**Magic 以外**のモジュールを呼び出す場合に使います。これには、以下のものが該当します。



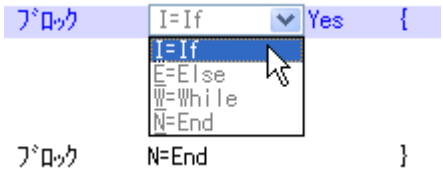
COM	COM モジュールを呼び出します。ActiveX 型の変数と、メソッド名を指定します。
OS コマンド	OS レベルで実行する実行可能モジュール（EXE や BAT ファイル）を実行します。

UDP	DLL として定義された外部ユーザ定義関数を実行します。DLL のファイル名と、公開されている関数名を指定します。
Web サービス	Web サービスとして公開されているサービスを呼び出します。

ブロック コマンド

ブロックコマンドは基本的に V9Plus と同じですが、細かな変更があります。

- V9Plus の **ブロック ループ** は、**ブロック While** と名前が変わりました。
- V9Plus の **ブロック終了** コマンドは、**ブロック End** となりました。



フォーム コマンド

V9Plus の **データ入力**と**データ出力**コマンドは、V10 では **フォーム** という一つのコマンドになりました。**フォーム** コマンドのタイプとして、**入力** か **出力** かを指定するようになっています。

V9Plus の セレクト コマンド

V9Plus の セレクト コマンドは、いくつかの機能を持っていましたが、V10 では次のように整理されています。

V9Plus のセレクトコマンドの機能	V10	
	利用場所	方法
レコードメインで、実項目、変数項目、パラメータ項目を定義する。	データビュー 画面	**C=カラム**、**V=変数**、**P=パラメータ** 行として定義する。
イベントハンドラ内で、イベントのパラメータ、あるいはハンドラ内でのみ用いる変数項目を定義する。	ロジック エディタの、イベントハンドラ	**項目** コマンドとして定義する。
(V10 新機能)	項目変更 ハンドラ	**項目** コマンドとして、パラメータなどを定義する。

4.3.4. ハンドラの順序

V10 の動作エンジンはイベントドリブン型なので、ロジックエディタで定義されているハンドラの順序は、基本的に実行に影響を与えません。そのため、開発者が見てわかりやすいよう、自由な順序で定義して構いません。

ただし、次のハンドラは、定義されている順序によって実行時の動作が変わります。

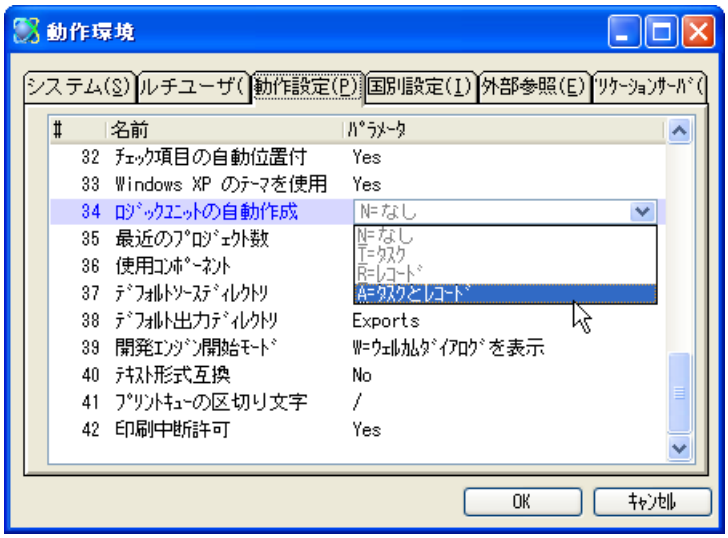
- **グループハンドラ**: 二つ以上の変数に対してグループ化する場合、ロジックエディタで上に定義されているグループハンドラの方が上位のグループと解釈されます。
- **イベントハンドラ**: あるイベントが発生した場合、イベントハンドラは、ロジックエディタで下に定義されているものから上に向かって検索が行われます。

4.3.5. ハンドラの自動作成

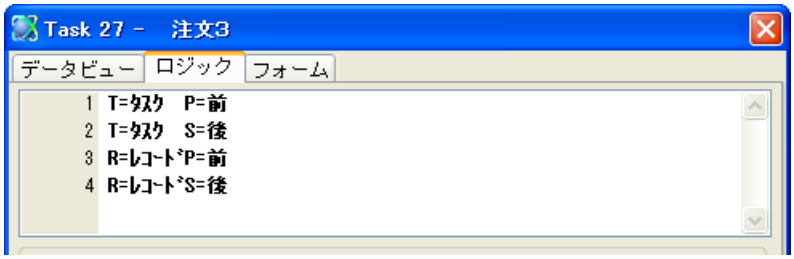
内容が空のハンドラは、ロジックエディタで定義しておく必要はありません。従って、新規にプログラムを作成した直後では、ロジックエディタの内容は空になっていて、ハンドラが一つも定義されていません。
しかし、プログラム開発の便宜のため、新規にプログラムを作成したときに、タスクレベル、レコードレベル、あるいはその両方のハンドラをデフォルトで作成するように設定することもできます。

新規タスクで作成されるデフォルトハンドラを指定するには・・・

1. **動作環境** ダイアログを開き、**動作設定(P)** タブを開きます。
2. **ロジックユニットの自動作成** を **T=タスク**、**R=レコード**、あるいは **A=タスクとレコード** のいずれかに選びます。
3. **OK** で動作環境ダイアログを閉じます。

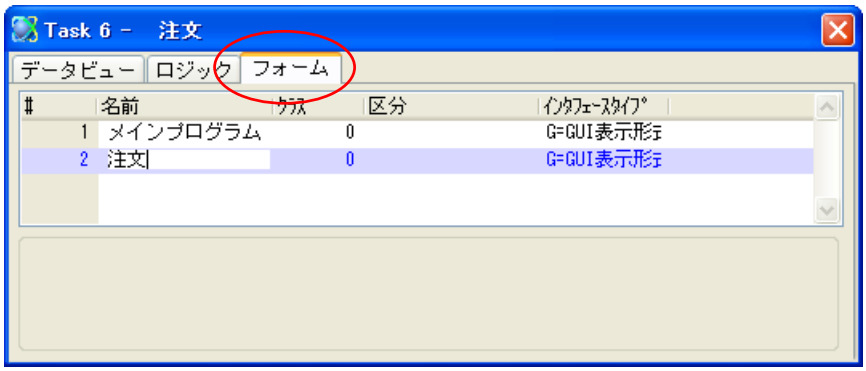


このように設定した後では、プログラムリポジトリで新しくタスクを作成した場合、設定した値に従って、タスク、レコード、あるいはその両方のレベルの 前処理・後処理 ハンドラが自動的に作成されます。

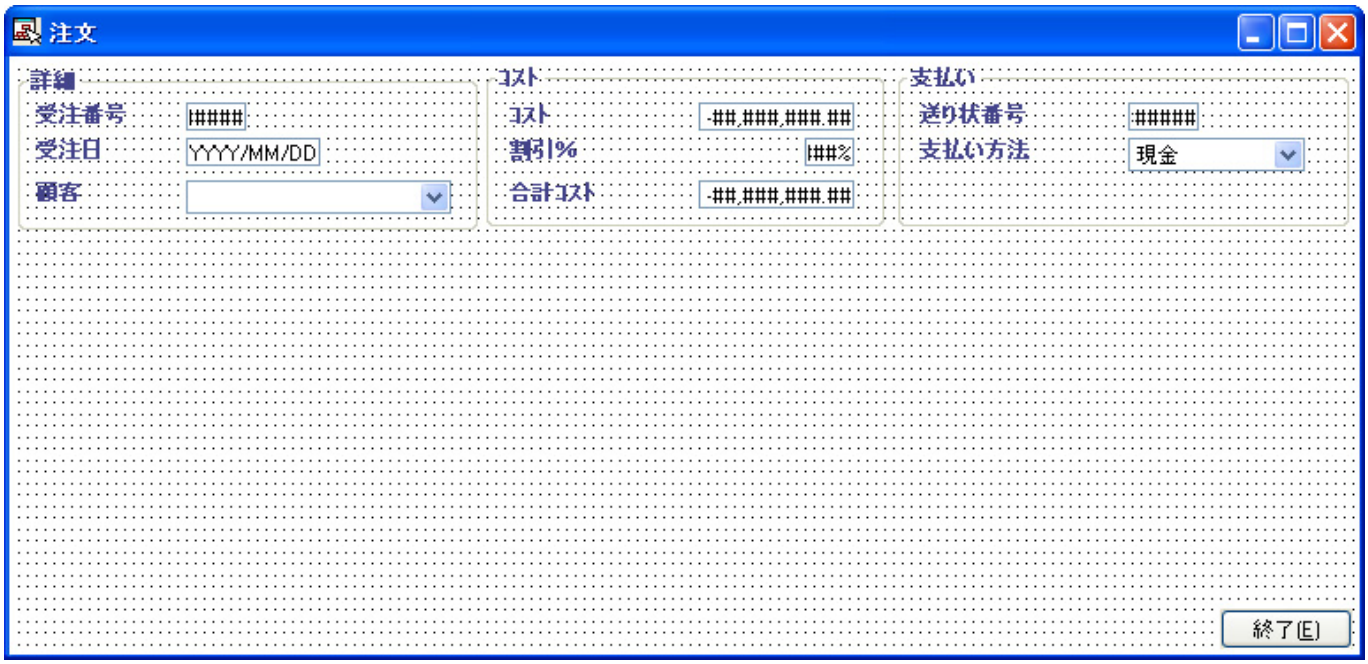


4.4. フォームエディタ

フォームエディタは、 **フォーム** タブをクリックするか、あるいは **Ctrl+3** キーを押すと開きます。



ここで表示される一覧から、**F5** でズームすると、画面が表示されます。下図は、注文プログラムの親タスクのフォームです。



V9 Plus では、フォーム一覧に **子ウィンドウ** 欄がありましたが、V10 では削除されています。子ウィンドウについての設定を行うには、GUI 表示形式フォームのフォーム特性中の **ウィンドウ タイプ** 特性により行います。

フォームエディタでの新機能については、GUI の強化（第 7 章）で詳しく説明します。

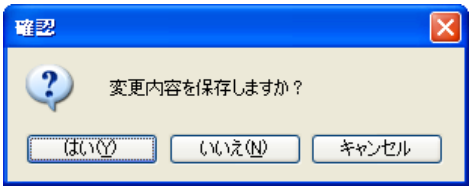
4.5. タスクエディタの保存と終了

4.5.1. タスクエディタの終了

タスクエディタを終了させるには・・・

タスクエディタは、データビュー、ロジック、フォーム エディタが開いている状態で、次のようにして終了します。

- Enter キーを押すと、修正内容を保存して終了します。
- ESC キー、あるいは Ctrl+F4 キー を押すか、またはウィンドウの ボタンを押すと、**保存確認**ダイアログが表示されます。
  - **はい(Y)** を押すと、変更内容を保存して、タスクエディタを終了します。
  - **いいえ (N)** を押すと、変更内容を破棄して、タスクエディタを終了します。
  - **キャンセル** を押すと、何もせずにタスクエディタに戻ります。



4.5.2. タスク修正内容の保存

V9Plus では、タスクの修正内容を保存するには、プログラムを閉じてタスクエディタをいったん終了する必要がありました。V10 では、タスクエディタを終了することなく、現時点での修正内容を保存することができます。

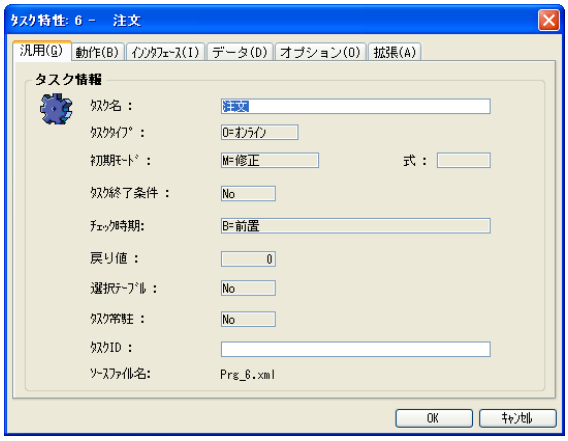
修正内容を保存するには・・・

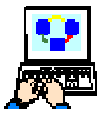
次のいずれかを行います。

- Ctrl+S キーを押します。
- **オプション** メニューから **プログラムの保存(V)**、 あるいは **オブジェクトを保存/閉じる(J)** を選びます。

4.6. タスク特性ダイアログボックス

タスク特性ダイアログは、タスク全体に関わるパラメータ類を設定します。





タスク特性ダイアログを開くには・・・

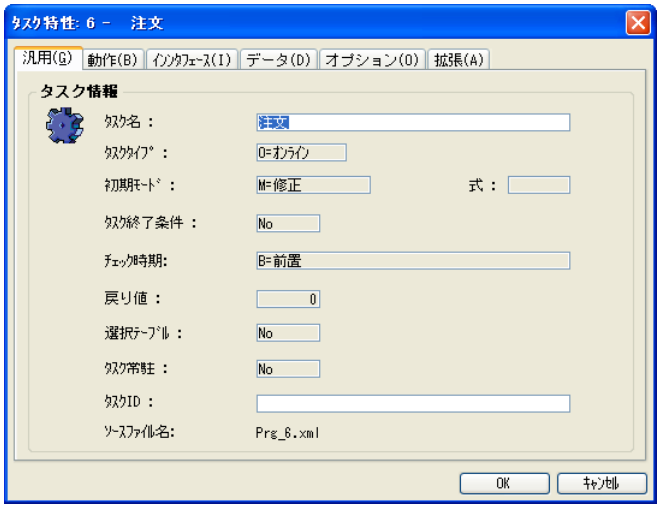
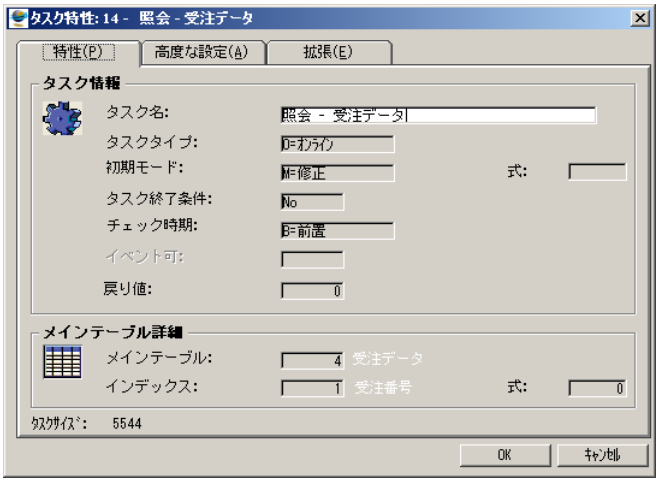
タスクを開いた状態で、次のいずれかにより行います。

- **Ctrl+P キー** を押します。
- **タスク(K) → タスク特性(T)** メニューを選びます。

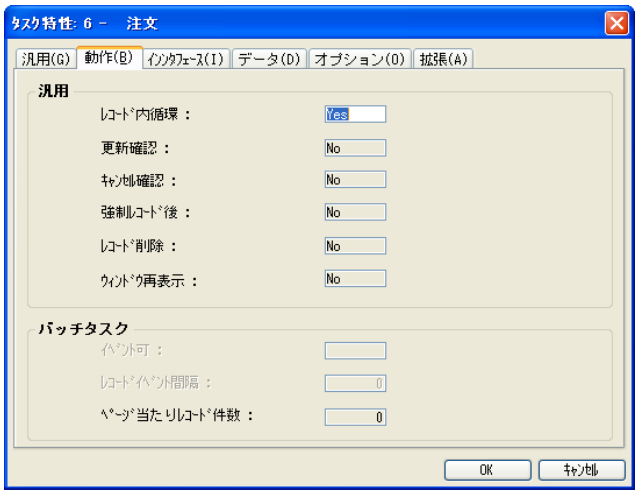
4.6.1. V9Plus のタスク特性/タスク制御と、V10 でのタスク特性の対応

V9Plus でのタスク特性およびタスク制御ダイアログでの各特性が、V10 ではどこで設定するのかをまとめたの下表です。

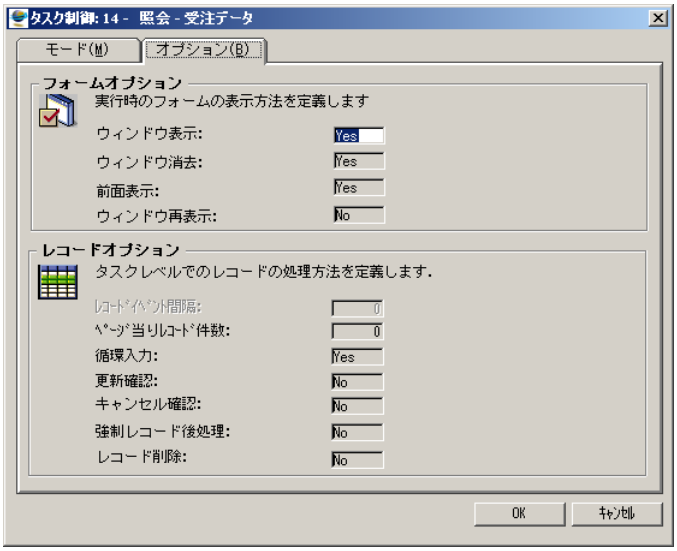
V9Plus では、**タスク特性、タスク制御** の二つのダイアログで設定していましたが、V10 では **タスク特性** ダイアログひとつにまとめ、タブで切替えて表示するようになりました。

V10	V9Plus
汎用タブ 	タスク特性 → 特性(P) タブ 

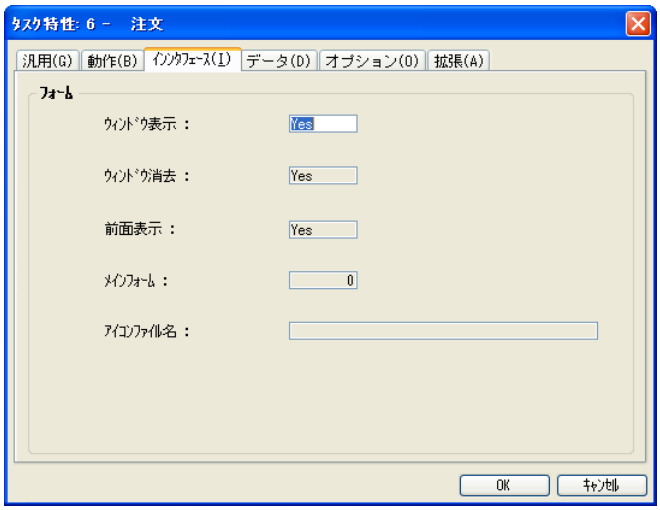
動作タブ



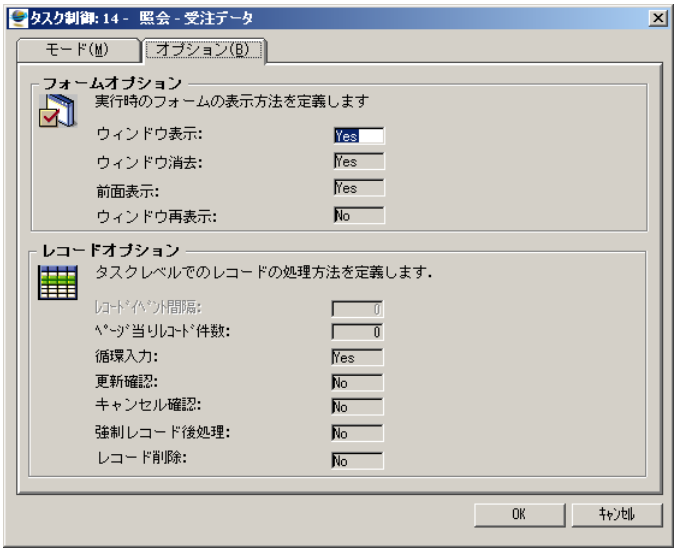
タスク制御 → オプションタブ



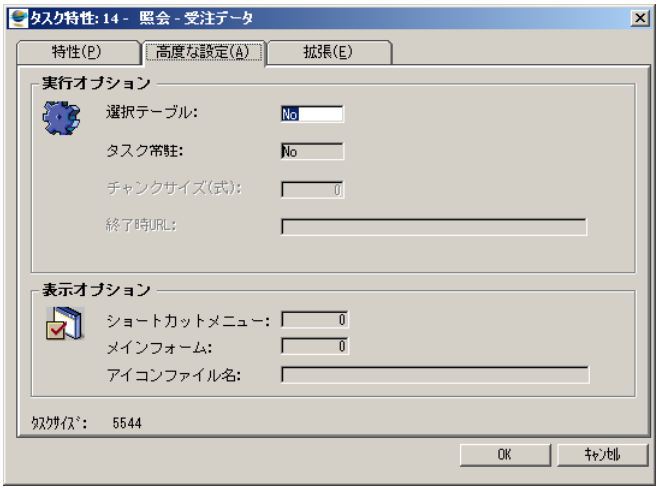
インターフェースタブ



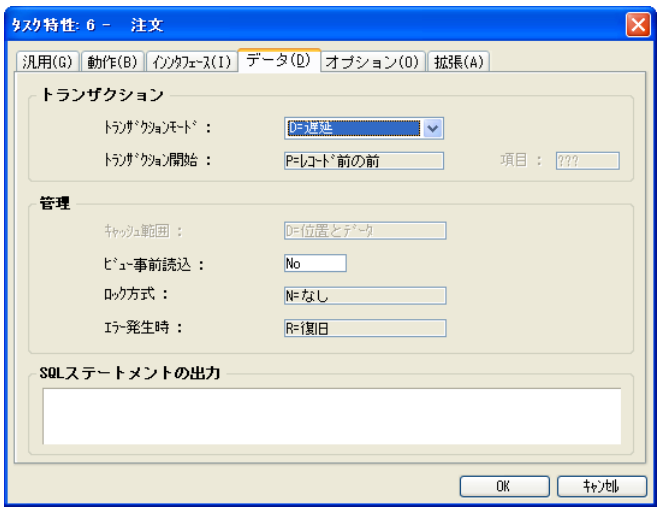
タスク制御 → オプション(B) フォームオプション



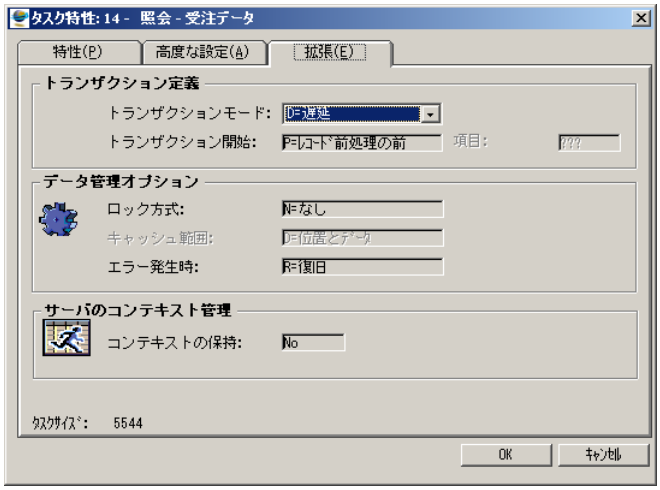
タスク特性 → 高度な設定(A) 表示オプション



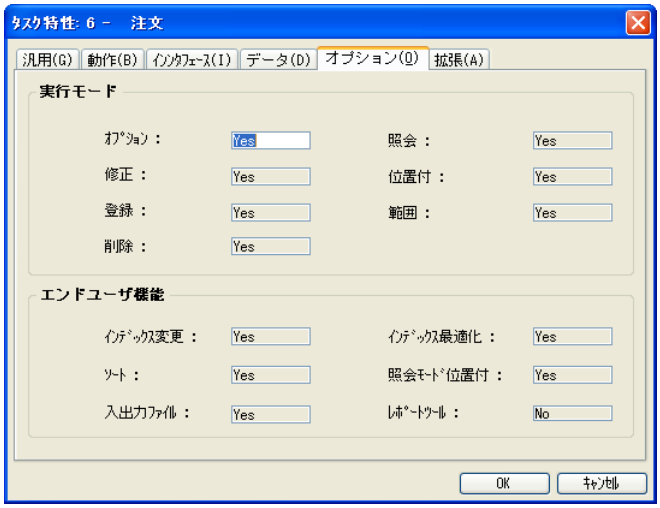
データタブ



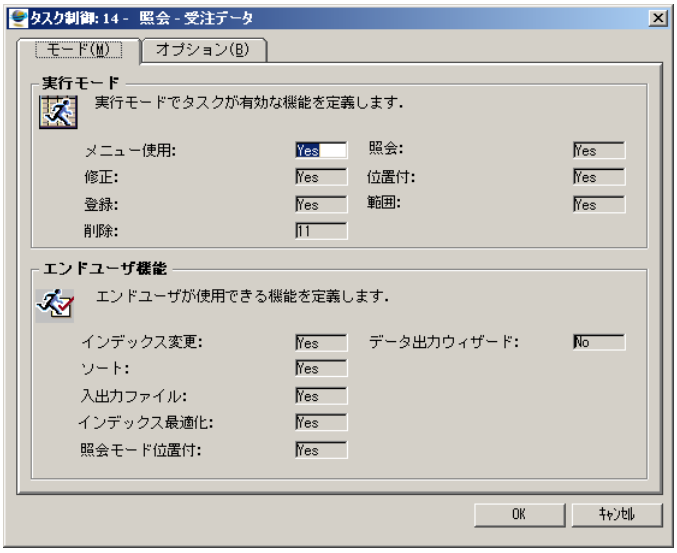
タスク特性 → 拡張(E) タブ



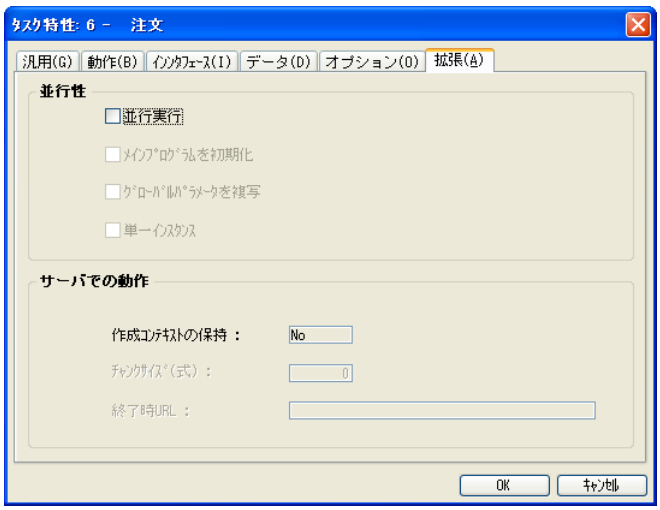
オプションタブ



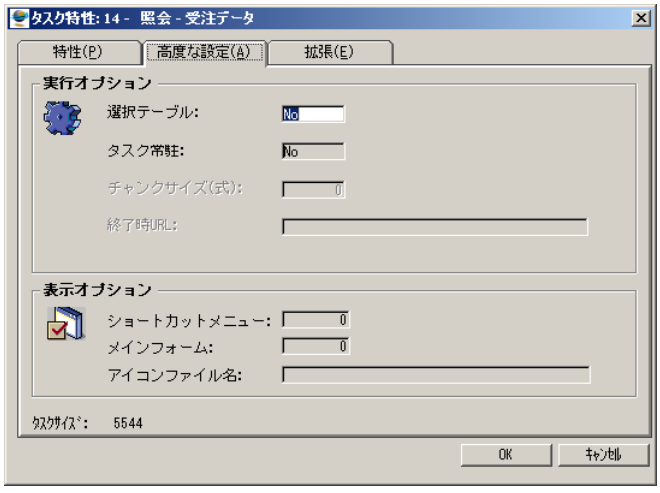
タスク制御 → モード(M) タブ



高度タブ



タスク特性 → 高度な設定(A)



V9Plus		V10
ダイアログ名	特性名	タスク特性ダイアログでのタブ名
タスク特性 → 特性(P)	タスク名 タスクタイプ 初期モード タスク終了条件 チェック時期 戻り値	汎用(G)
	イベント可	**動作(B)**
	メインテーブル インデックス インデックス式	（データビューに移動）
高度な設定(A)	選択テーブル タスク常駐	**一般(G)**
	チャンクサイズ 終了時 URL	拡張(A)
	ショートカットメニュー	
	メインフォーム アイコンファイル名	インターフェース(I)
拡張(E)	トランザクションモード、 トランザクション開始、 ロック方式、 キャッシュ範囲 エラー発生時	データ(D)
タスク制御 → モード(M)	メニュー使用、 照会、修正、登録、 位置付、範囲、削除、 インデックス変更 ソート、 入出力ファイル、 インデックス最適化、 照会モード位置付、	オプション(O)
	データ出力ウィザード	オプション(O) (レポートツール)
タスク制御 → オプション(B)	ウィンドウ表示 ウィンドウ消去 前面表示	インターフェース(I)

	ウィンドウ再表示 レコードイベント間隔 ページあたりレコード件数 循環入力 更新確認 キャンセル確認 強制レコード後処理 レコード削除	動作(B)

4.6.2. V10 で新たに追加されたタスク特性

V9Plus にはなく、V10 で新たに追加されたタスク特性には、次のようなものがあります。

	特性名	機能
汎用(G)	タスク ID	開発者がこのタスクの識別名を定義することができます。この特性は、トップレベルの親タスクのほかに、サブタスクにも設定することができます。V10 の新しい関数 TaskID により、実行時にタスク ID を知ることができます。
	ソースファイル名	(読込専用) このプログラムに対応する XML ソースファイル名を表示します。
データ(D)	ビュー事前読込	この特性が Yes の場合には、タスク開始時に、範囲指定を満たすレコードがすべて読み込まれ、該当するレコード数が求められ、スクロールバーはそのレコード数をもとにして表示されるようになります。スクロールバーで正確な表示を行いたい場合、スクロールスピードを上げたい場合などに用います。
	SQL ステートメントの出力	これは読込専用の特性で、埋め込み SQL タスクの場合の SQL 文を表示します。
拡張(A)	並行実行 メインプログラムを初期化 グローバルパラメータを複写 単一インスタンス	これらはタスクの並列実行を制御するパラメータです。並列実行については、複数のタスクを同時に実行する(第 14 章)で説明します。

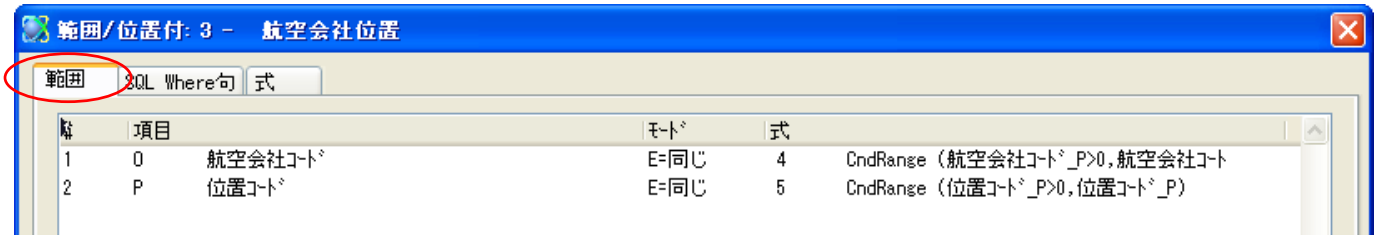
4.7. 範囲ダイアログボックス

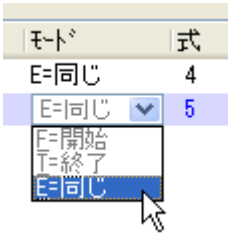
範囲/位置付 ダイアログボックスは、ビューに対する **範囲指定** および **位置付指定** などを設定・表示するものです。ここには、次のようなタブがあります

- 範囲
- SQL Where 句
- 式

4.7.1. 範囲タブ

範囲/位置付 タブは、V10 で新しく作られたもので、**データビュー** エディタで指定された範囲指定をまとめて表示します。



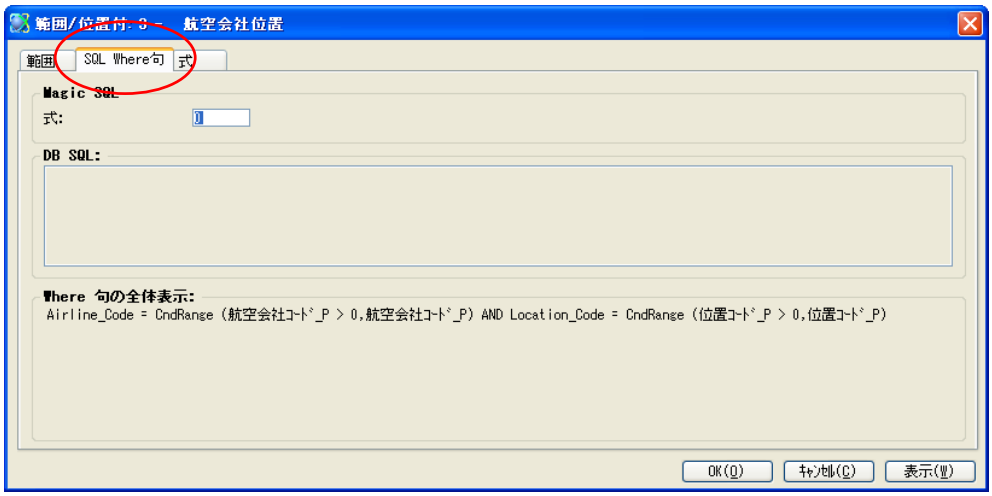


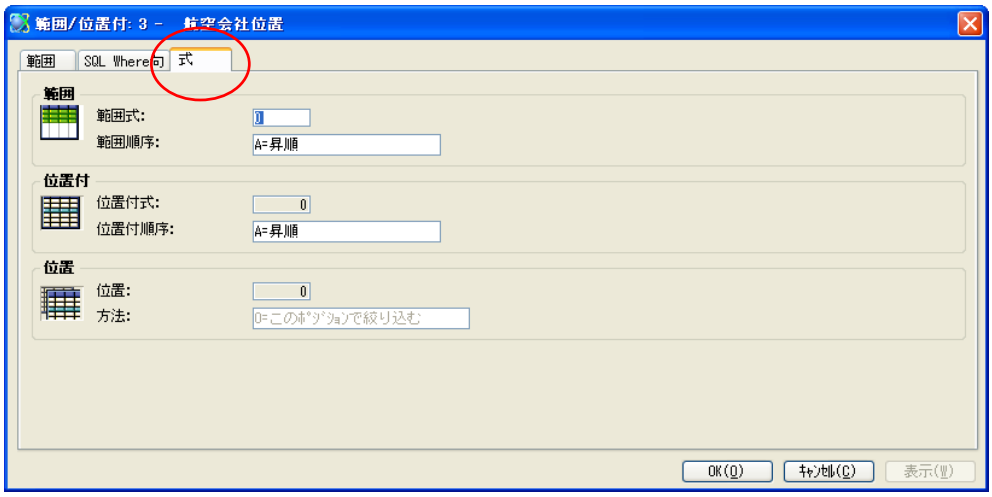
このテーブルでも、次の規則に従って範囲指定を設定することができます。

- **モード** 欄では範囲条件の評価方法として、**F=開始**、**T=終了**、あるいは **E=同じ** を指定します。
- 一つの項目に対して、範囲の **F=開始** と **T=終了** を、それぞれひとつづつ式で指定することができます。
- 開始と終了が同じ場合には、**E=同じ** として、1 行で指定することもできます。**E=同じ** と指定した場合には、同一の項目に対して **F=開始**、**T=終了** を指定することはできません。
- ここに設定された式は、タスクの開始時、タスク前処理よりも前に評価されます

4.7.2. SQL Where と 式 タブ

SQL Where および **式** タブで表示される画面は、V9Plus の時と変わりありませんので、ここでは説明を省略します。





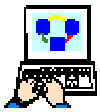
4.8. データビュー／ロジックエディタの共通機能

4.8.1. クイック式入力

Magic のプログラム作成時には、式を定義することが頻繁にあります。
V9Plus では、式を新規に定義する場合には、式番号を入力するカラムで、

1. **F5** でズームして式テーブルを開く。
2. **F4** で新規行を作成する。
3. 式を定義する。
4. OK (または Enter キー)で、式テーブルを閉じる。

という手順を踏む必要がありました。
V10 では、**クイック式入力** 機能があり、この一連の処理すばやく行うことができます。

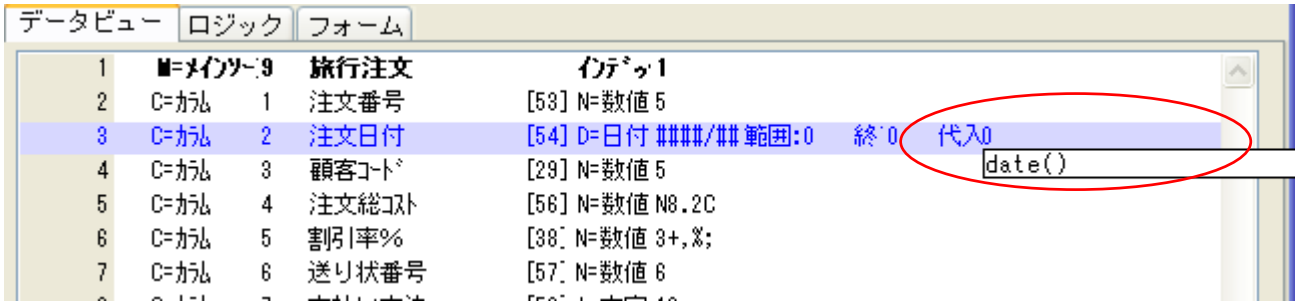


クイック式入力で式を入力するには・・・

1. イコール文字 （＝)をタイプします。クイック入力ウィンドウが開きます。(式テーブルを開く必要はありません)
2. 式を書きます。

これだけの操作で、式テーブルに式が追加され、式番号が欄に設定されます。

下図は、**注文日付** の **代入** 欄で、クイック式入力を使って **date()** と入力しているところです。



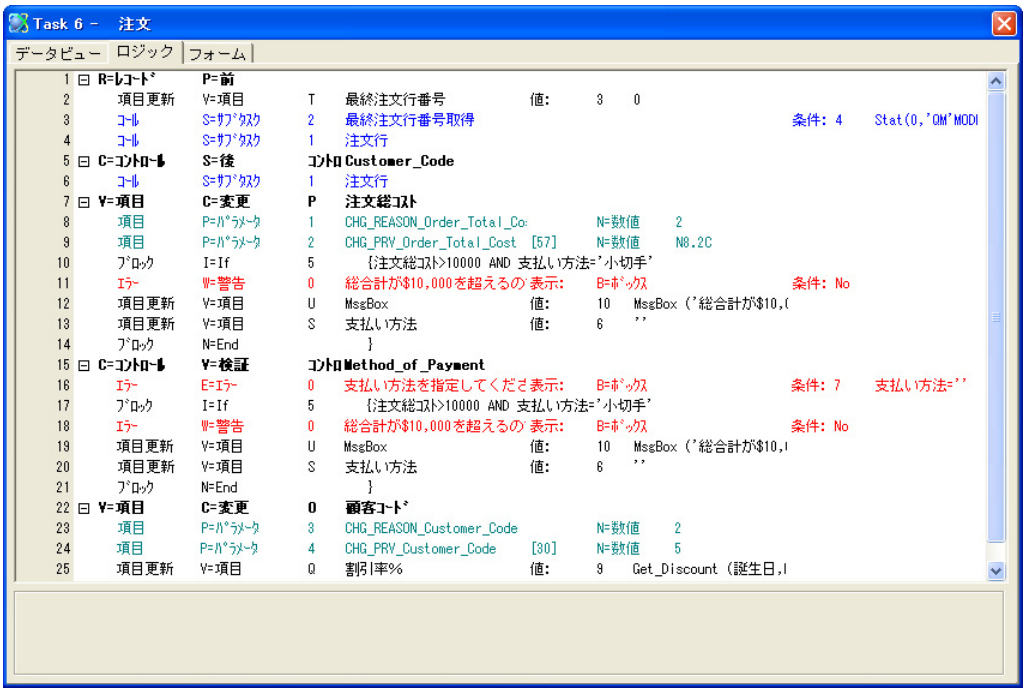
- 同じ式は再利用されます。すなわち、クイック式入力で入力した式と同一の式がすでに式テーブルに定義されていた場合には、新しく行は作成されず、既存の式番号が設定されます。
- 式番号の入力には、クイック式入力を使わず、V9Plus と同じ方法で(式欄からズームして)設定することもできます。式の再利用をしたくない場合、複雑な式を入力する場合には、このほうが便利でしょう。

4.8.2. 表示の展開と縮小

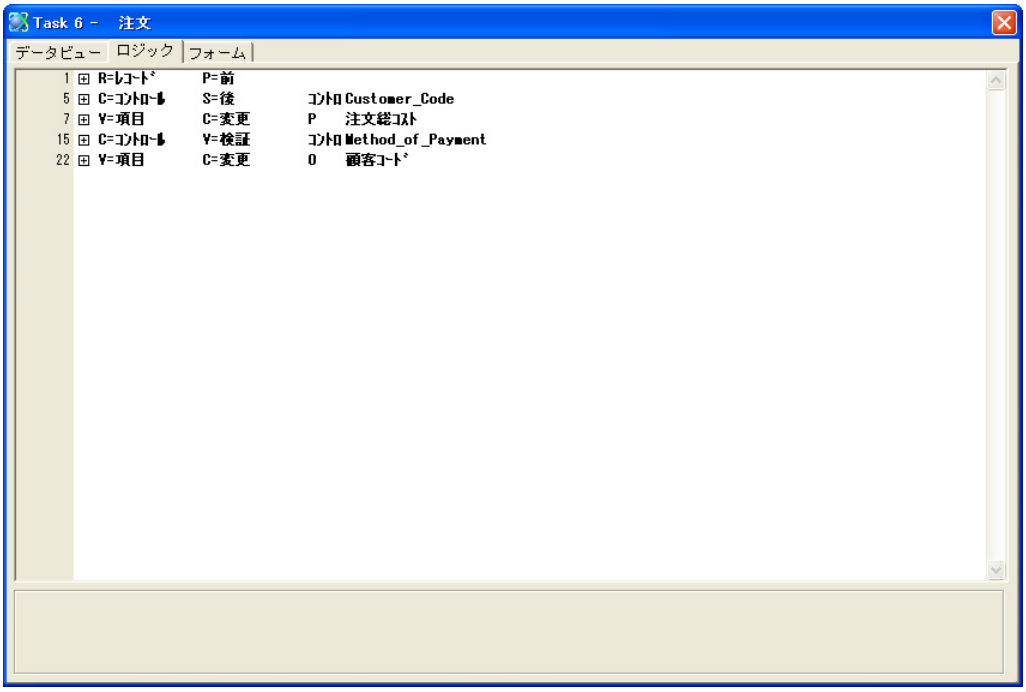
データビューおよびロジックエディタでは、詳細行の表示・非表示を設定することができます。詳細行が非表示の場合には、ヘッダ行のみが表示されます。
下図は、プログラム 6 番 **注文** プログラムで、ロジックエディタの画面をすべて展開した状態(上) と、すべて閉じた

状態(下)を表しています。

ロジックエディタですべて展開した状態



すべて閉じた状態

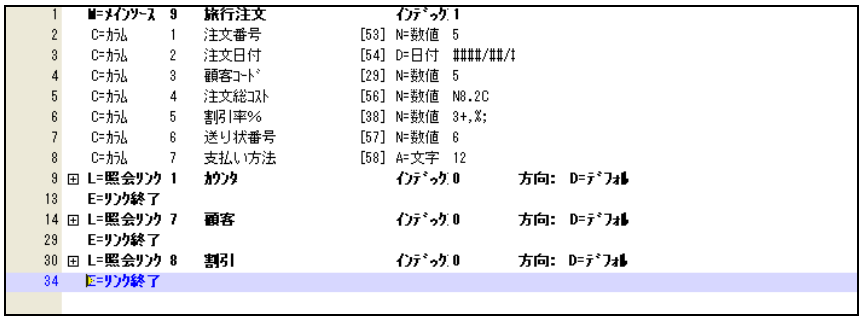


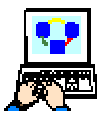
展開されているノードは ⊟ 記号で、縮小されているノードは ⊞ 記号で表されます。

データビューエディタの場合

データビュー エディタでは、リンクコマンドごとに、ノードを開いたり閉じたりすることができます。

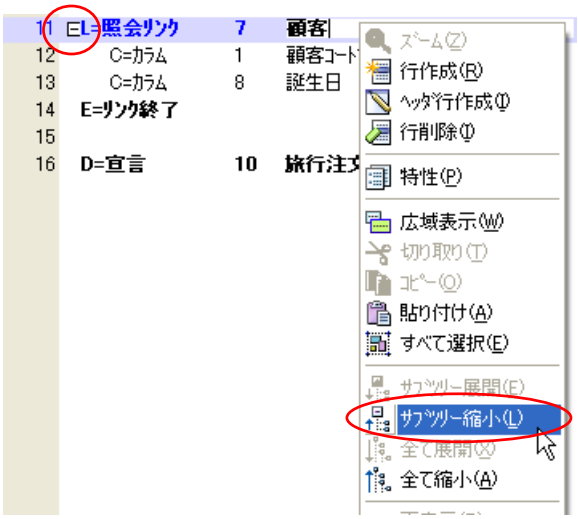
次の図は、データビューエディタで、リンクコマンドをすべて閉じた状態の画面です。



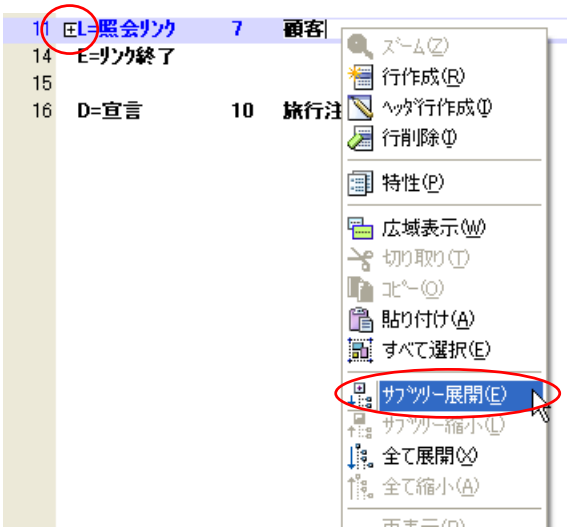


データビュー エディタで、開いているノードを閉じたり、開いたりするには・・・

1. デフォルトではすべてのノードが開いた状態です。
2. このとき ⊟ 記号をクリックするか、右マウスクリックでメニューを開いて **サブツリー縮小(L)** を選ぶと、このリンクコマンドのノードが閉じて、従属する **C=カラム** コマンドが非表示となります。このときには、⊞ 記号が表示されます。



3. ⊞ 記号をクリックするか、右マウスクリックでメニューを開いて **サブツリー展開(E)** を選ぶと、このリンクコマンドのノードが開いて、従属する **C=カラム** コマンドが表示されるようになります。



4. ⊟ 記号、⊞ 記号をクリックして、表示・非表示を切替えることができます。

一度に全てのノードを開いたり閉じたりすることもできます。

データビュー エディタで、一度に全てのノードを開いたり閉じたりするには・・・

1. 右マウスクリックでメニューを出し、 **全てを縮小(A)** を選ぶと、データビュー中のリンクコマンドに従属する **C=カラム** コマンドがすべて隠れます。
2. 逆に、メニューから **すべてを展開(X)** を選ぶと、閉じているノードがすべて開きます。

ロジックビューの場合

データビューの場合と同様、ヘッダ行を閉じたり開いたりして、詳細行を非表示・表示を切替えることができます。

ロジックビューで、個々のハンドラの閉じる・開くを切替えるには・・・

ヘッダ行の先頭にある ⊞、⊟ 記号をクリックして切替えます。

レコード前処理ハンドラを縮小した状態

```
⊞R=ﾚｺｰﾄﾞ      P=前
5 ⊞C=ｺﾝﾄﾛｰﾙ    S=後          ｺﾝﾄ Customer_Code
```

レコード前処理ハンドラを展開した状態

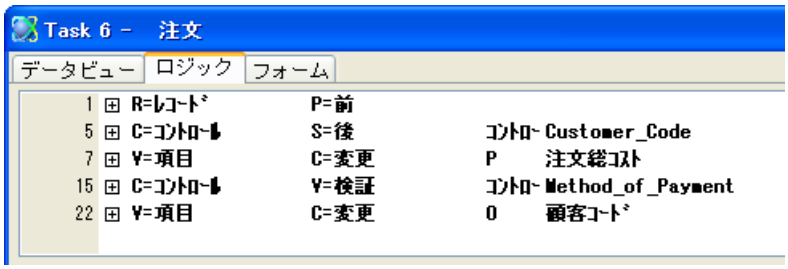
```
⊟R=ﾚｺｰﾄﾞ      P=前
2    項目更新   V=項目        T   最終注文行番号
3    ｺｰﾙ        S=ｻﾌﾞﾀｽｸ     2   最終注文行番号取得
4    ｺｰﾙ        S=ｻﾌﾞﾀｽｸ     1   注文行
5 ⊞C=ｺﾝﾄﾛｰﾙ    S=後          ｺﾝﾄ Customer_Code
```

ロジックビューで、すべてのハンドラを一度に閉じたり開いたりするには・・・

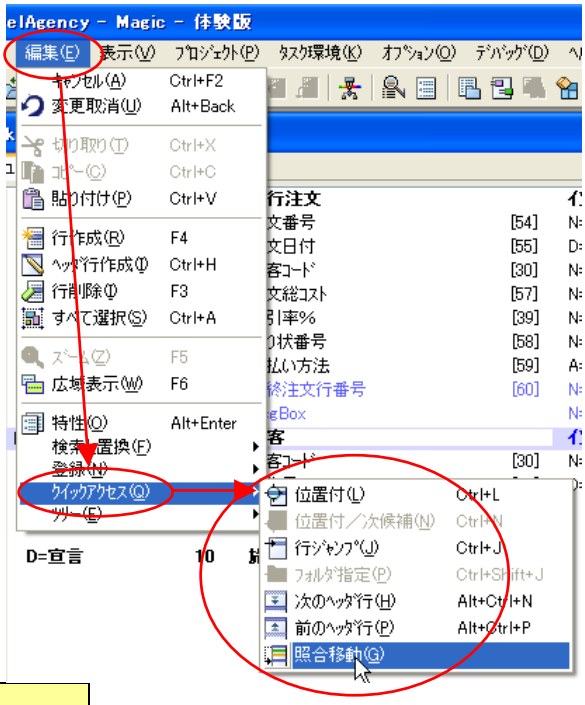
データビューハンドラの場合と同じく操作で行えます。

1. 右クリックでメニューを出し、**全てを縮小(A)** を選ぶと、すべてのハンドラが一度に閉じられます。
2. 逆に、メニューで **全てを展開(X)** を選ぶと、閉じられているハンドラがすべて開かれます。

下図は、ロジックビューのハンドラをすべて縮小した状態の画面です。



4.8.3. クイックアクセスメニュー



ヘッダ行を単位として、カーソルを移動することができます。

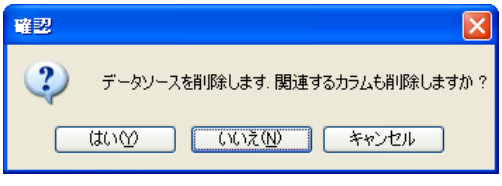
編集(E) メニューの **クイックアクセス** サブメニューには、以下のようなものがあります。

メニュー	ショートカット	機能
次のヘッダ行	Alt+Ctrl+N	次のヘッダ行にカーソルを移動させる。
前のヘッダ行	Alt+Ctrl+P	前のヘッダ行にカーソルを移動させる。
照合移動		現在カーソルのあるヘッダ行に対応するヘッダ行に移動する。(ヘッダ行にカーソルがあるときに有効となる。)
行ジャンプ	Ctrl+J	指定された行に移動する。

ここで、**照合移動** メニューで言う 対応するヘッダ行、とは、次のような対応を言います。

- **リンク** と **リンク終了**
- **タスク前処理** と **タスク後処理**
- **レコード前処理** と **レコード後処理**
- **グループ前処理** と **グループ後処理**
- **コントロール前処理**、**コントロール検証**、および **コントロール後処理**

4.8.4. ヘッダ行の削除



ロジックエディタでは、ヘッダ行(ハンドラレベルを定義する行)を削除すると、そのハンドラ内のコマンド行も同時に削除されます。

データビュー エディタでは、リンク コマンドの行を削除しようとすると、関連するカラムも同時に削除するかを確認するダイアログが開きます。

- **はい(Y)** で答えると、リンクコマンドと、それから選択されていたカラムとが同時に削除されます。
- **いいえ(N)** で答えると、リンクコマンドの行だけが削除され、カラム行はメインソースに属するものとなります。
- **キャンセル** で答えると、何も削除されません。

4.8.5. マルチマーキング

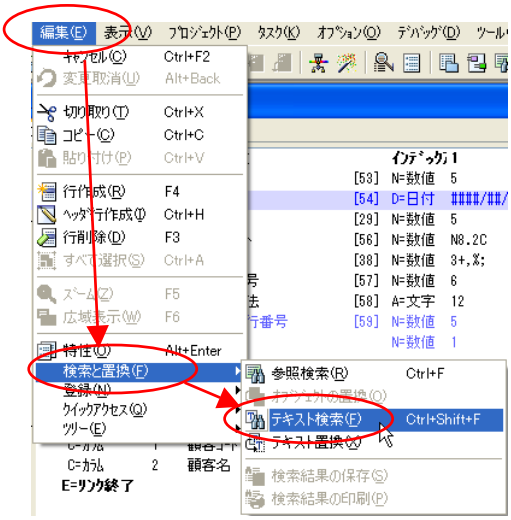
マルチマーキングは、V9Plus と同様に V10 のデータビュー、ロジックエディタでも有効です。マルチマーキングした行は、コピー、切り取り、貼り付けなどを行うことができます。

ロジックエディタの場合には、ハンドラレベルを定義するヘッダ行をマークすると、ハンドラ全体をマークしたことになります。従って、切り取りして別のところに貼り付けすると、ハンドラ全体が移動させられます。
ヘッダ行(ハンドラレベル)と詳細行(コマンド)を混合してマルチマークすることはできません。すなわち、詳細行をマークしている状態ではヘッダ行をマークに含めることができず、その逆も同じです。

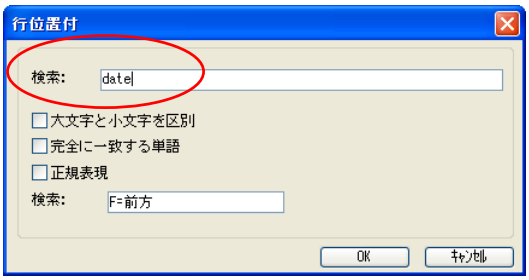
4.8.6. テキストの検索

データビュー、ロジック エディタ上で、テキスト検索を行うことができます。
V10 のデータビュー、ロジック エディタ上でのテキスト検索は、表示されている文字列に対して行われます。例えば、コマンド名、カラムや変数名、書式、更には、設定されている式の展開内容までが検索対象になります。このため、V9Plus のときよりも検索の自由度が高くなりました。

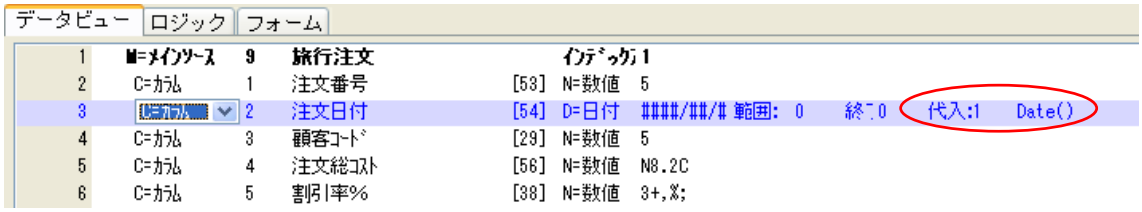
例: 注文 タスクのデータビューで、Date() 関数が使われている行を検索するには・・・



1. **編集(E) → 検索と置換(F) → テキスト検索(F)** メニューで **行位置付** ダイアログを開きます。
2. **検索文字列**として date を指定します。



3. OK ボタンを押すと、3 行目の **注文日付** カラムの代入式にあるのが見つかります。



4.9. 式エディタの強化

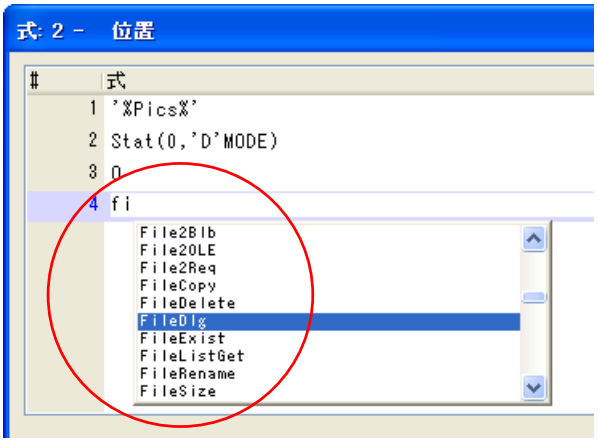
V10 の式エディタは、式の入力や表示を容易にするために、数々の改善がされています。

4.9.1. 関数の自動コンプリート

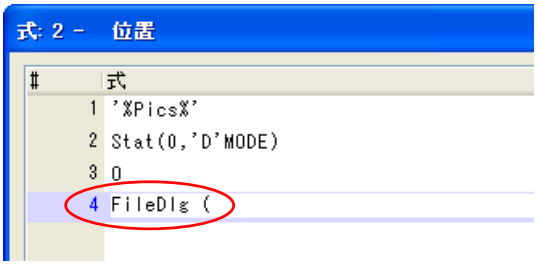
関数の自動コンプリート機能は、式エディタで関数名を入力しているときに、**Ctrl+スペース** キーを押すと、その時点までで入力された文字列から始まる関数名を一覧で表示する機能です。

例: FileDlg 関数を自動コンプリートを使って入力するには・・・

1. 式エディタを開き、**F4** で新規行を追加します。
2. fi と入力した状態で、**Ctrl+スペース** キーを押します。すると、関数名の一覧が表示され、fi で始まるものに位置付られます。



3. この中から、FileDlg を選びます。すると、「**FileDlg (」**が式エディタに表示されます。



もし、該当する関数が一つしかない場合には、一覧は表示されずに即座に関数名が入力されます。例えば、上の例では、filed まで入力してから **Ctrl+スペース** を押すと、すぐに「**`FileDlg (`**」が表示されます。

4.9.2. 式の展開

V9Plus と同じく、式エディタで、**F6** キーを押すと、広域表示がされます。長い式を入力する場合には、自動的に広域表示になります。

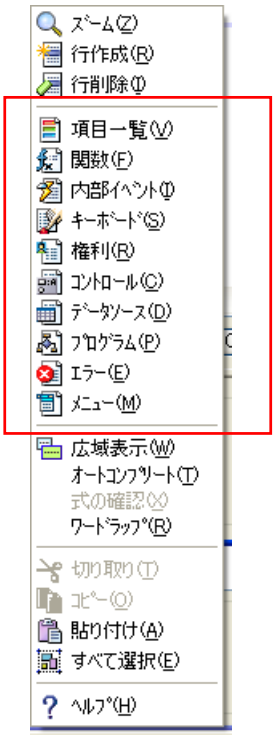
V10 では、広域表示画面になっている状態で、改行やタブを使うことができるようになりました。

- 改行（**Enter** キー)を入力すると、式が改行されて表示されます。
- タブキーを入力すると、式はインデントされて表示されます。

4.9.3. 選択リスト

式エディタで右マウスクリックでメニューを表示させると、各種の一覧が表示され、選択して式に入力することができるようになります。

メニュー	ショートカット	説明
項目一覧	F5 または Ctrl+0	現在のタスクで有効な項目が一覧で表示されます。
関数	Ctrl+1	関数一覧ダイアログを表示します。
内部イベント	Ctrl+2	システム既定義の内部イベントの一覧を表示します。
キーボード	Ctrl+3	キーとその組み合わせです
権利	Ctrl+4	権利リポジトリに登録されている権利の表示します。
コントロール	Ctrl+5	現在のタスクの GUI0 フォームで定義されているコントロールの一覧を表示します。
データソース	Ctrl+6	利用可能なデータソースの一覧を表示します。
プログラム	Ctrl+7	呼び出し可能なプログラムの一覧を表示します。
エラー	Ctrl+8	利用可能なエラーの一覧を表示します。
メニュー	Ctrl+9	トップレベルのメニューの一覧を表示します。



V9 Plus

- **内部イベント**は、V9Plus では **アクション** と呼んでいたものです。また、V9Plus では、リテラルとして **ACT** を使っていましたが、V10 では **EVENT** を使います。
- **データソース**は、V9Plus では、**テーブル**と呼んで、リテラルとしては **FILE** を使っていました。V10 では、より一般化した**データソース**という名前になり、リテラルとして **DSOURCE** を使います。

4.9.4. コンテキスト依存ヘルプ

V10 の式エディタでは、関数のコンテキスト依存ヘルプの機能があります。
これは、式エディタ上で、関数名の上にカーソルがある状態で **F1** キーを押すと、その関数に関するリファレンス ヘルプ ページを表示するものです。

4.9.5. V9 よりの関数の変更

V9 Plus **サポートされなくなった関数**: 以下の関数は、V10 ではサポートされなくなりました。

DSTR3	ISTR	EVAL	TSTR3	EuroCnv
DVAL3	IVAL	RVAL	TVAL3	EuroDel
ESTR	LSTR	RSTR	VAL3	EuroGet
FLOW3	LVAL	STR3	Pref	EuroSet
				EuroUdp

V9 Plus **関数名の変更**: 以下の関数は、V10 で名前が変わりました。

V9Plus での名	V10 での名前
IOCopy	FileCopy
IODel	FileDelete
IOExist	FileExist
IORen	FileRename
IOSize	FileSize
CtrlName	LastClicked
ASC	ASCIIVal
CHR	ASCIIChr
MaxMagic	WinMaximize
MinMagic	WinMinimize
ResMagic	WinRestore
MTblSet1	MTblSet

名前の変更された関数は、V9Plus コンバータ(V9Plus よりのマイグレーションユーティリティ)が自動的に改名します。V9Plus コンバータについては、第 8 章 以前のバージョンから V10 への移行 で説明します。

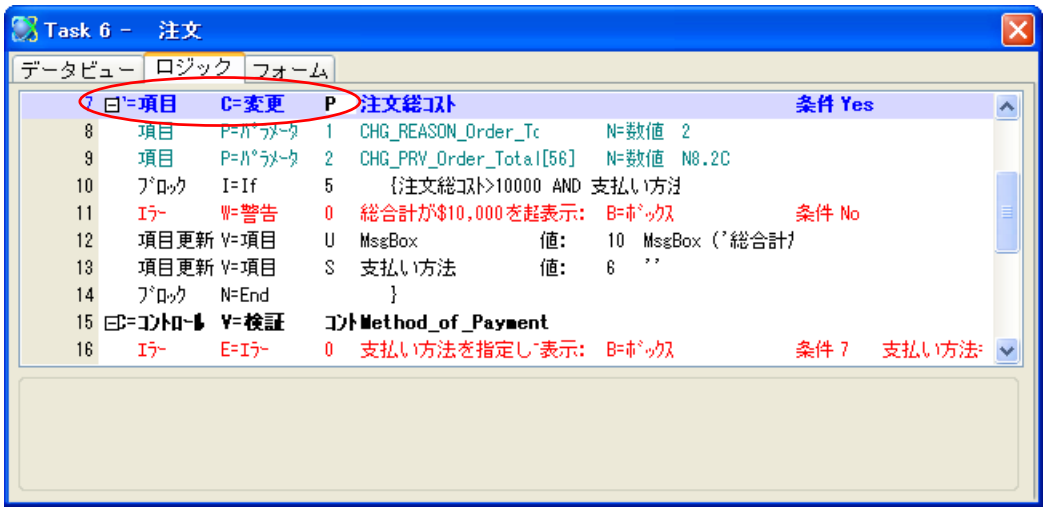
5. 項目変更ロジックユニット

V9Plus では、オンラインプログラムで、あるコントロールの値がユーザにより変更された場合にトリガされる、**コントロール 変更** というハンドラを定義することができましたが、以下のような場合には、項目の値が変更されたときにハンドルする方法がありませんでした。

- バッチタスクの場合
- 項目がオンラインフォームに配置されていなかった場合。
- 項目更新コマンドなど、ユーザの入力以外の方法で、項目の値が変更された場合

V10 では、このような場合でも、項目の値に変更が加えられた場合にトリガされる、**項目 変更** というハンドラが新たに定義されました。これに伴い、V9Plus の **コントロール 変更** ハンドラは廃止になりました。

サンプルでは、プログラム 6 番 **注文** で、項目変更ハンドラが定義されています。ここでは、**注文総コスト**（項目 P）の値が変わったとき、10,000 ドル以上の場合に支払い方法が小切手になっていないかどうかを確認しています。



本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

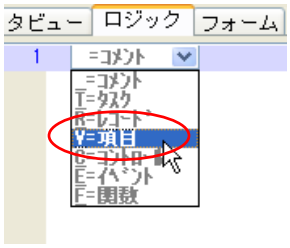
項目変更ロジックユニットの定義	ロジックエディタ → ロジックユニット → 項目変更特性
項目変更ロジックユニットの実行時動作	**Magic エンジン → エンジン実行レベル → 項目レベル**

5.1. 項目変更ハンドラの作成

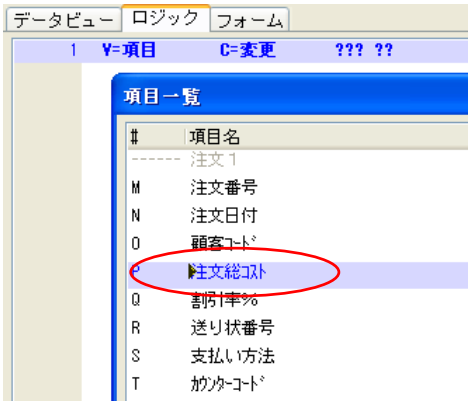
項目変更ハンドラは、次のように作成します。

項目変更ハンドラを作成するには・・・

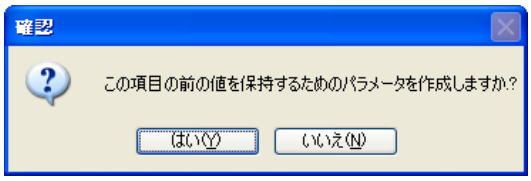
1. ロジックエディタを開きます。
2. **Ctrl+H** により、ヘッダ行を作成します。
3. ロジックユニットのタイプとして、**V=項目** を選びます。



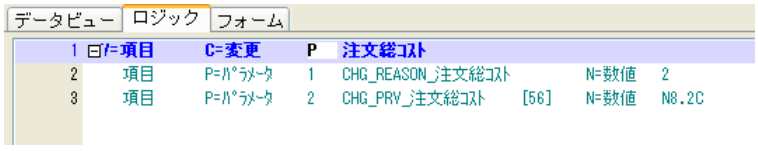
4. 項目欄からズームして、項目変更の監視の対象となる変数を選択します。



5. 項目変更ハンドラへのパラメータ を作成するかどうかを聞いてきます。



6. **はい(Y)** を選択すると、パラメータが二つ自動的に設定されます。
   いいえ(N) を選ぶと、パラメータは作成されません。



このパラメータには、実行時に次の値が自動的に設定されます。

#	パラメータ名	データ型	意味
1	CHG_REASON_(項目名)	数値	数値型で、以下の値が渡される。 0 – ユーザ入力による変更 1 – ユーザ入力以外の原因による変更。
2	CHG_PRV_(項目名)	(項目名)と同じ型と書式	変更される前の値。

5.2. 実行時の動作

項目変更ハンドラは、次の順序で実行されます。

1. 項目の値の更新が実行される。
2. 必要な再計算がすべて実行される。
3. 項目変更ハンドラの内容が実行される。

従って、項目変更ハンドラの実行時には、データビューの項目は、再計算の対象となるものも含め、すべて更新されたあとの値となっています。

その他、項目変更ハンドラに関する一般規則は以下の通りです。

- 複数の項目変更ハンドラが実行対象となる場合には、データビューエディタ上で上に現れるものが先に順番に実行されます。
- 同じ項目に対して複数の項目変更ハンドラが定義されている場合には、ロジックエディタ上で一番下にあるものだけが実行されます。
- 取り消し やトランザクションのロールバックの際には、項目変更ハンドラは実行されません。
- 項目変更ハンドラに指定されている項目が、オンラインフォーム上でチョイス コントロール（コンボボックスやリストボックスなど)に配置されている場合、チョイス コントロール上で選択値を変えただけでも、項目変更ハンドラが実行されます。

Level 関数（タスクの実行レベルを文字列で返します）が返す値が変わりました。

- V9Plus で **コントロール 変更** ハンドラ実行中に Level 関数を実行した場合、**CC_(変数名)** という値が返ってきました。
- V10 では、**項目変更**ハンドラ実行中に Level 関数を実行すると、**VC_(変数名)** という値が返ってきます。

6. ユーザ定義関数

Magic には、200 を超える組み込み関数が定義されており、プログラム中で利用することができます。
V10 では、標準の組み込み関数に加えて、ユーザがアプリケーション独自の関数を定義し、プログラム中で利用することができるようになりました。

<table>
<tr><td colspan="2"> 本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。</td></tr>
<tr><td>ユーザ定義関数の定義と動作</td><td>Magic エンジン → ユーザ定義関数</td></tr>
</table>

6.1. ユーザ定義関数の定義

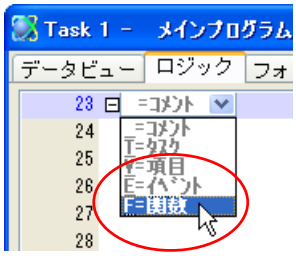
ユーザ定義関数は、ロジック エディタで以下のように定義します。

ユーザ定義関数を定義するには・・・

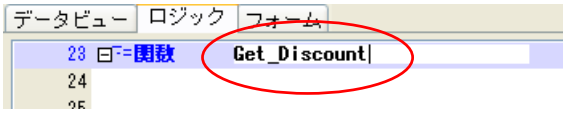
1．ロジックエディタを開きます。

2．**Ctrl+H** で、ヘッダ行を作成します。

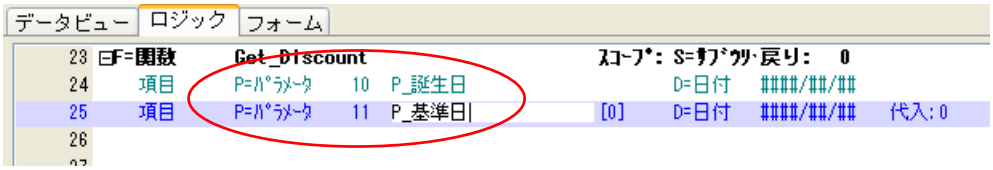
3．ロジックユニットのタイプとして、**F=関数** を選びます。



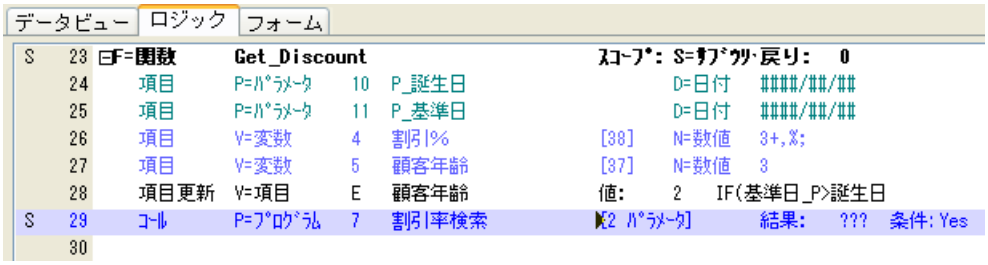
4．関数名を指定します。



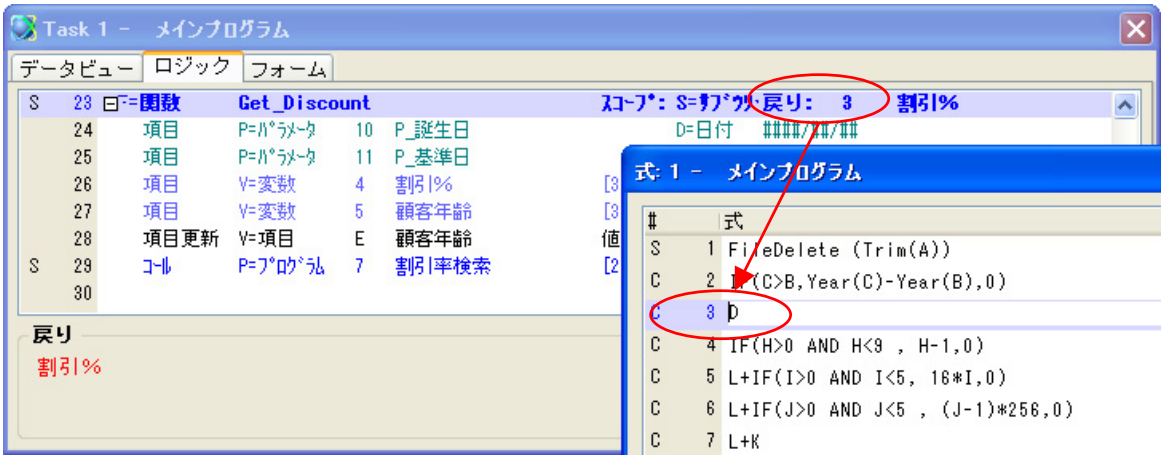
5．パラメータを必要なだけ指定します。(**V=項目 P=パラメータ** で定義します)



6．必要なロジックを記述します。

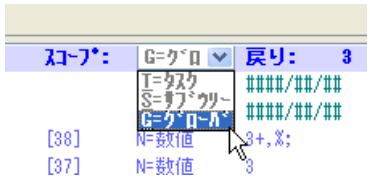


7．最後に、戻り値を式で設定します。



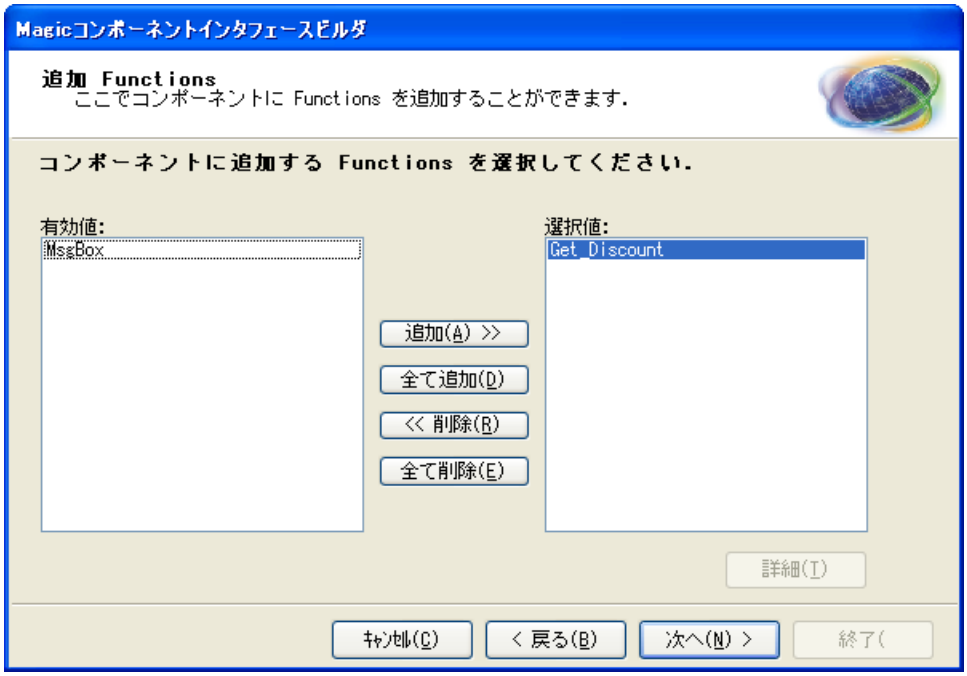
6.2. ユーザ定義関数のスコープ

ユーザ定義関数のスコープ（有効範囲)は、イベントハンドラなどと同じで、関数のヘッダ行にある **スコープ** パラメータにより決まります。



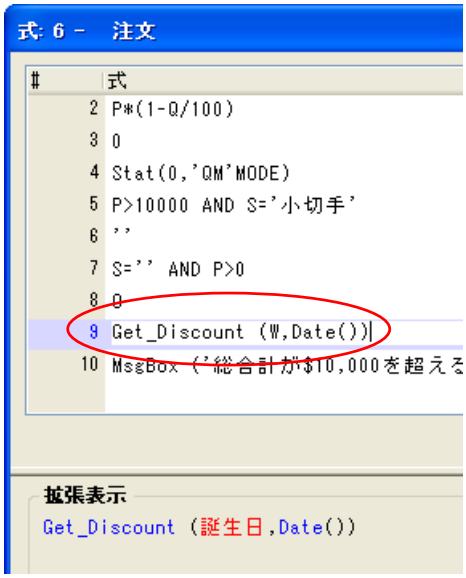
スコープ パラメータ	説明
T=タスク	関数が定義されているタスクでだけ有効です。
S=サブツリー	関数が定義されているタスクと、それに従属するタスク(サブタスク、孫タスク、・・・)で有効です。
G=グローバル	関数はプロジェクト全体で有効です。メインプログラムで関数を定義した場合にだけ選択することができます。

このプロジェクトをコンポーネントとする場合、グローバル関数は、**コンポーネント インターフェース ビルダ**で選択することができます。選択すると、コンポーネントを利用するプロジェクトからも、この関数を利用することができるようになります。



6.3. ユーザ定義関数の参照

ユーザ定義関数は、以下に説明するように、標準の組み込み関数と同等なものとして利用することができます。

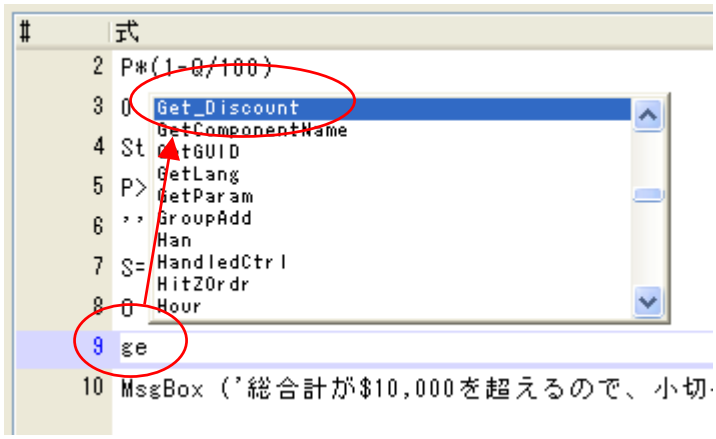


6.3.1. 式エディタ

ユーザ定義関数は、有効範囲内のプログラムの式エディタで参照することができます。

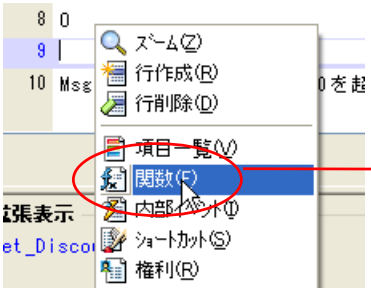
6.3.2. 自動コンプリート

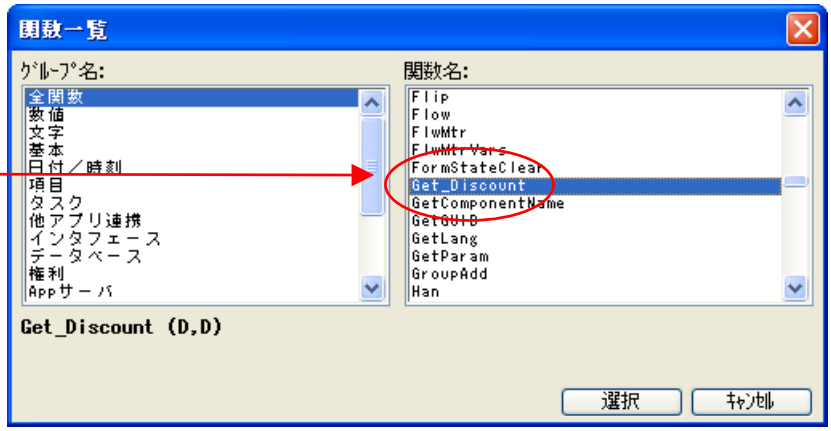
ユーザ定義関数は、自動コンプリートの対象となり、関数名の一部だけ入力した状態で **Ctrl+スペース** キーを入力すると、関数一覧の中に表示されます。



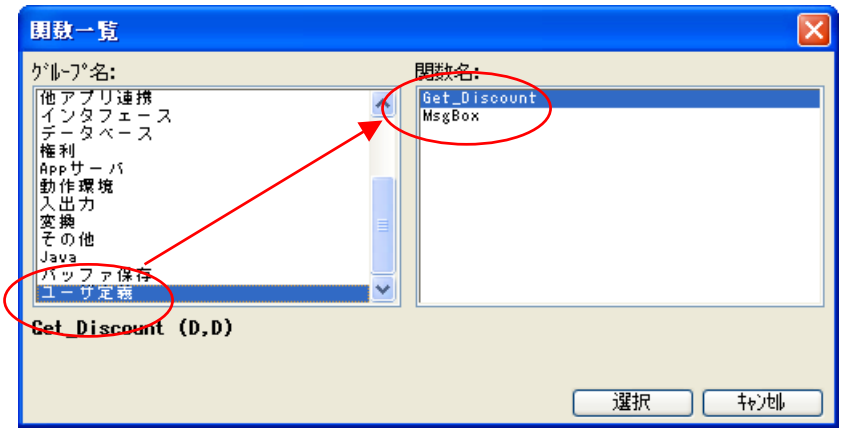
6.3.3. 関数一覧

式エディタで右マウスクリックによるメニューから 関数(F) を選択するか、あるいは Ctrl+1 キーを押すと、関数一覧が表示されます。この中に、他の関数に混じって、ユーザ定義関数が表示されます。パラメータのタイプも表示されます。





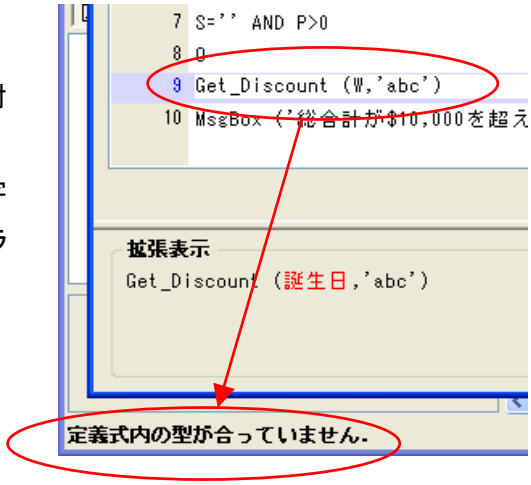
ユーザ定義関数は、**全関数** グループ、あるいは **ユーザ定義** グループに表示されます。



6.3.4. シンタックスチェック

ユーザ定義関数のパラメータおよび戻り値は、シンタックスチェックの対象となります。

例えば、Get_Discount の第2パラメータは日付型ですが、これに文字型のパラメータを与えると、「定義式内の型が合っていません」というエラーが表示されます。



6.4. その他の注意事項

6.4.1. 既定義の関数と同じ関数名を使う場合

ユーザ定義関数の名前として、標準で定義されている組み込み関数と同じ名前を使うことは可能です。この場合にはエラーにはならず、ユーザ定義関数の方が実行されます。
また、同一の名前を持つユーザ定義関数を複数定義することも可能です。この場合には、ロジックエディタ上で、後(下)の方に定義されている方が実行されます。

いずれの場合にも、関数一覧には名前は一つだけしか現れません。

6.4.2. ユーザ定義関数を削除する場合

ユーザ定義関数を削除することは可能ですが、削除すると、その関数を使っている式はすべてシンタックスエラーとなります。
ユーザ定義関数の名前を変更することもできますが、その関数を参照している式は自動的に更新されません。従って、この場合も式はシンタックスエラーとなります。
ユーザ定義関数を削除あるいは改名する場合には、予め **編集(E)** → **検索と置換(F)** → **テキスト検索(F)** メニューなどにより、そのユーザ定義関数を使っている式がないかどうか確認してください。

6.4.3. パラメータと戻り値

パラメータは、値渡しで渡されます。従って、関数内でパラメータの値を更新した場合にも、そのもとの変数が更新されることはありません。
戻り値は、関数のロジックがすべて実行された後に、**戻り** に設定されている式を評価して計算します。
戻り に式番号が設定しないでおく(0 のままにしておく)ことも可能です。この場合には、関数の戻り値として Null 値が返ります。

6.4.4. レベルとタスク世代

ユーザ定義関数が定義されているタスクがどこであっても、Level 関数の結果、および タスクの世代 をパラメータとして使う関数 (Stat 関数など)の評価には影響を与えません。常にその関数を参照する式が定義されているタスク、あるいはロジック ユニットを基準として計算されます。

7. GUI の強化

V10 では、オンラインフォームの GUI が強化され、最新の Windows コントロールの Look＆Feel が利用できるようになりました。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

GUI の詳細	**表示フォーム → GUI 表示フォーム** **表示フォーム → GUI コントロール**

7.1. TAB 順序

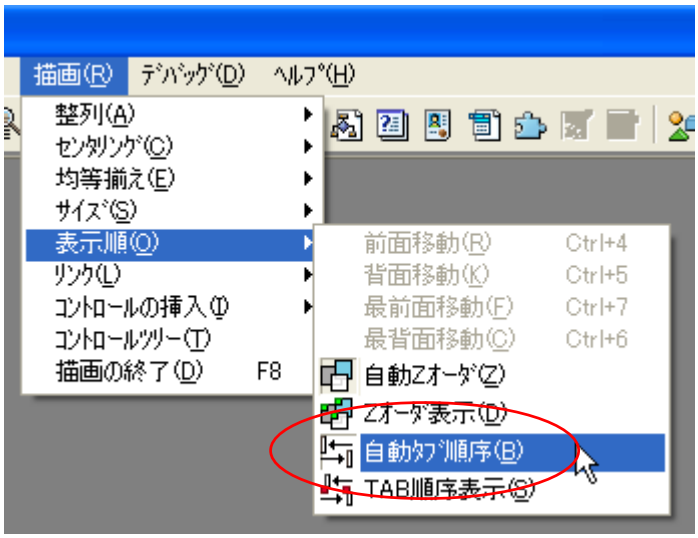
V9Plus までは、オンラインフォーム上のカーソルの TAB 順序(移動順序)は、レコードメインでのセレクトコマンドの順番に従っていました。

V10 では、データビューエディタでの項目の定義順序と、オンラインフォーム上での TAB 順序とは関係がなくなり、次のいずれかの方法により TAB 順序が決められます。

- 手動 TAB 順序
- 自動 TAB 順序

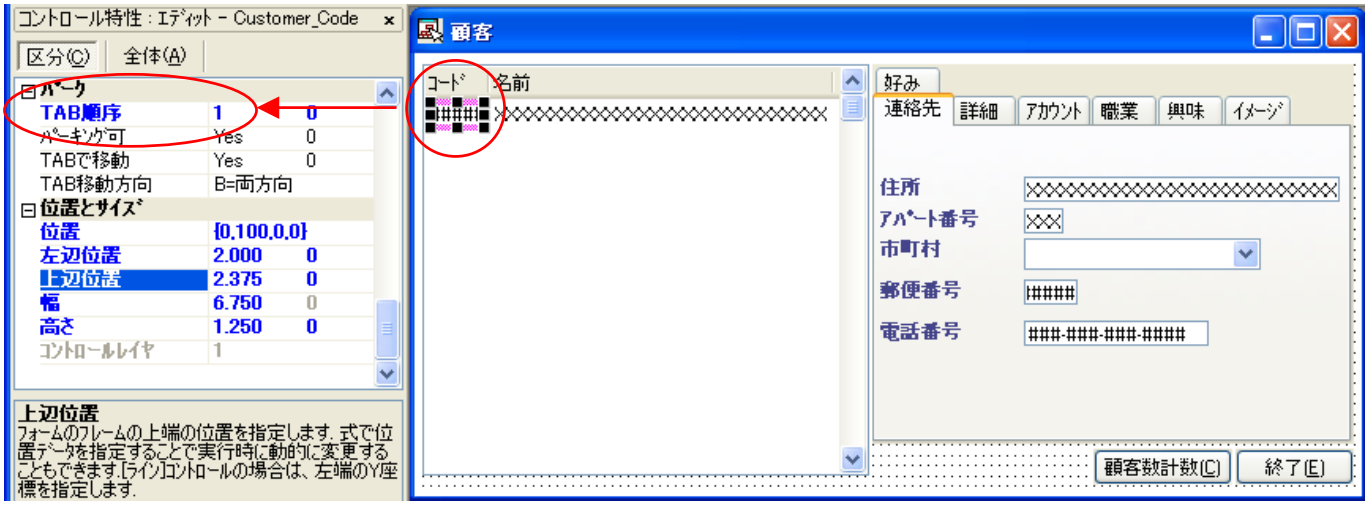
7.1.1. 自動／手動 TAB 順序の設定

自動タブ/手動タブは、フォームエディタを開いて、**描画(R)** → **表示順(O)** → **自動 TAB 順序(B)** メニュー、あるいは **コマンド** パレットの **自動 TAB 順序** ボタンによりトグルさせることができます。



7.1.2. 手動 TAB 順序

手動タブ順序では、コントロール特性として **TAB 順序** という数値型の特性があり、ここに設定されている値が小さいものから大きなものへ、カーソルが移動していきます。



手動タブ順序のモードでは、**TAB 順序** 特性を開発者が設定できます。TAB 順序の値を開発者が変更した場合には、Magic は他のコントロールの TAB 順序を自動的に裁番しなおします。
TAB 順序 特性の値は、式で設定することもできます。式で設定した場合には、すべてのコントロールの式が評価され、ソートされ、値の小さい順に TAB 順序が決まります。

7.1.3. 自動 TAB 順序

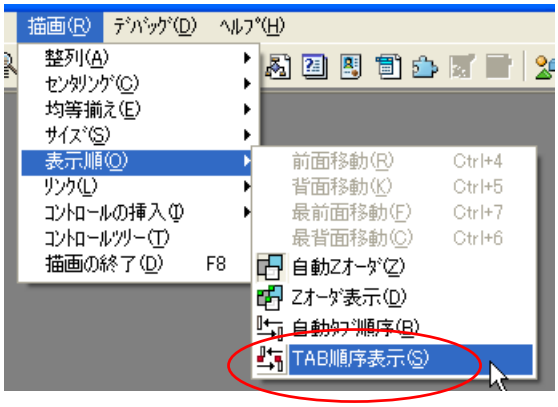
自動タブ順序では、フォーム上のコントロールを、次の順序で移動します。

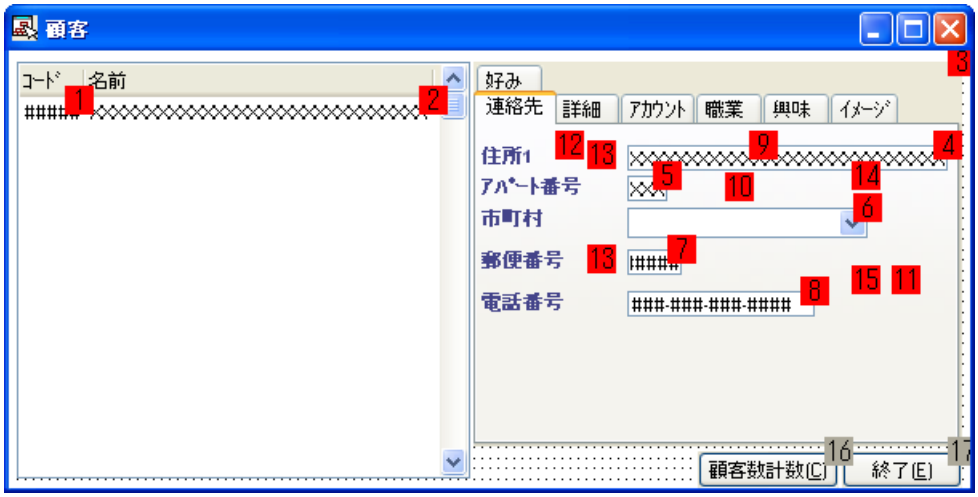
1. 画面上で上にあるコントロールから、下にあるコントロールへ。
2. 水平位置が同じ複数のコントロールがある場合には、左から右へ。

これは通常の場合、画面操作上、もっとも自然なカーソルの動きとなります。

7.1.4. TAB 順序の表示

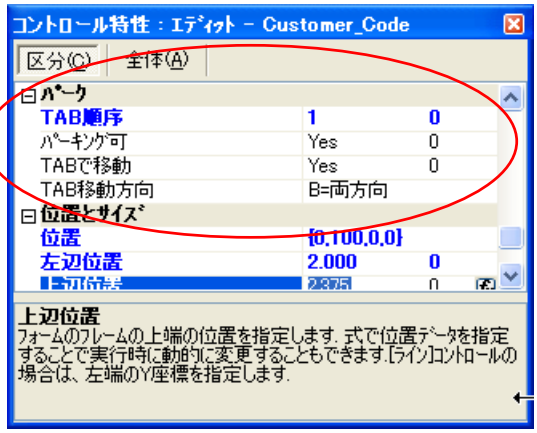
TAB 順序は、 **描画(R)** → **表示順(O)** → **TAB 順序表示(S)** メニュー、あるいは **コマンド** パレットの **TAB 順序表示** ボタンで表示させることができます。





7.2. GUI 特性の パーク カテゴリ

GUI コントロールには、**パーク** というカテゴリが新たに設定されました。

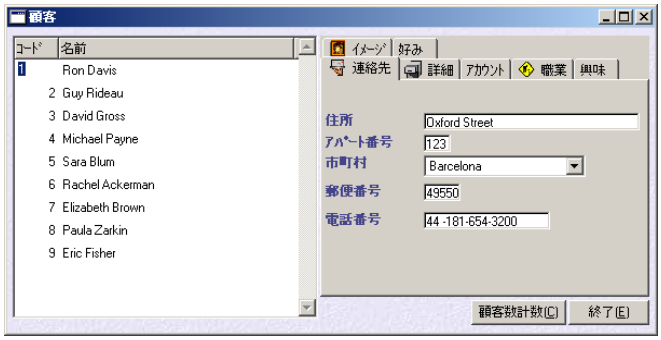


ここには、次のようなコントロール特性があります。

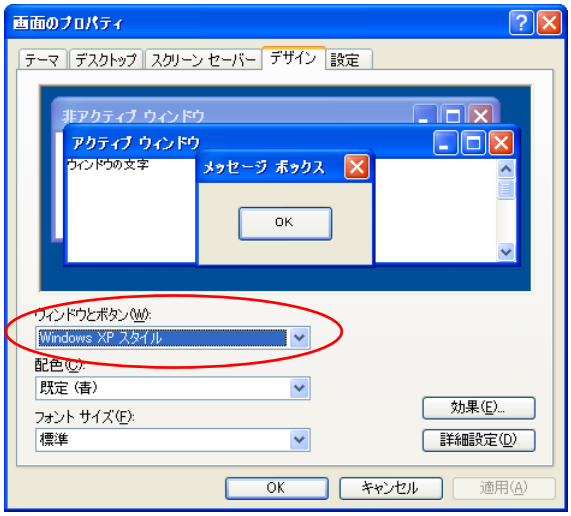
特性	説明	モデルリポジトリにあるか？
TAB 順序	手動 TAB 順序の場合、カーソルの移動する順序を数値で設定します。実行時には、値の小さい順にカーソルが移動します。 自動 TAB 順序の場合には、無効化され、開発者は設定できません。この場合には、値として、自動的に生成された TAB 順序の値が設定されます。	なし
パーキング可	カーソルがこのコントロールにパークすることができるかを設定します。 **Yes** の場合には、パーク可能です。**No** の場合には、TAB キー、マウスクリックに関わらず、パークすることはできません。	あり
TAB で移動	TAB キーによりカーソルを動かす場合に、カーソルがこのコントロールにパークするかを設定します。 **Yes** の場合には、パークします。**No** の場合にはパークしません。 いずれの場合にも、ユーザはマウスクリックによってカーソルをパークさせることは可能です。	あり
TAB 移動方向	TAB キーによりカーソルを動かす場合、カーソルの動く方向によりパークの可否を設定した場合に設定します。 **F=前方のみ** の場合には、順方向で動くときのみパークします。 **C=後方のみ** の場合には、逆方向で動くときのみパークします。 **B=両方向** の場合(デフォルト)には、いずれの場合もパークします。	あり

7.3. Windows コントロール

V10 では、Windows XP の GUI インターフェースをサポートするようになりました。
下図は、プログラム 8 番 **顧客** プログラムの実行画面です。XP のテーマを利用する場合(左側)と、利用しない場合(右側)の表示を見比べてください。



7.3.1. 前提条件



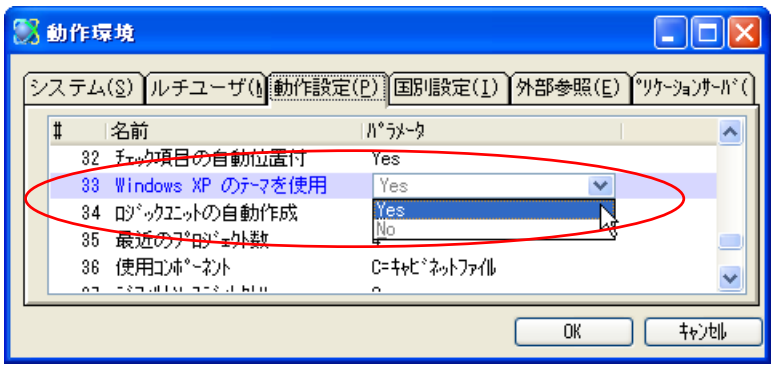
システム要件

Windows XP スタイルを利用する前提条件として、システムが次のような要件を満たしている必要があります。

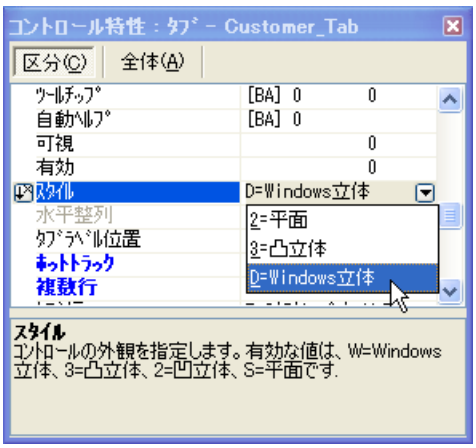
- Windows XP を使っていること。Windows 2000 では Windows XP スタイルは利用できません。
- 画面のプロパティで、デザインとして Windows XP スタイルが選択されていること。

動作設定

Windows XP の GUI インタフェースを有効化するかどうかは、**動作環境** ダイアログから **動作設定(P)** タブを開き、**Windows XP のテーマを使用** パラメータでで設定することができます。インストール時のデフォルトは **Yes** となっていますが、Windows XP の GUI インターフェースを利用したくない場合には、**No** を設定します。

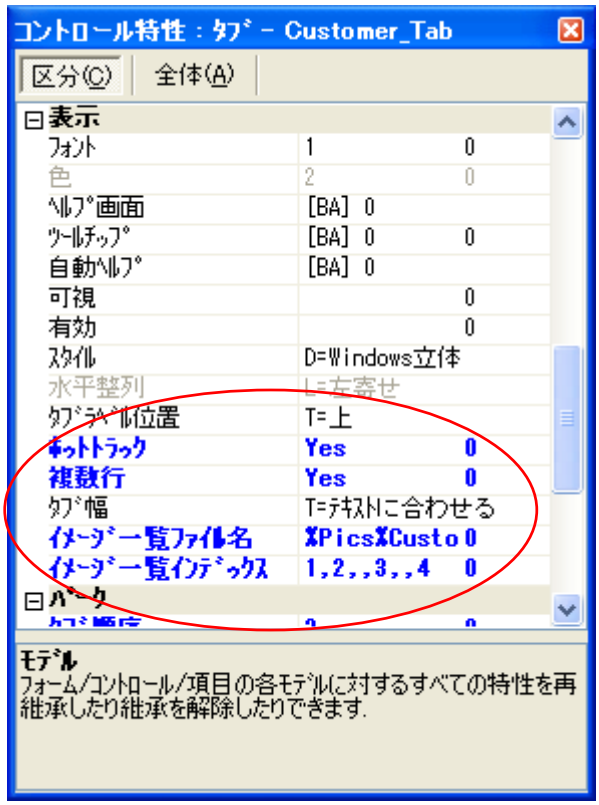


7.3.2. コントロール スタイルの設定



Windows XP スタイルを使いたい場合、各コントロールの スタイル 特性を **D=Windows 立体** に設定しておきます。これは V10 のデフォルト値です。

7.3.3. タブ コントロール



Windows XP スタイルの場合、タブコントロールは次のような特性が有効になります。

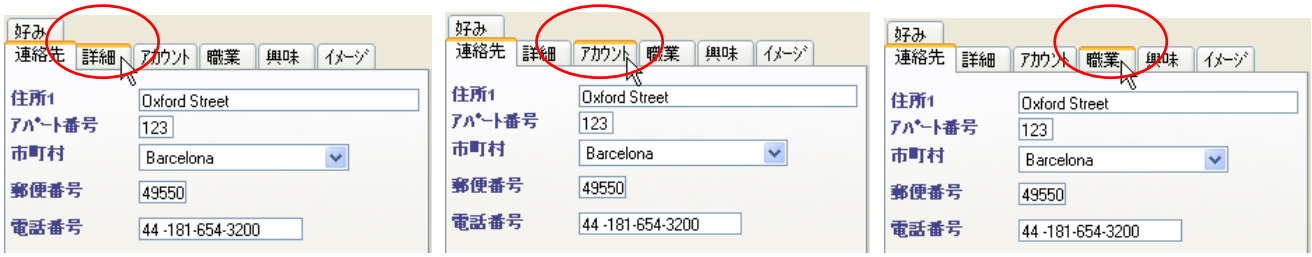
- ホットトラック
- 複数行
- タブ幅
- イメージ一覧ファイル名
- イメージ一覧インデックス

以下にそれぞれの機能について説明します。

ホットトラック

ホットトラックは、マウスカーソルの動きに合わせて、タブの色が変わる機能です。

下図は **顧客** プログラムの実行画面上のタブコントロールですが、マウスカーソルの位置にあるタブの上辺がオレンジ色で表示されています。



ホットトラックは、タブコントロールの **ホットトラック** 特性が Yes の場合に有効になります。デフォルトでは No です。

複数行

タブの数が多くて 1 行に収まりきらないとき、タブを 2 行以上に分けて表示するかを指定します。

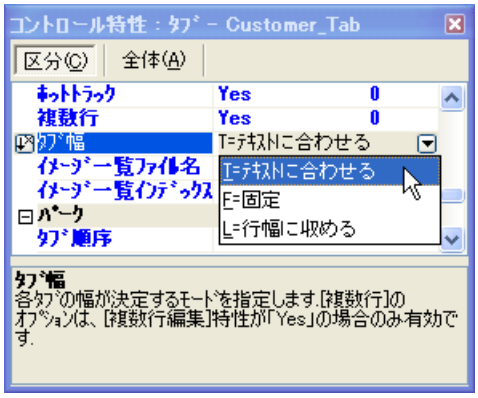
複数行 特性が No の場合(デフォルト値) には、タブは1行に表示できるだけ表示され、表示されないタブは、右端にあるスクロールボタンによって表示するようにします。

複数行 特性が Yes の場合には、2 行以上に分けてタブが表示されます。

タブ幅

タブ幅 特性は、次のような選択肢があります。

- T=テキストに合わせる（デフォルト）
- F=固定
- L=行幅に収める



T=テキストに合わせる の場合： タブの幅は、表示されているテキストに合わせて自動的に計算され調整されます。タブはすべて右寄せで配置されます。

F=固定 の場合：タブは全ての選択肢のテキストを収められる固定幅になります。

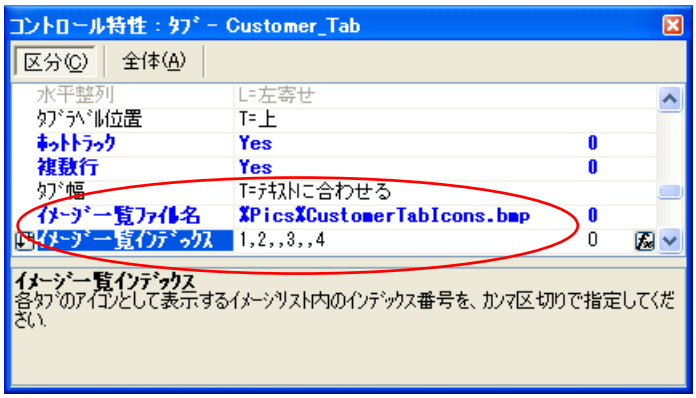
L=行幅に収める の場合：タブコントロールの幅に広がり、各タブは均等になるように表示されます。

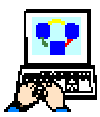
イメージ一覧ファイル と イメージ一覧インデックス

イメージ一覧ファイル と **イメージ一覧インデックス** 特性は、タブにアイコンイメージを表示させるための設定です。

イメージ一覧ファイル には、アイコンイメージをすべて一列に並べた形のビットマップイメージファイル名を指定します。

イメージ一覧インデックス は、このビットマップイメージの中の何番目のイメージを、何番目のタブに割り当てるかを、カンマで区切った数字で指定します。





例：タブにアイコンを右図のように付加したい場合には・・・

1. 次のような4つのアイコンを含む形のイメージファイルを作成し、`%Pics%CustomerTabIcons.bmp` に格納しておきます。

4 つのアイコンはそれぞれ、**連絡先**、**詳細**、**職業**、**イメージ** のタブに表示させるものとなります。

2. このイメージファイル名を、**イメージ一覧ファイル** 特性に設定します。
3. 各タブに対応するイメージの番号を **イメージ一覧インデックス** 特性に設定します。上の場合には、

```
1,2,,3,,4
```

と設定します。アイコンを表示させる必要がないタブは空にしておきます。

7.3.4. チェックボックス コントロール

V10 でのチェックボックスは、V9Plus に比べ、表示上のバラエティが多くなっています。また、3 ステートタイプ(真、偽 のほかに Null を持つ)もサポートしています。

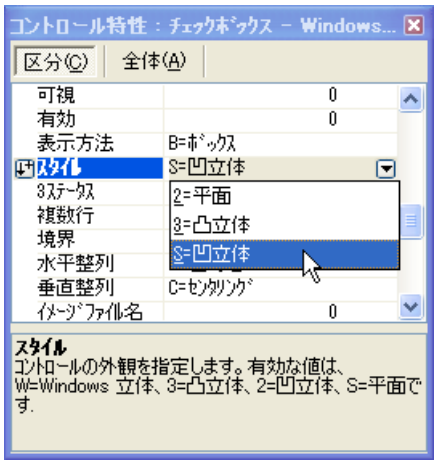
スタイル

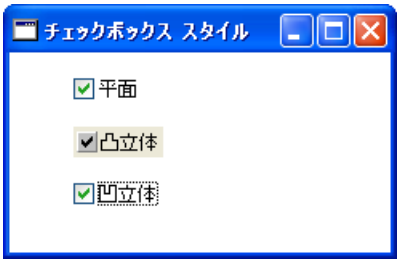
スタイル 特性は、

- 2=平面
- 3=凸立体、
- S=凹立体

から選択します。

実行時、**2=平面**、および **S=凹立体** の場合には、Windows チェックボックを表示します。 **3=凸立体**の場合には、V9Plus と同じ形式でチェックボックスを表示します。



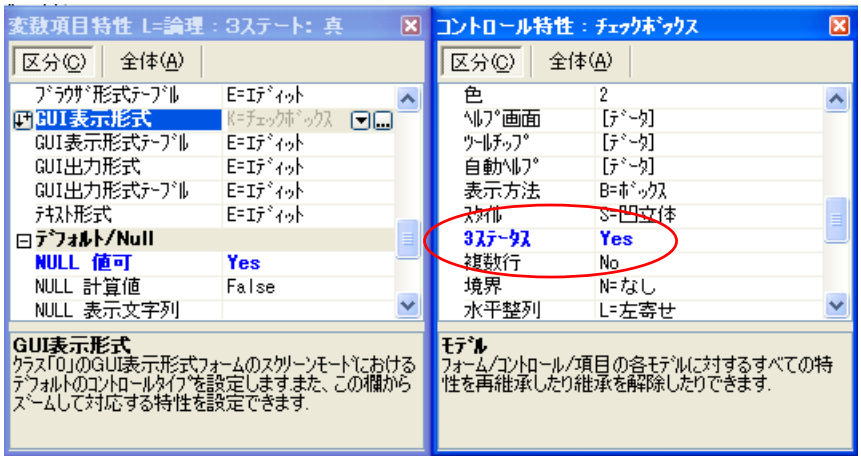


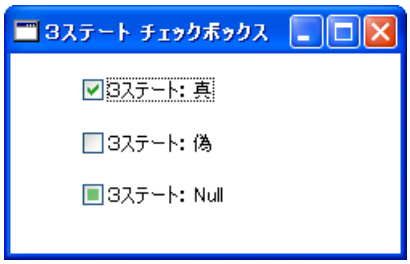
3 ステータス 特性

3 ステータス 特性は、Yes または No を設定します。デフォルトは No です。

Yes の場合には、真、偽 のほかに、「不明」の意味を表す Null も設定することができます。Null 値可 の場合には、対応する論路型項目の Null 値可 特性も Yes である必要があります。

実行時、Null は、 により表示されます。





表示方法

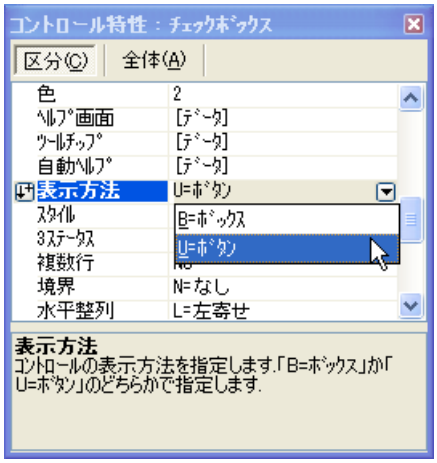
表示方法 特性は、

- B=ボックス（デフォルト）
- U=ボタン

を設定できます。

B=ボックス の場合は、普通のチェックボックスとして表示されます。

U=ボタン の場合には、ボタンで真偽状態を表示します。真の場合には、ボタンが押されたように、凹型に表示され、偽の場合には 凸型に表示されます。

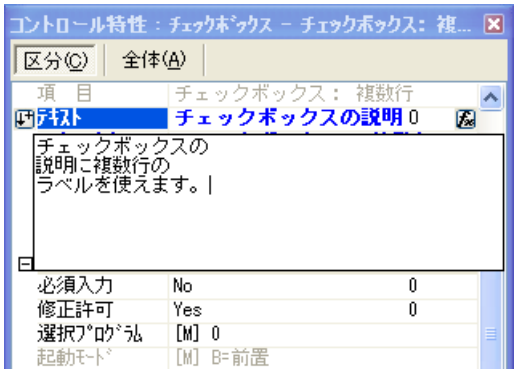


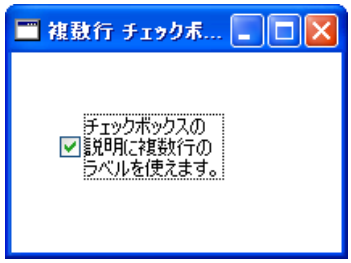
表示方法 ＝ **U=ボタン** の場合の表示

複数行

複数行 特性が Yes の場合には、ラベルを複数行に分けて表示することができます。

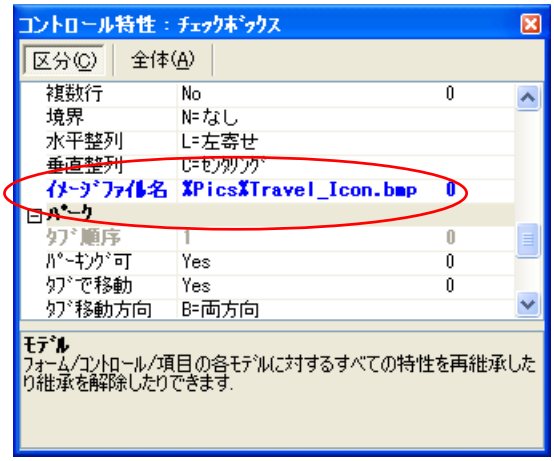
ラベルは、テキスト 特性で指定します。F6 で拡大表示させると、Enter キーを使って、改行の位置を指定することができます。





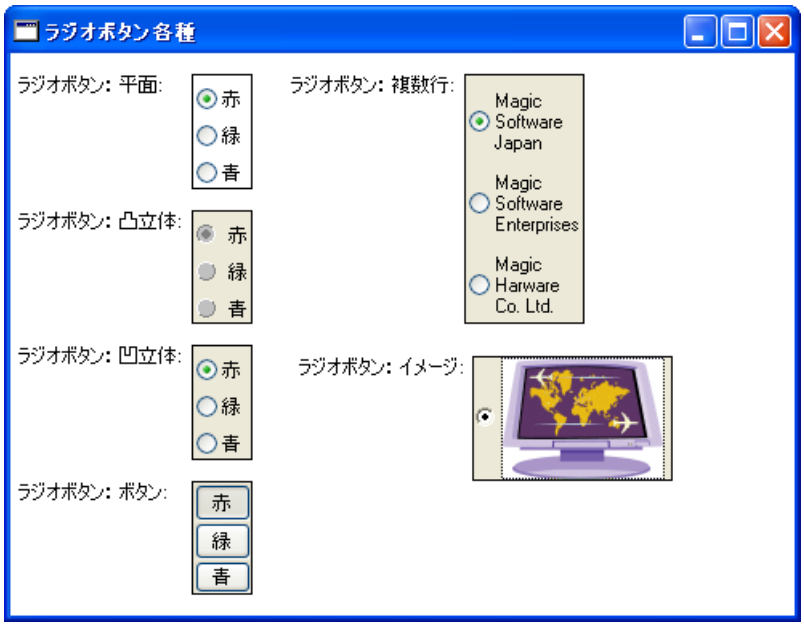
イメージファイル名

イメージファイル名 特性にイメージファイル名を指定すると、実行時にそのイメージがラベルの代わりに表示されます。



7.3.5. ラジオボタン コントロール

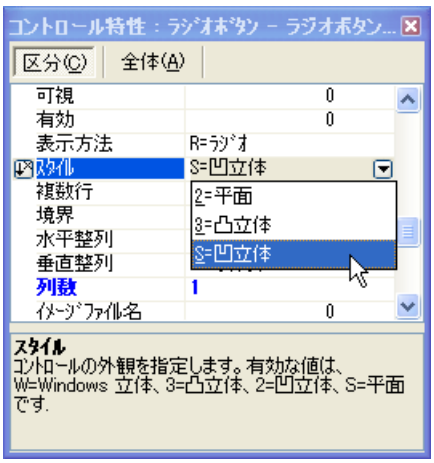
ラジオボタンコントロールも、表示上のバリエーションが増えました。

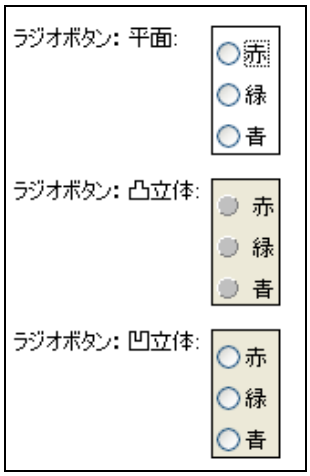


スタイル

スタイル 特性は、次のものから選択します。

スタイル 特性値	説明	備考
2=平面	平面で表示します。	Windows のラジオボタンを使って表示します。
3=凸立体	3D で盛り上がるようなイメージで表示します。	V9Plus と同様な表示方法です。
S=凹立体	3D で引っ込むようなイメージで表示します。	V10 でのデフォルト。Windows のラジオボタンを使って表示します。

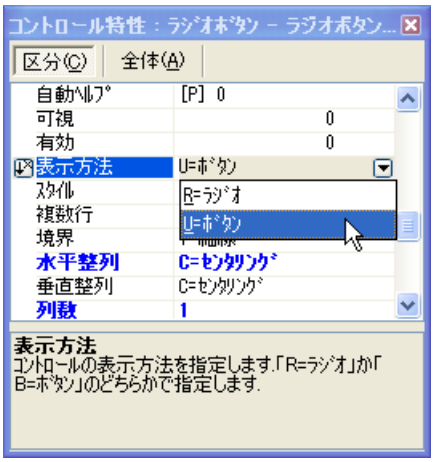


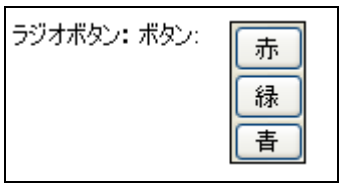


表示方法

表示方法 特性は、次の値から選択します。

表示方法 特性値	説明	備考
R=ラジオ	通常のラジオボタンの表示です。	デフォルトの設定
U=ボタン	各々の選択肢をボタンで表示します。	



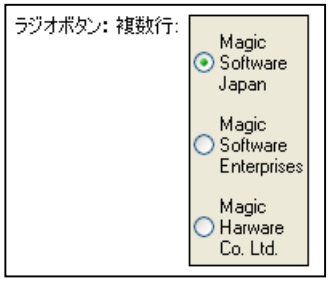


表示方法 = **U=ボタン** のラジオボタン

複数行

複数行 特性は、ラベルを複数行に分けて表示するかを指定するもので、Yes / No あるいは論理式で指定します。

右図は、**複数行** = **Yes** の場合の表示例です。



イメージ

イメージ特性にイメージファイル名を指定した場合には、そのイメージがラベル代わりに表示されます。

この場合、実行時の動作は次のようになります。

- このラジオボタンは、選択肢(**選択項目リスト**、および **選択表示リスト** 特性)が一つだけのものとします。複数の選択肢がある場合には、同じイメージが複数表示されてしまいます。
- 実行時には、選択、非選択、の二つの状態しかありません。
- クリックすると選択された状態になります。この後には、マウスクリックで非選択の状態にすることはできません。

イメージ ラジオボタンを単独で使った場合、いったん選択すると非選択に戻すことができないので、通常は単独で用いることはなく、次に説明するグループ化機能を用いて、グループの中の一選択項目として用います。

ラジオボタンのグループ化

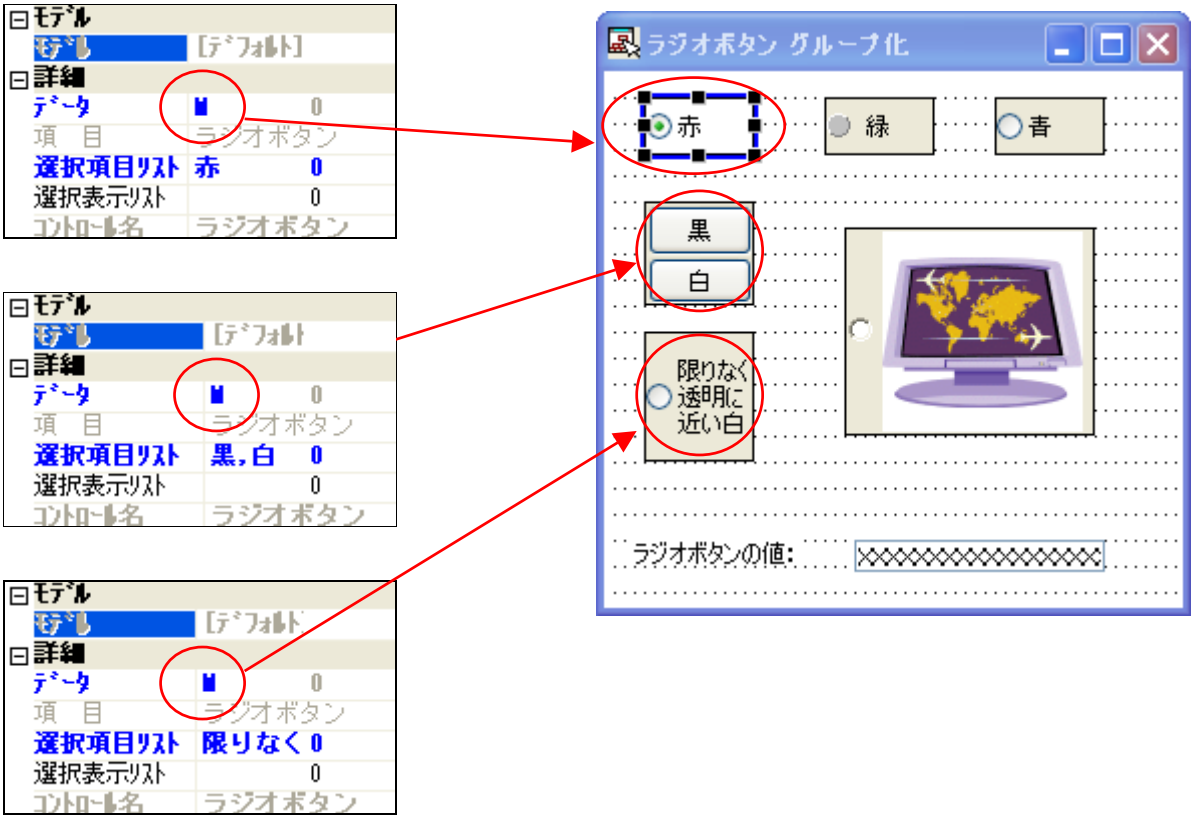
V9Plus までは、ラジオボタンの選択肢は一つのコントロールとして、1 列、あるいは 1 行に、同じスタイルで表示することしかできませんでした。

V10 では、複数のラジオボタンコントロールを、一つの変数に結び付けることができるようになりました。これを **ラジオボタンのグループ化** と呼びます。右図は、すべて異なったスタイルを持つラジオボタンが 6 つありますが、これらはすべて同一の変数に結び付けられています。

この場合、

- グループ化は、フォームエディタ上で、各ラジオボタンに、**データ** 特性として同一の項目を割り当ててやることにより設定します。
- 各ラジオボタンコントロールのスタイルはそれぞれ独立して指定できますので、異なるスタイルのラジオボタンであってもグループ化することができます。
- これらのラジオボタンは、物理的に隣接している必要はなく、画面上のどこにでも配置することができます。
- グループ化されているラジオボタンは、お互いに排他的に選択されます。例えば、上の図では、イメージで表示されているラジオボタン(内部の値は 紫）が選択されていますが、このときには他のラジオボタンは非選択の状態になっています。ここで、例えば左上の 赤 をクリックすると、赤が選択状態になり、紫（イメージのラジオボタン)は非選択となります。
- グループを構成する各々のラジオボタンには、複数の選択肢を持たせることができます。例えば、左側上から 2 番目のラジオボタン（ボタンのスタイルで表示されているもの）では、**黒** と **白** の二つの選択肢があります。

上の例を、スタジオのフォームエディタ上で見ててみると、すべて **データ** 特性に、項目 M が割り当てられています。他のラジオボタンも同様に、項目 M が割り当てられています。



グループを構成する各ラジオボタンでは、**選択項目リスト** 特性に、そのラジオボタンでの選択肢を設定します。
上の例では、左上のボタンは **赤**、その下のボタンは **黒** と **白**、その下は **限りなく透明に近い白** などとなっています。

- フォームエディタ上でいずれかのラジオボタンをクリックすると、グループ化されたラジオボタンがすべて選択されます。
- 個々のラジオボタンだけを選択したい場合には、Ctrl キーを押しながらクリックします。

7.4. サブフォーム

Magic では、親子タスクでヘッダレコードと明細レコードを表示させる画面設計が良く使われます。このような場合、従来は **ファントムタスク** としてプログラムを作りましたが、これには独特のプログラミング技法と、微妙な画面設計が必要でした。

サブフォーム は、ファントムタスクを、より簡単に作りやすくする、V10 の新機能です。

V9Plus でも、ブラウザクライアントタスクでサブフォームを使うことができました。V10 では、ブラウザクライアントでも引き続きサブフォームを使うことができますが、オンラインプログラムでも作れるようになりました。サブフォームやサブタスクの設定のしかたなどは、V9Plus でのブラウザクライアントのサブフォームと似ています。

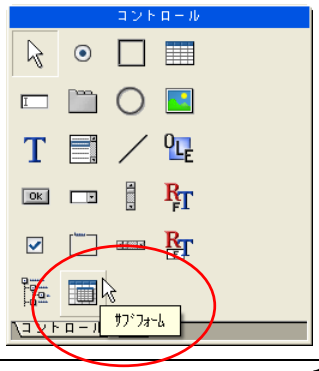
サンプルでは、プログラム 9 番 **注文(サブフォーム利用)** でサブフォームを使っています。

7.4.1. サブフォームをフォームに配置する

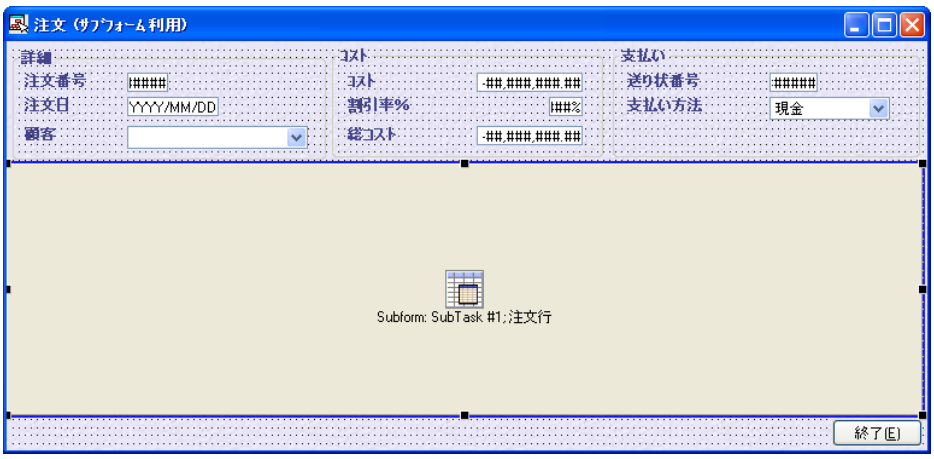
サブフォームは、オンラインの GUI フォームで使えます。

サブフォームを配置・設定するには・・・

1. オンラインの GUI フォームをフォームエディタで開きます。
2. コントロールパレットから、サブフォーム コントロールを選びます。

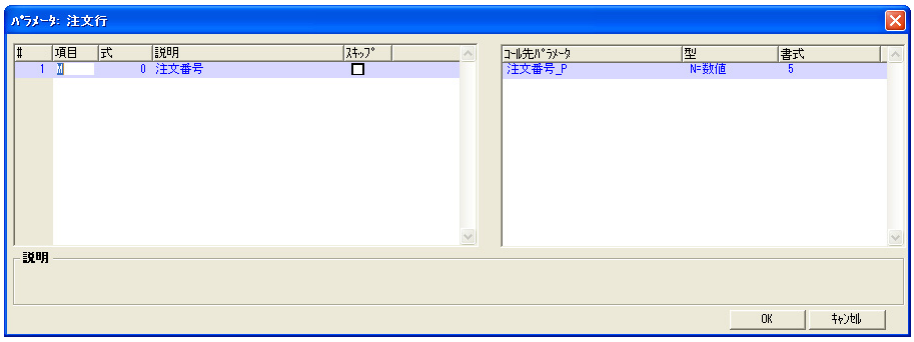


3. サブフォームコントロールをフォームにドロップし、サイズを適当に変更します。実行時には、このサブフォームの中にサブタスクの画面が表示されます。



4. サブタスクコントロールのコントロール特性を開きます。
5. **接続先** および **プログラム/タスク番号** 特性は、このサブフォーム中に表示させるタスクの指定を行います。接続先 特性は、**S=サブタスク** あるいは **P=プログラム** のいずれかであり、**プログラム/タスク番号** は、呼び出すプログラム番号（**P=プログラム** の場合）、あるいは サブタスクの番号（**S=サブタスク** の場合）を指定します。
6. **パラメータ** 特性では ズームしてパラメータテーブルを開き、呼び出すタスク／プログラムに渡すパラメータを指定します。

以上で、サブフォームの設定は終わりです。



従来の ファントムタスク の構造では、親・子タスクに次のような特別な設定が必要でした:

- 親タスクでは、レコード前処理とレコードメインで、子タスクを呼び出す。
- 子タスクでは、**終了条件** に Level(1) = 'RP' を設定し、タスク制御の **ウィンドウを閉じる** パラメータを No にする。

サブフォームを使った場合には、これらの設定は不要になります。

V9Plus との互換性のために、従来のファントムタスクの構造もそのまま動作します。

7.4.2. サブフォーム タスク実行ルール

サブフォームで指定されているプログラムあるいはサブタスク（サブフォームタスク)は、以下のタイミングで呼び出さ

れます。

- 親プログラムが起動され、親タスクの最初のレコード前処理が終了した直後。
- 親タスクでのタブの移動で、サブフォームタスク コントロールに達したとき。
- ユーザがサブフォームタスク上の画面領域をクリックした場合。

サブフォームタスクが起動されたときには、次のことが起こります。

1. フォーカスはサブフォームタスクに移動します。
2. ユーザがサブフォームタスクのパーク可能なコントロール上でクリックした場合には、そのコントロールにカーソルが移動します。それ以外の場合には、タブ順序で最初のコントロールにカーソルが移動します。
3. タスク前処理が実行されます。
4. レコード前処理が実行されます。
5. カーソルの移動先のコントロールに、コントロール検証 ハンドラが定義されていれば、それが実行されます。
6. 同じく、コントロール前処理 ハンドラが定義されていれば、それが実行されます。

7.4.3. サブフォーム タスク特性

サブフォームコントロールから呼び出される場合、そのタスクは次のようなタスク特性を持っているものとして実行されます。

タスク特性 パラメータ	値
ウィンドウ消去	No
前面表示	Yes
アイコンファイル名	無視される

その他の注意点は以下の通りです。

- 子タスクの トランザクションモード が **W=親と同一**、あるいは **D=遅延** となっている場合には、サブフォームタスクは、あたかも **同期** が Yes として設定されたコールコマンドで呼び出されたのと同様に動作します。
- サブフォーム タスクを呼び出す場合、親タスクのレコードで暗黙のロックはかかりません。

7.4.4. サブフォーム タスク終了ルール

サブフォームタスクは、次のタイミングで終了します。

- ユーザがサブフォーム コントロールの外の領域でクリックした場合。
- **C:閉じる** イベントが発行されたとき。
- **終了** 条件が真になったとき。
- 親タスクが終了したとき。

サブフォーム タスクが終了するとき、次のようなことが行われます。

1. 現在パークしているコントロールにコントロール後処理が定義されていたら、それを実行します。
2. レコード後処理が実行されます。

3. タスク後処理が実行されます。
4. フォーカスが親タスクに戻り、カーソルは TAB 順序に従い、サブフォームコントロールの次のコントロールに移ります。

サブフォーム コントロール自体に、コントロール前処理/後処理/検証 などのハンドラを定義することはできません。

7.4.5. サブフォームの自動再表示

サブフォームの内容は、次の条件が両方満たされている場合に自動的に再表示されます。

- サブフォーム コントロールの **自動再表示** 特性が Yes になっていること。No になっている場合には、自動再表示はされません。
- 親タスクの処理の結果として、パラメータとしてサブフォーム タスクに渡される値に変更があった場合。親タスクの項目に変更があり、サブフォームタスクがその項目を参照していたとしても、パラメータとして渡されていない場合には、自動再表示はされません。

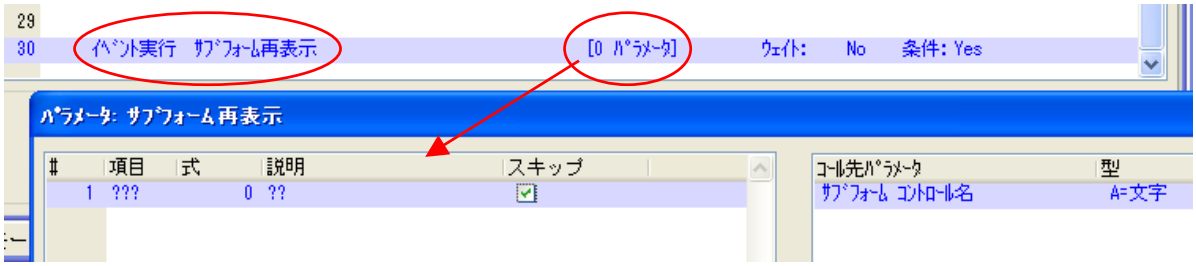
サブフォーム コントロールの **自動再表示** 特性が No になっている場合でも、次に説明する **サブフォーム再表示** イベントを発行すれば、サブフォームの内容が強制的に再表示されます。

7.4.6. サブフォーム再表示イベント

サブフォーム再表示 イベントを発行すると、そのタスクに配置されているサブフォームタスクが、強制的に再表示させられます。
例えば、親タスクで検索条件を設定し、サブフォームタスクで検索結果を表示させるようなプログラムでは、ユーザがパラメータをすべて入力した後に検索をかけたい、という場合が多いと思いますが、このような場合に、**自動再表示** 特性を No にしておいて、**検索** ボタンなどに **サブフォーム再表示** イベントを割り当てておくことで、そのような動作をさせることができるようになります。

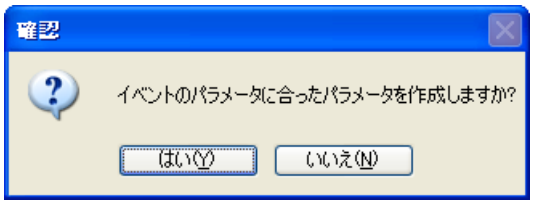
パラメータ

サブフォーム再表示 イベントには、**サブフォームコントロール名** というパラメータがあります。
イベント実行 コマンドで **サブフォーム再表示** イベントを発行する場合には、パラメータテーブルでこのパラメータを与えることができます。



他の内部イベントと同様に、**サブフォーム再表示** イベントは非同期で実行されます。このため、**イベント実行** コマンドの **ウェイト** パラメータは No (内部イベントを選択した場合のデフォルト値)にしておかなければなりません。**ウェイト** を Yes にした場合には、イベントの効果はありません。

サブフォーム再表示 イベントに対するイベント ハンドラを定義する場合には、パラメータも共に定義するかどうかを聞かれます。
Yes と答えると、**サブフォーム コントロール名** という名前のパラメータが定義されます。




```
29
30 ⊟E=ｲﾍﾞﾝﾄ      ｻﾌﾞﾌｫｰﾑ再表示          ｺﾝﾄ                    ｽｺｰﾌﾟ S=ｻﾌﾞﾂﾘｰ
31     項目        P=ﾊﾟﾗﾒｰﾀ      5   ｻﾌﾞﾌｫｰﾑ ｺﾝﾄﾛｰﾙ名          A=文字    30
```

7.4.7. サブフォーム関連の関数

サブフォームに関連して、次のような関数が使えます。

関数	説明
CtrlGoTo	CtrlGoTo 関数のパラメータとして、サブフォーム コントロールのコントロール名を指定することができます。この場合には、次のことが起こります。 1. 現在パーク中のコントロールのコントロール後処理が実行されます。 2. TAB 順序に従って、現在パーク中のコントロールから、サブフォームコントロールまでの間にあるコントロールに定義されているコントロール検証が実行されます。 3. サブフォームタスクにフォーカスが移り、最初のパーク可能なコントロールにカーソルが移動します。
HandleCtrl	この関数が サブフォーム コントロールのコントロール名を返すことはありません。
Level	サブフォーム タスク中から Level(1) を実行すると、'Subform' という文字列を返します。
MainLevel	MainLevel 関数は、通常の子タスクと同様に動作します。
SubformExecMode	これは V10 の新しい関数で、タスクがサブフォームタスクとして実行されているかどうかを判定するために用います。詳細は以下に説明します。

SubformExecMode 関数

SubformExecMode 関数は V10 での新しい関数で、呼び出し形式は以下の通りです。

シンタックス `SubformExecMode(世代)`

パラメータ **世代** パラメータは、タスクの世代（現タスク = 0、呼び出しタスク=1、・・・)を表します。

戻り値

－1 タスクはサブフォームタスクとして実行されていません。

0 タスクはサブフォームタスクとして通常に実行されています。

1 タスクはサブフォームタスクですが、親タスクから最初に呼び出されて実行されています。

2 タスクはサブフォームタスクで、再表示のために実行されています。(これには、自動再表示の場合も、サブフォーム再表示 イベントによる再表示のいずれの場合も含みます)。

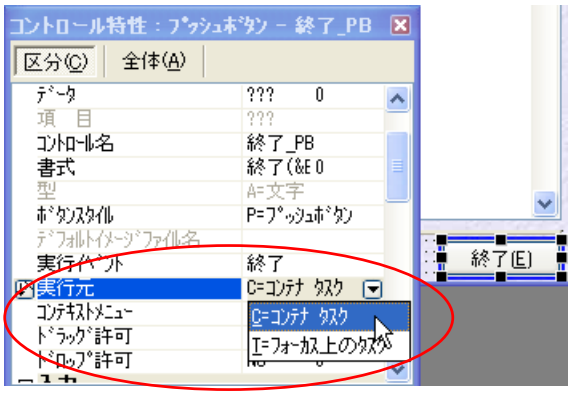
7.5. プッシュボタン コントロール

7.5.1. 実行元 特性

プッシュボタンコントロールでイベントが発行される場合、どのタスクから発行されるのかは、Magic エンジンがイベントハンドラの検索順序を決定する上で必要な情報です。
ファントムタスクの構造をしている場合、V10 では親タスク(コンテナタスク)、あるいはフォーカスのあるタスクのいずれかから発行されるかを、プッシュボタンの **実行元** 特性により設定できるようになりました。

実行元 特性には、次の二つの設定があります。

実行元 特性値	意味
C=コンテナタスク	(デフォルト値) コンテナタスクからイベントが発行されます。
T=フォーカス上のタスク	現在フォーカスのあるタスクからイベントが発行されます。



右図は、プログラム 9 番の **注文(サブフォーム利用)** プログラム上に配置された **終了(E)** ボタンのコントロール特性です。ここでは、デフォルトの **C=コンテナ タスク** になっています。
このプッシュボタンでは、**実行元** 特性が **C=コンテナタスク** になっているので、フォーカスが親タスクにあっても、サブフォームタスクにあっても、このボタンを押すことによって、親タスクが終了します。
もし、このボタンの **実行元** 特性が **T=フォーカス上のタスク** だったとすると、フォーカスがサブフォームタスク上にあるときにこの **終了(E)** ボタンを押すと、サブフォームタスクが終了してフォーカスが親タスクに戻るだけで、親タスクは終了しません。

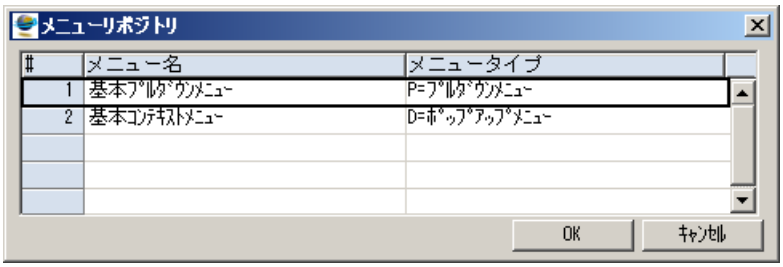
7.6. メニュー

V9Plus では、メニューリポジトリにおいて、1 番は基本プルダウンメニュー、2 番は基本コンテキストメニュー と固定されていました。
V10 ではこのような決まりはなくなり、プルダウンメニューやコンテキストメニューは自由に設定できるようになりました。また、コントロール単位でメニューを設定できるようになりました。

7.6.1. メニューリポジトリ

メニュータイプ

V9Plus でのメニューリポジトリ



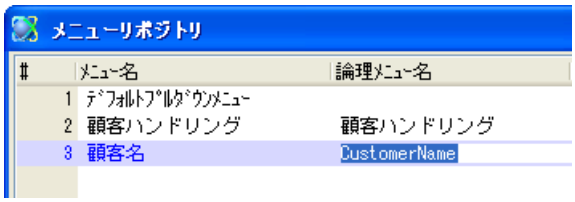
V9Plus では、メニューリポジトリの各エントリには、**メニュータイプ** があり、

- **P=プルダウンメニュー**
- **D=コンテキストメニュー**
- **C=ショートカットメニュー**

などが設定されていて、各メニューの用途は固定されていました。

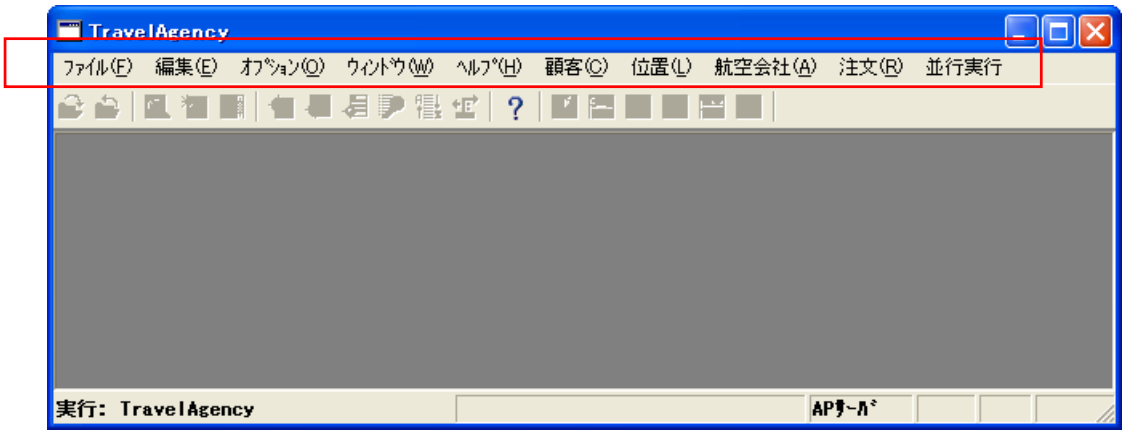
V10 では、**メニュータイプ** の指定はなくなり、どのメニューエントリでも、いずれの用途にでも利用することができます。

V10 でのメニューリポジトリ

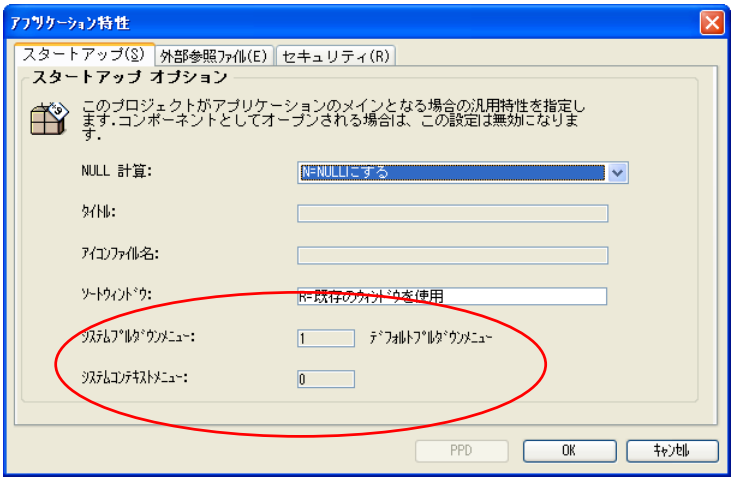


システムプルダウンメニューとシステムコンテキストメニューの設定

実行エンジンでアプリケーションを開いたとき、デフォルトのプルダウンメニューを **システムプルダウンメニュー** と呼びます。また、アプリケーションを開いた直後、まだプログラムが実行されていない状態で、右マウスクリックで表示されるコンテキストメニューを **システムコンテキストメニュー** と呼びます。
下図は、サンプルアプリケーションのシステムプルダウンメニューです。

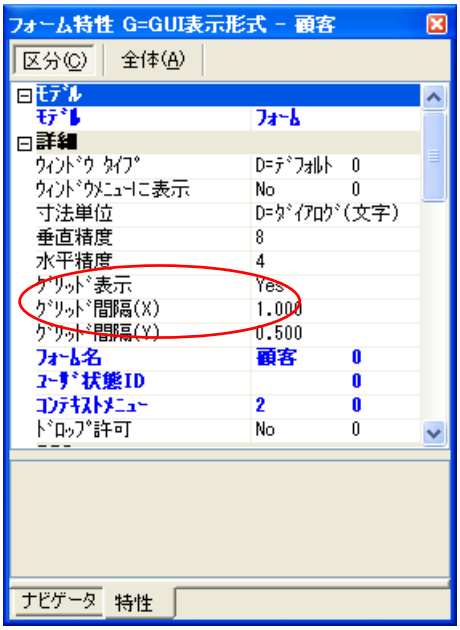


システムプルダウンメニューとシステムコンテキストメニューは、アプリケーション特性の **システムプルダウンメニュー** および **システムコンテキストメニュー** パラメータで、メニューエントリの番号で設定します。



7.6.2. フォームにコンテキストメニューを設定する

V10 では、フォームごとにコンテキストメニューを設定することができます。フォームごとのコンテキストメニューは、フォーム特性の **コンテキストメニュー** 特性で、メニューのエントリ番号で指定します。

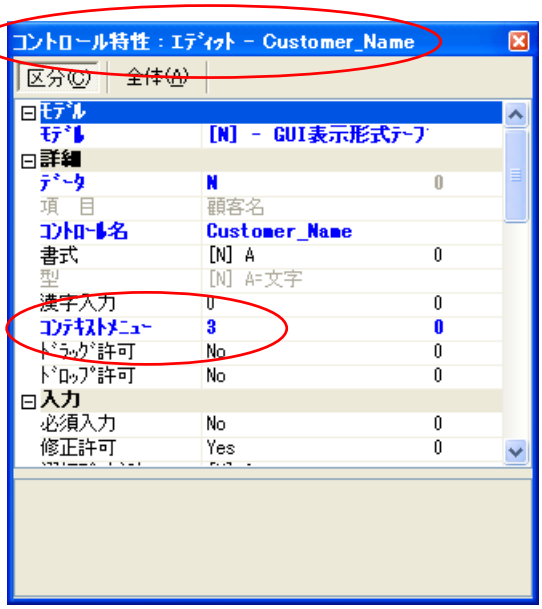


- **コンテキストメニュー** 特性は、式で設定することもできます。式で設定する場合には、 **MENU** リテラルを使うと便利です。
- V9Plus では、タスク特性でメニュー番号を指定することができましたが、V10 ではフォーム特性で設定します。

7.6.3. コントロールにコンテキストメニューを設定する

V10 では、個々のコントロールにコンテキストメニューを設定することもできます。このメニューは、コントロールにフォーカスがあるときに有効となります。

コンテキストメニューは、コントロール特性の **コンテキストメニュー** 特性で、メニュー番号で設定します。ここには式を指定することもできます。



7.6.4. コンテキストメニューの表示ルール

実行時、あるコントロールの上でマウスの右クリックがなされたら、次のルールに従ってコンテキストメニューが表示されます。

1. もしこのコントロールに **コンテキストメニュー** 特性が設定されていれば、それに従ってコンテキストメニューが表示されます。

2. コントロールに**コンテキストメニュー** 特性が設定されていない場合には、コントロールのタイプに固有なコンテキストメニューが表示されます。例えば、エディットコントロールには、切り取り、コピー、貼り付け、IME の制御その他のコンテキストメニューがあります。

3. もしコントロール タイプに固有なコンテキストメニューが定義されていなければ、フォームの **コンテキストメニュー** 特性に設定されているメニューが表示されます。

4. フォームにもコンテキストメニューが定義されていなければ、コンテキストメニューは表示されません。

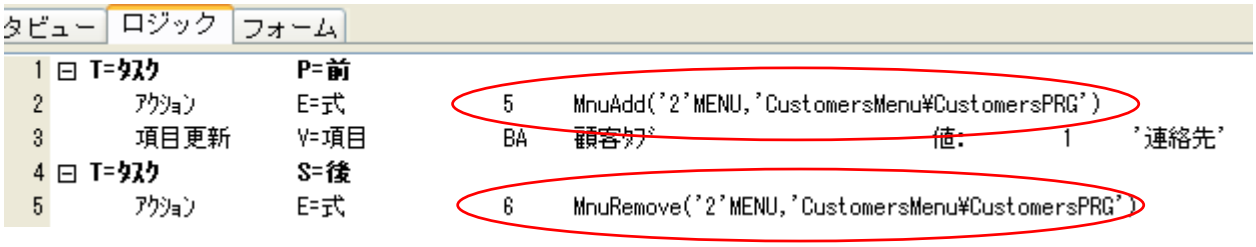
- コントロールが無効化されている状態では、コンテキストメニューは表示されません。
- OLE と ActiveX コントロールの場合には、それに固有なコンテキストメニューが表示されます。

7.6.5. 実行時にプルダウンメニューにメニューを追加する

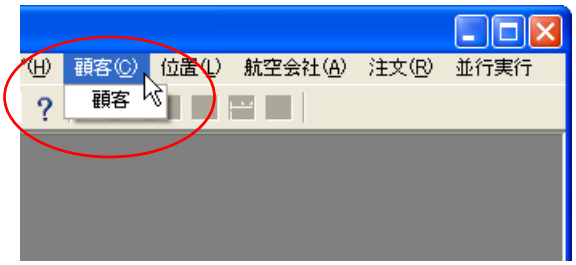
メニューは、実行時にダイナミックに追加・削除することができます。このために次の関数が定義されています。

関数	意味
MnuAdd	プログラムによって動的にメニュー構造を現在実行中のプルダウンメニュー構造に追加します。
MnuRemove	プログラムによって動的にメニュー構造を現在実行中のプルダウンメニュー構造から削除します。

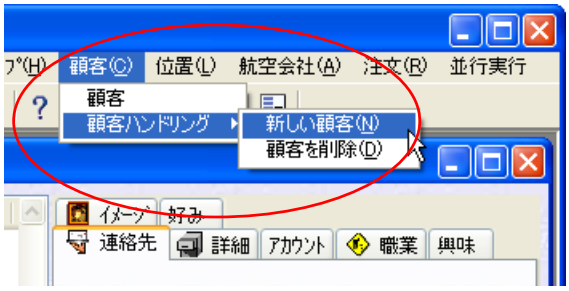
サンプルでは、プログラム8番 顧客 プログラムで、この関数が使われています。ここでは、タスク前処理で **顧客** メニューの追加し、タスク後処理で削除しています。



この結果、このタスクが実行中の間だけ、**顧客** メニューのエントリが増えています。



通常の場合



顧客 プログラム実行中の場合

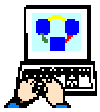
7.7. フォーム状態の保存と復元

V9Plus までは、フォームのサイズやコントロールのサイズ、位置などをいったんマウスで変更しても、タスクを閉じて再度開いた場合には、すべてリセットされてしまいました。V10 ではフォーム特性の設定をひとつ入れるだけで、ユーザごと、フォームごとに閉じたときの状態を保存し、次回に自動的に復元されるようにできます。保存情報は Windows にログオンしているユーザごと別々に格納されます。

7.7.1. フォーム状態 ID 特性

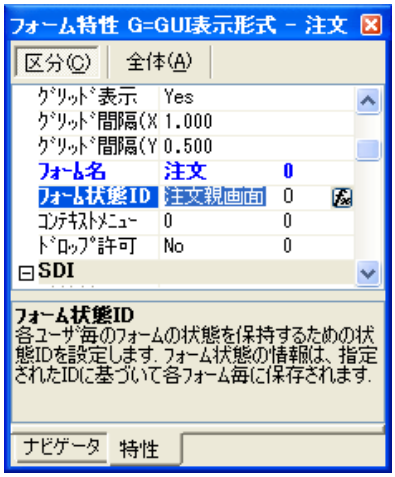
デフォルトでは、フォームの状態は保存されず、V9Plus までと同じ動作となります。フォームの状態を保存したい場合には、フォーム特性の「フォーム状態 ID」特性に、識別 ID を設定します。

例として、プログラム 6 番 **注文** にフォーム状態 ID を設定してみましょう。

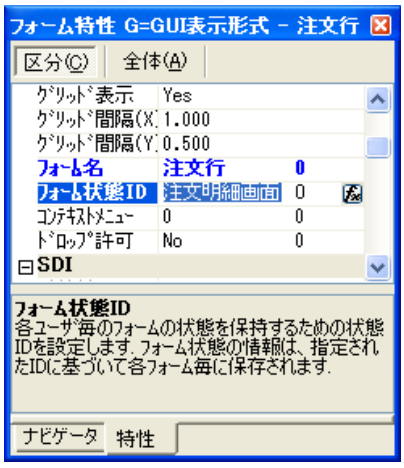


フォーム状態を保存・復元するようにするには・・・

1. プログラム 6 番 **注文** を開き、**フォーム** タブをクリックしてフォーム一覧を表示させます。
2. **注文** フォームのフォーム特性を開きます。
3. **フォーム状態 ID** に **注文親画面** と指定します。
4. タスクを閉じます。



5. これで、親タスクのフォームについては、状態が保存・復元されますが、**注文** プログラムは親子タスクの構造をもっているので、子タスクのフォームの状態は保存・復元されません。親子両方のフォーム状態を保存・復元するために、子タスク 注文行 のフォームについても、同様に、**フォーム状態 ID** 特性に **注文明細画面** などと設定してください。
6. タスクを閉じます。



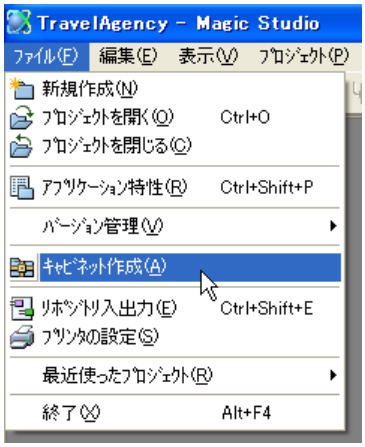
7.7.2. フォーム状態 ID の効果の確認

これで設定ができたので、次にフォーム状態 ID の動作の確認をしてみましょう。

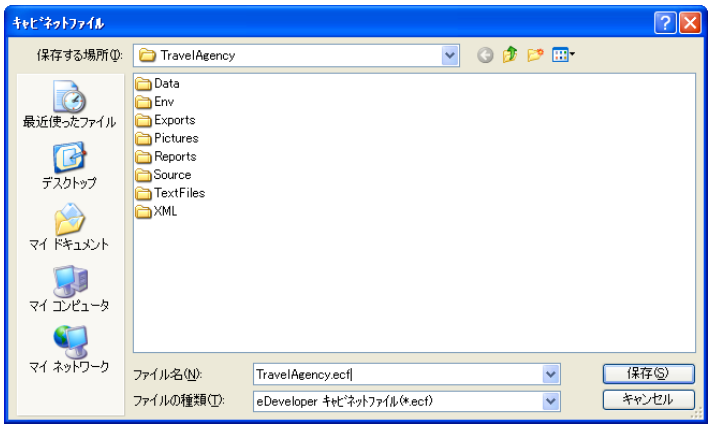
ここで注意があります。フォーム状態 ID は、実行版でアプリケーションを実行している場合にのみ有効で、スタジオから F7 キーなどでから実行する場合には、つねにフォーム状態がリセットされてしまいます。実行版で実行するには、プロジェクトを **キャビネットファイル** という形式に変換して保存する必要があります。キャビネットファイルというのは、V9Plus でのフラットファイル（MFF）と同じもので、詳しくは 13.2 アプリケーションの配備 で説明しますが、ここでは、フォーム状態 ID の動作確認のために、簡単にキャビネットファイルの作成方法を説明します。

キャビネットファイルに保存するには・・・

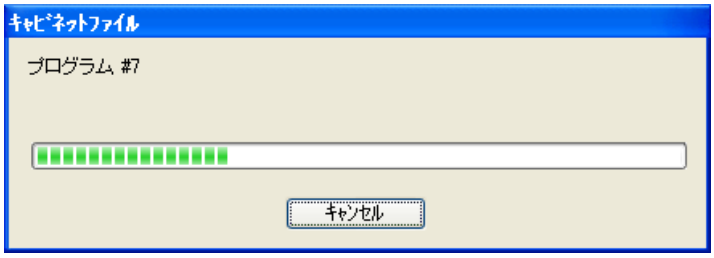
1. 開いているリポジトリがあれば、すべて閉じます。アプリケーションは開いたままにしておきます。
2. **ファイル** メニューから、**キャビネット作成** を選択します。



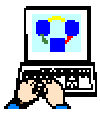
3. キャビネットファイル 保存のダイアログが開くので、TravelAgency ディレクトリの下に、TravelAgency.ecf と指定して、保存(S) をクリックします。



4. プログレスバーが消えれば、キャビネットファイル作成完了です。

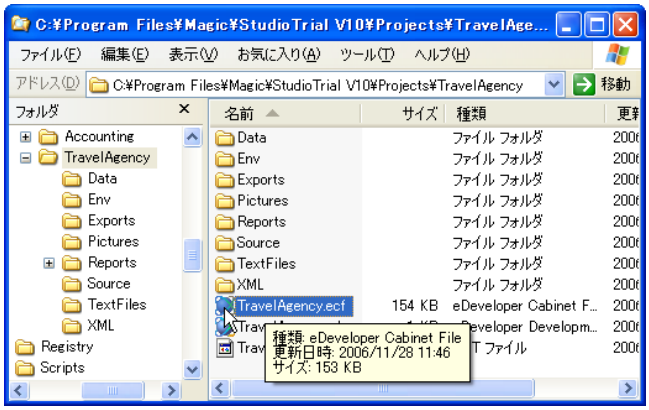


キャビネットファイルができたら、実行版で実行してみましょう。

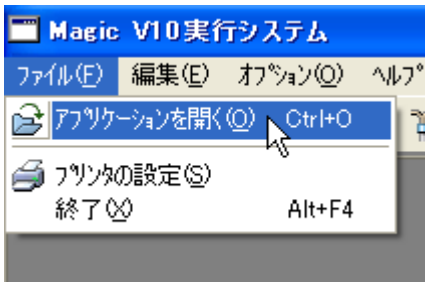


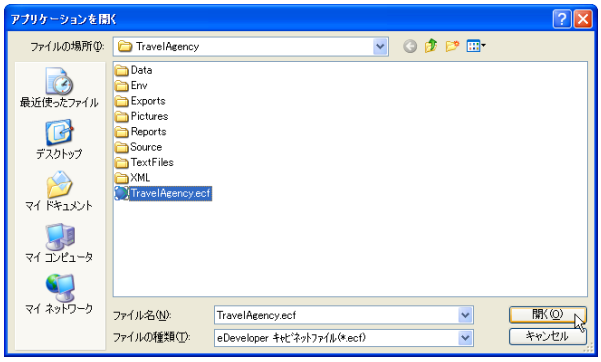
キャビネットファイルを実行するには・・・

1. キャビネットファイルを実行するには、エクスプローラからキャビネットファイルをダブルクリックするのが一番簡単です。

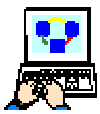


2. その他に、実行版を起動してから、**ファイル** メニュー より **アプリケーションを開く** で、キャビネットファイルを指定して開いて、実行する方法もあります。



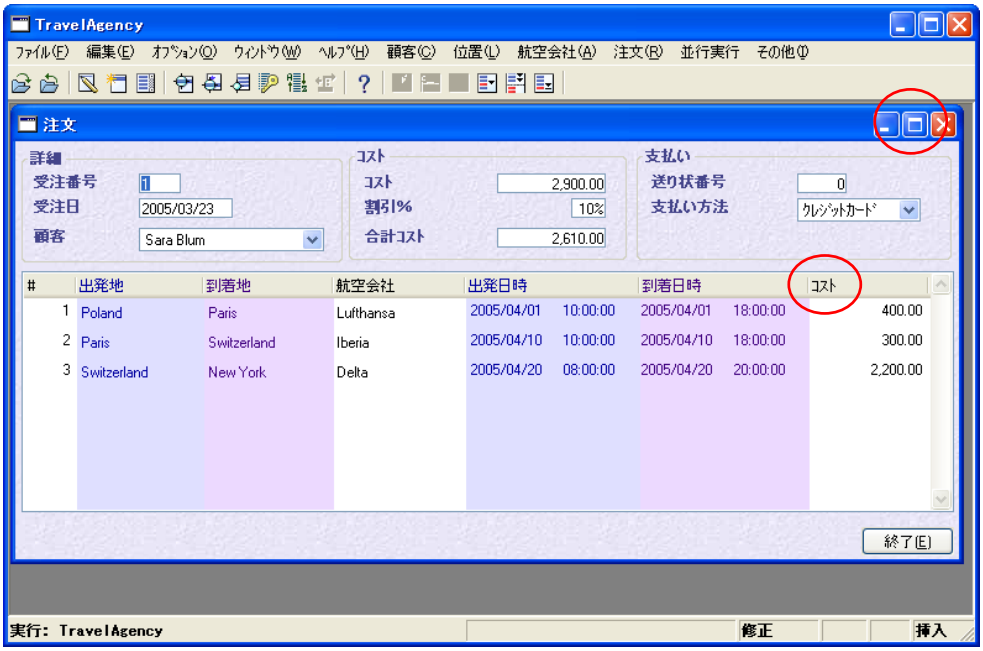


次のようにして、**注文** プログラムでフォーム状態が保存されることを確認してください。

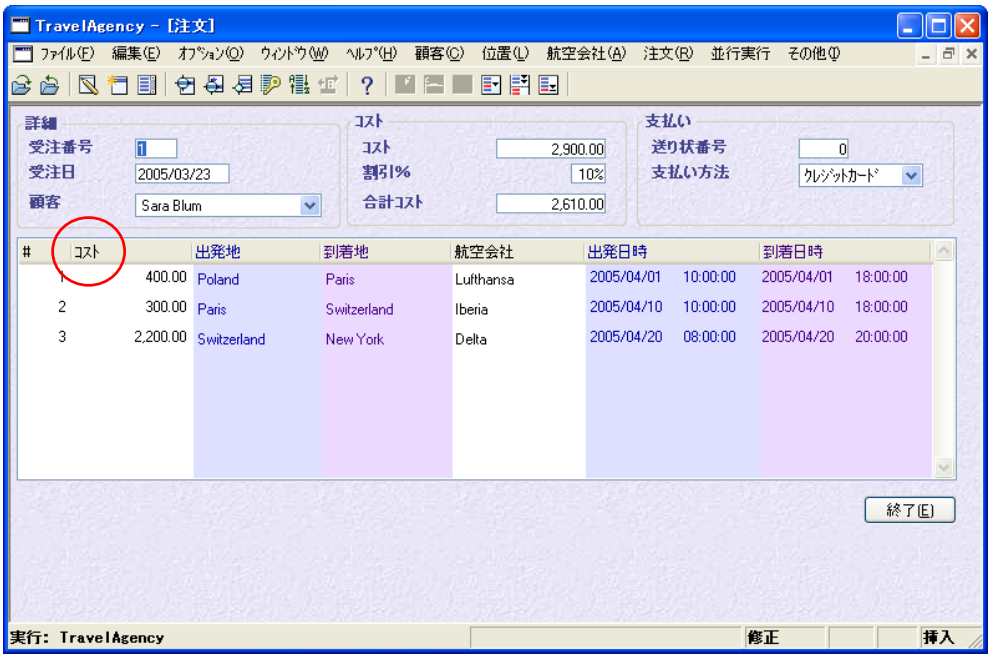


フォーム状態 ID を設定した効果を確認しましょう

1. キャビネットファイルを実行し、プルダウンメニュー 注文(R) を選んで、注文 プログラムを起動します。初期状態では右図のようになっています。



2. 内部のウィンドウを最大化し、さらに、明細行テーブルのタイトルをドラッグ&ドロップして、**コスト** カラムが **#** の次に来るようにします。

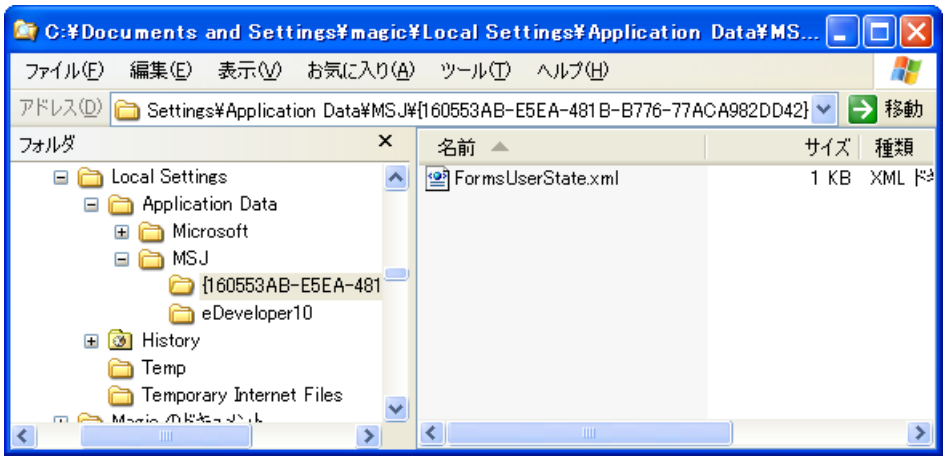


3. **終了(E)** ボタンを押して、**注文** プログラムをいったん終了します。
4. その後、再度 **注文** プログラムを起動します。ここで、3 で閉じたときと同じ状態、すなわち、内側のウィンドウは最大化され、明細行の **コスト** カラムは 2 番目に表示されていることを確認してください。

7.7.3. フォーム状態 ID 利用上の注意点

フォーム状態 ID を使う場合、次のような点に注意してください。

- **フォーム状態 ID** 特性には、アプリケーション内で一意な名前を設定してください。重複する名前があると、正しく復元されないことがあります。
- 先の **注文** プログラムで見たように、親子構造のタスクでは、親フォームと子フォームの両方にそれぞれ **フォーム状態 ID** を設定してください。
- フォーム状態の保存と復元の機能は、実行版で実行した場合にのみ有効です。スタジオから起動する場合には、**フォーム状態 ID** 特性が設定されていても保存・復元がされません。
- フォーム状態の情報は、C:¥Documents and Settings¥(ユーザ ID)¥Local Settings¥Application Data¥MSJ¥(プロジェクト ID) などのフォルダに、XML データの形式で格納されます。ここで、(ユーザ ID)というのは、Windows にログインしているユーザ名で、(プロジェクト ID)というのは、プロジェクトごとに一意に決められている内部 ID です（例：{160553AB-E5EA-481B-B776-77ACA982DD42} など)。



7.7.4. フォーム状態関連関数

フォーム状態は自動的に更新されますが、場合によってはこれをリセットして、初期状態に戻したい場合があります。このようなときのために、フォーム状態をリセットする関数が用意されています。

シンタックス　FormStateClear (フォーム状態 ID)

パラメータ　フォーム状態 ID(文字) … 初期化するフォームの[フォーム特性]の[フォーム状態 ID]特性に定義されている文字。

以下の予約語を使用することができます。

‘’(空白)	空白は、現在のフォームの状態を初期化します。
‘*’(アスタリスク)	アスタリスクは、現在のプロジェクトの全てのフォームの状態情報を初期化します。

戻り値　論理値。指定されたフォームに保存されている状態情報が存在する場合は、True が返ります。

8. 以前のバージョンから V10 への移行

V10 が持つ多様な先進機能を実現するため、V10 の開発においては Magic エンジンの根本的な再設計が行われました。同時に、過去のバージョンとの互換性維持にも努めた結果、V9 から V10 への移行はたいへん容易に行うことができるようになっています。

本章では、過去の Magic バージョンから V10 へ移行する場合の移行パス、注意点、移行手順などについて説明します。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

V10 へのマイグレーションの詳細	**マイグレーション** の章
RM 互換レベル	**ロジックエディタ → ロジックユニット → RM 互換**
作業ディレクトリ	**プロジェクトとアプリケーション → プロジェクト開発 → プロジェクトディレクトリの取得** **プロジェクトとアプリケーション → アプリケーション実行**
権利設定	**認証システム → 実行/APG 権利**
内部イベント(アクション)	**Magic エンジン → エンジン実行レベル → イベントレベル → イベント一覧**

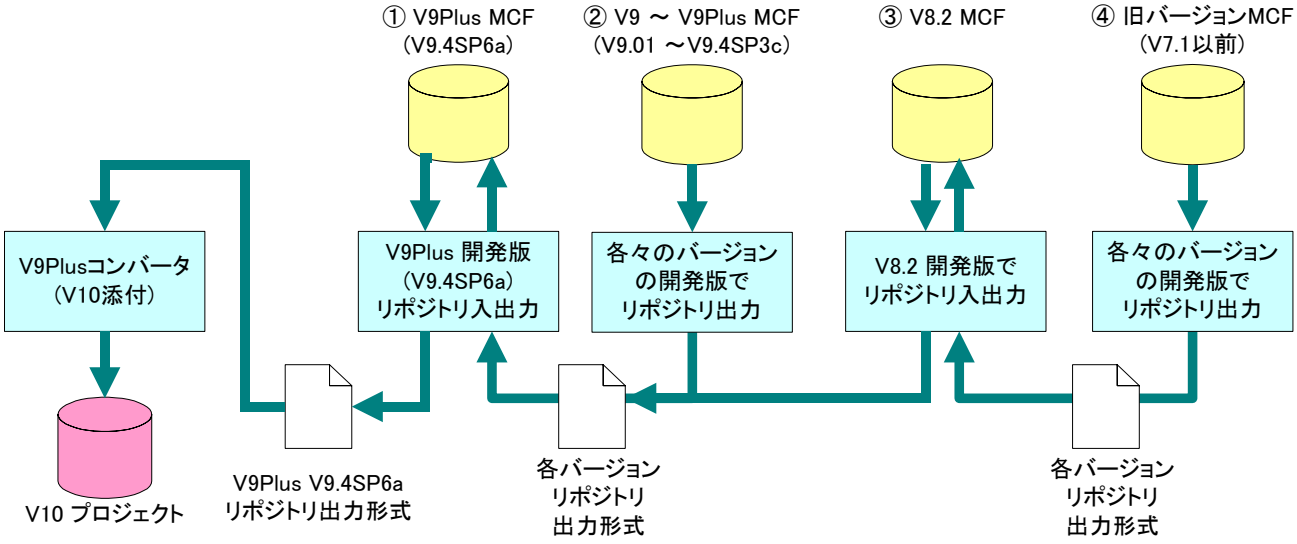
8.1. 移行パス

V10 のプロジェクトは XML をベースとするソース形式ファイルなので、V10 への移行は、過去のバージョンのリポジトリ出力形式ファイルを、V10 添付の V9Plus コンバータにより、V10 のプロジェクト形式に変換することにより行います。

8.1.1. バージョン別の移行パス

ここで注意が必要なのは、V9Plus コンバータは、Ver 9.4SP6a およびそれ以降のリポジトリ出力形式ファイルのみを入力とすることができます。このため、次に示すように、移行前の MCF のバージョンにより、移行手順が異なります。

V10への移行パス



ケース	移行前のバージョン	手順
上図 ①	V9.4SP6a およびそれ以降	1. V9Plus Ver 9.40J SP6a 以降で、アプリケーション全体をリポジトリ出力します。 2. その後、V10 製品に添付の V9Plus コンバータプログラムを用いて、V10 形式のプロジェクトに変換します。
上図 ② 又は ③	V8.2 ～ V 9.40J SP3c	いったん V9Plus Ver 9.4J SP6a 以降に移行した上で、上記手順を行ってください。
上図 ④	V4 ～ V7	まず V8.2 に移行し、更に V9Plus Ver 9.4J SP6a 以降に移行した上で、上記手順を行ってください。

8.1.2. 移行のための旧バージョン

上記のように、V9Plus Ver9.4SP6a より以前のバージョンから V10 にアプリケーションを移行するには、V9Plus Ver9.4SP6a、あるいは、V8.2 が必要となります。Magic Studio V10 製品には、開発者の移行の便のために、V9Plus Ver9.4SP6a および V8.2K4SP2 の簡易版が同梱されています。

注意：これらのバージョンは、移行目的のためだけに提供されますので、次のような制限があります。

- 基本的に、リポジトリ入力と出力を行うためにだけ利用します。
- 開発・実行機能も利用できますが、移行の確認と最小限必要な修正のためにだけ利用してください。アプリケーション単純移行の目的を超える、新しいアプリケーション開発や新しい機能の追加のために利用することは認められていません。
- Web 機能実行のためのモジュール類（MRB、リクエスタ、ブラウザクライアントモジュール等)のインストール・設定は行われません。
- 実運用環境で利用することは認められません。
- 使用権は、ご購入になった Magic Studio V10 製品に準ずるものとし、特定の PC1 台にのみインストール可能です。

8.2. サポートされなくなった機能

V10 への移行を容易にするため、V9Plus の開発版で、アプリケーションを見直しておきましょう。特に、V10 でサポートされなくなった以下のような機能については、V9Plus で予め削除しておくか、あるいは別の方法に変更しておくことをお勧めします。

8.2.1. サポートされなくなった関数

V10 でサポートされなくなった関数については、V9 よりの関数の変更(4.9.5 参照)を見てください。これらの関数があると、V9Plus コンバータが正しく動作しません。

8.2.2. Web Online

V10 では、Web Online の機能はサポートされなくなりました。V9Plus コンバータは、WebOnline 関係のフォームはすべて削除します。従って、このフォームを参照している データ出力 コマンドはシンタックスエラー となります。

8.2.3. HTML フォーム

V10 では、インターフェースタイプが HTML のフォームをサポートしなくなりました。V9Plus コンバータは、HTML フォームを削除し、HTML フォームを参照しているデータ出力コマンドでは、フォーム番号が 0 となります。

これに関連して、以下のような変更がなされています。

- 項目やモデルで、**スタイル** セクション中の **HTML 形式** 特性が削除されています。
- モデルリポジトリでは、**HTML 形式** のクラスが削除されています。
- アプリケーション特性では、インターネット開発ファイルルート 特性が削除されています。
- APG では、**インターネット** オプションが削除されています。
- **HTML マージ形式** は、単に **マージ形式** と改名しました。

8.2.4. エディット コマンド

エディット コマンドはサポートされなくなりました。
V9Plus コンバータは、**エディット** コマンドを **OS コマンド** タイプの **外部コール** コマンドに変更します。

8.2.5. 動作環境

動作環境テーブルに関しては、次のような変更があります。

- **マルチユーザ アクセス**（**マルチユーザ(M)** タブ）は削除されました。V10 では常に マルチユーザアクセス ＝ Yes の動作をします。
- Euro 関係の機能が削除されたことに伴い、**通貨変換ファイル**（**外部参照(E)** タブ）が削除されました。また、CTL 特性の **基準通貨** および **通貨変換ファイル** 特性が削除されました。
- Java のブラウザクライアント アプレットはサポートされなくなったので、**アプリケーション サーバ (N)** タブの **ブラウザクライアント方式** パラメータが削除されました。

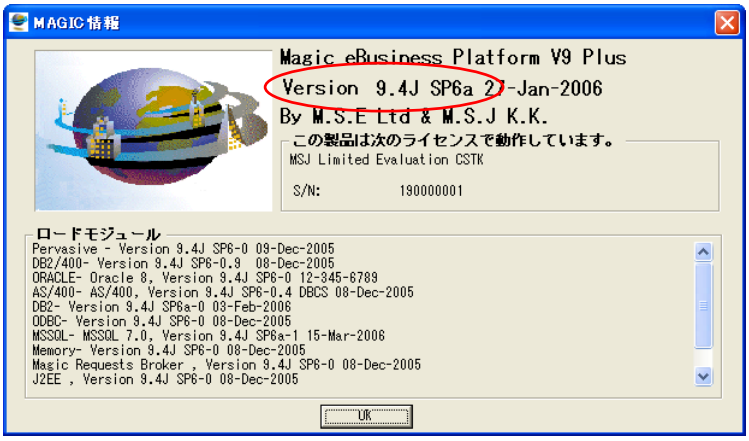
8.3. V9Plus でのリポジトリ出力

準備作業が終わったら、V9Plus でアプリケーションをリポジトリ出力します。

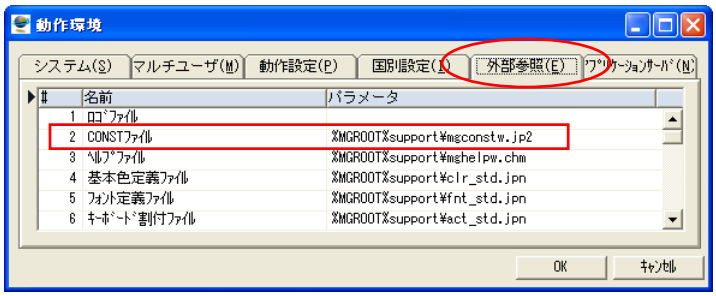
8.3.1. リポジトリ出力時の注意事項

重要：リポジトリ出力に際しては、以下の項目が必須となりますので、必ず確認してください。

- V9Plus のバージョンは、9.4S6a あるいはそれ以上であること。
  - それより以前のバージョンでリポジトリ出力をしたものは、V9Plus コンバータで受け付けません。
  - V9Plus のバージョンは、ヘルプ → Magic 情報 メニューで確認することができます。



- CONST ファイルとして、mgconstw.jp2 を使っていること。
  - 使用している CONST ファイルは、**設定 → 動作環境** メニューで **外部参照** タブを開くと確認できます。
  - CONST ファイルの設定を変更した後は、Magic をいったん終了して再起動してください。
  - これ以外の CONST ファイルを利用している場合には、アプリケーションで使っているアクション名が正しく変換されない場合があります。



8.3.2. リポジトリ出力ファイルの確認

リポジトリ出力ファイルがすでにある場合、上の注意事項に従ってリポジトリ出力がなされたかどうか確認したい場合には、次の点をチェックしましょう。

Magic バージョンの確認

リポジトリ出力をした Magic のバージョンは、リポジトリ出力ファイルの先頭に記録されます。右図にあるように、キーワード VRSN= の値が 940.06 以上であれば OK です。それ以下の場合には、V9Plus Ver 9.4SP6a 以降のバージョンで、リポジトリ入力・出力が必要です。

```
VRSN=940.06
APPLICATION=Y
MODEL=Y
TABLE=Y
PROGRAM=Y
MAIN_PROGRAM=Y
```

CONST ファイルの確認

リポジトリ出力時に使った CONST ファイルが何であったかは、アクション名を調べてチェックすることができます。リ

ポジトリ出力ファイルをテキストエディタなどで開き、「'ACT」 をキーワードとして文字列検索してみてください。アクションを使っているアプリケーションならば、いくつかヒットすると思います。例えば、以下のように使われているでしょう。

```
EXP="1",
EXP="KbGet (1)='終了'ACT OR KbGet (1)='ｷｬﾝｾﾙ(C)'ACT",
EXP="Flow ('C')",
```

ここで出力されるACTのリテラル名は、CONSTにより違っています。以下にいくつかの代表的なアクションを示します。

mgconstw.jp2 の場合（正しい）	mgconstw.jpn の場合（正しくない）	mgconst.eng の場合（正しくない）
'終了'ACT	'E:終了'ACT	'Exit'ACT
'ｷｬﾝｾﾙ(C)'ACT	'C:取消'ACT	'Cancel'ACT
'クローズ(C)'ACT	'C:クローズ'ACT	'Close'ACT
'次項目'ACT	'次項目'ACT	'Next Field'ACT
'ズーム(Z)'ACT	'ズーム'ACT	'Zoom'ACT

アクション名をチェックして、正しく mgconstw.jp2 で出力されているかを確認してください。

8.4. 移行ユーティリティの処理プロセス

前節の内容に従って、V9Plus でアプリケーションの修正を行った後、リポジトリ出力を行います。その後、V10 が提供する V9Plus コンバータにより、V10 が読み込めるプロジェクトファイルを作成します。

8.4.1. 移行処理の内容

ここでは、V9Plus コンバータが実行する処理を簡単に説明します。

項目	説明
タスクレイヤー	● メインテーブルとデータリンクの定義などは、データビューエディタに移動します。 ● タスクのロジック部分は、ロジックエディタに移動します。 ● タスクのフォームは、フォームエディタにまとめます。
DB テーブル	DB テーブルでの追加テーブルは、データビュー エディタの 宣言 コマンドに移動されます。
タスク/レコードレベルのハンドラ	V9Plus のレコード前処理、後処理、タスク前処理・後処理は、ロジック エディタの、それぞれ対応するハンドラに配置されます。
レコードメイン中のロジック	V9Plus でレコードメインで定義されていたロジックは、ロジックエディタの RM 互換 レベルに移動させられます。
コマンド	V9Plus のコマンドは V10 のコマンドと 1 対 1 に対応していません。コマンド体系の違いについては、コマンドの種類（4.3.3） を参照してください。
リンク照会 の **確認** オプション	**リンク照会** では、リンクが失敗した場合にエラーメッセージを出す **確認** オプションがありましたが、V10 ではこの機能はサポートされていません。V9Plus コンバータは、**確認** が Yes のリンク照会 コマンドを、**リンク照会** コマンドと **エラー** コマンドとに分けます。
コントロール変更ハンドラ	V9Plus でのコントロール変更は、V10 では項目 変更 ハンドラに置き換えられます。 もし同一のコントロールに対してコントロール変更ハンドラが定義されており、異なる条件付けがされている場合には、同等の動作を行うように変換します。 もし同一の項目が異なるフォームに配置されており、それぞれのコントロールにそれぞれのコントロール変更ハンドラが定義されていた場合には、MainDisplay() 関数(V10 での新関数)により条件付けがされます。 コントロールでの変更以外の変更によるハンドラの実行を防止するため、ハンドラの内容はブロックコマンドを使い、「**変更理由** パラメータ = 0」という条件付けがされます。
強制終了＝レコード のイベント	ユーザ定義イベントで、強制終了 パラメータが レコード となっていたイベントは、レコード更新前に変更されます。強制終了 パラメータについては、8.4.2 ユーザ定義イベントの強制終了パラメータ を参照してください。
パーキング設定	オンラインプログラムで、あるコントロールのカーソルのパークについての設定は、V9Plus ではセレクト コマンドの **条件** 式や**フロー**設定により決定されていました。 V10 では、セレクトコマンドの条件、というものがなくなったので、コントロールの特性としてパーキング関係の設定を行うようになりました。 パーキングについて詳しくは、8.4.3 パーキング関係の設定 および 、7.1TAB 順序 を参照してく

	ださい。
関数名の変更	V10 ではいくつかの組み込み関数で名前の変更がありました。(4.9.5 V9 よりの関数の変更 を参照。) V9Plus コンバータでは、これらの関数が式中で使われている場合、自動的に名前を変更します。
リテラル	V10 では、以下のリテラルの記号が変更になりました。 ACT → EVENT KEY → INDEX FILE → DSOURCE V9Plus コンバータは、これらのリテラル記号を変更します。
特性の削除	V10 では EURO 関係の機能はサポートされないので、EURO 関係の特性はすべて削除されます。V9Plus コンバータは、EURO 関係のパラメータを削除した場合、その旨エラーメッセージをログに記録します。
メモ型データ	メモ型データはサポートされなくなりました。メモ型は、文字型に変換され、データの互換性を維持するため、記憶形式は String Memo あるいは Magic Memo に設定されます。
コール Web サービスコマンド	Web サービスのフレームワークが V10 では完全に変わるため、プログラム上は V9Plus と互換性がありません。V9Plus で コール Web サービス が使われている場合、V9Plus コンバータは、空の 外部コール Web サービス コマンドに変換します。開発者は、V10 に移行後、Web サービス呼び出しを再定義しなくてはなりません。
Unicode	V10 は Unicode をサポートするようになりました。V9Plus の以下のオブジェクトは、自動的に Unicode に変換されます。 ● Unicode 文字を格納するカラムのデータ型(文字型から Unicode 型に変換) ● メニューのエントリ テキスト ● ヘルプ・ツールチップの文字列 ● 値として文字型の式が設定されているコントロールのデータ型(文字型から Unicode 型に変換) ● プロパティシートに設定される種々の文字列

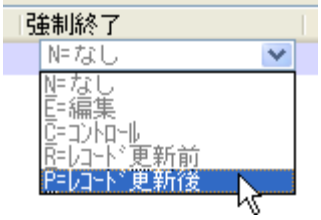
8.4.2. ユーザ定義イベントの強制終了パラメータ

V9Plus では、ユーザ定義イベントの **強制終了** パラメータには、

- N=なし
- E=編集
- C=コントロール
- R=レコード

という 4 つの選択肢がありました。

V10 では、**R=レコード** の動作をより正確に表現するため、**R=レコード更新前** と改名され、また、あらたなオプションとして、**P=レコード更新後** が追加されました。



次表は、**強制終了パラメータ** として **R=レコード更新前** あるいは **P=レコード更新後** となっている場合に、実行エンジンの実行順序を示したものです。

#	R=レコード更新前 に設定されている場合	P=レコード更新後 の場合
1.	コントロール検証	コントロール検証
2.	コントロール後処理	コントロール後処理
3.	レコード後処理	レコード後処理
4.	**[ユーザイベントのハンドラの処理]**	
5.	レコード更新	レコード更新
6.	レコード前処理（次のレコードサイクルの開始）	レコード前処理（次のレコードサイクルの開始）
7.		**[ユーザイベントのハンドラの処理]**
8.	コントロール前処理	コントロール前処理

これからわかるように、**R=レコード更新前**（V9Plus での**レコード**）の場合には、ユーザイベントのハンドラが実行されるタイミングでは、まだレコード更新処理が行われておらず、現在のレコードの変更内容はまだ DBMS に書き込まれていません。
一方、P=レコード更新後の場合には、ユーザイベントのハンドラはレコード更新の後で実行されるので、DBMSに現在のレコードの変更内容が書き出された後となります。

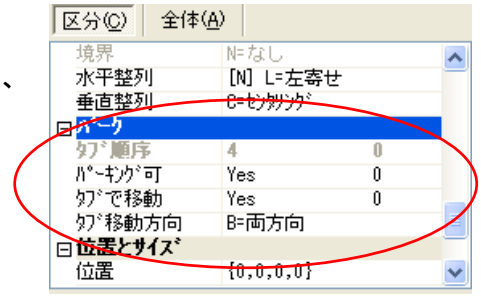
8.4.3. パーキング関係の設定

オンラインプログラムで、あるコントロールのカーソルのパークについての設定は、V9Plus では セレクト コマンドの **条件** 式や**フロー**設定により決定されていました。下図は、V9Plus でのレコードメインの画面です。

処理テーブル: レコード メイン

#	処理コマンド			内容			範囲			位置付		フロー		条件
1	ｾﾚｸﾄ	R=実ﾃﾞｰ:	1	受注番号	代入:	0	0	0		0	0	S	C	No
2	ｾﾚｸﾄ	R=実ﾃﾞｰ:	4	受注日	代入:	10	0	0		0	0	S	C	Yes
3	ｾﾚｸﾄ	R=実ﾃﾞｰ:	2	顧客番号	代入:	0	0	0		0	0	S	C	Yes
4	ﾘﾝｸ	Q=照会 :	2	顧客マスタ	ｲﾝﾃﾞｯｸ	1	順:	A=昇順		戻:	???			Yes
5	ｾﾚｸﾄ	R=実ﾃﾞｰ:	1	顧客番号	代入:	0	0	0		1	1	S	C	No
6	ｾﾚｸﾄ	R=実ﾃﾞｰ:	2	顧客名	代入:	0	0	0		0	0	S	C	No
7	ｾﾚｸﾄ	R=実ﾃﾞｰ:	3	住所	代入:	0	0	0		0	0	S	C	No

V10 では、セレクトコマンドの条件、というものがなくなったので、コントロールの特性としてパーキング関係の設定を行うようになりました。
以下のパーキング関連のパラメータが、V9Plus コンバータにより、V10 の対応するコントロール特性に設定されます。これらの詳細については、7.1TAB 順序 を参照してください。

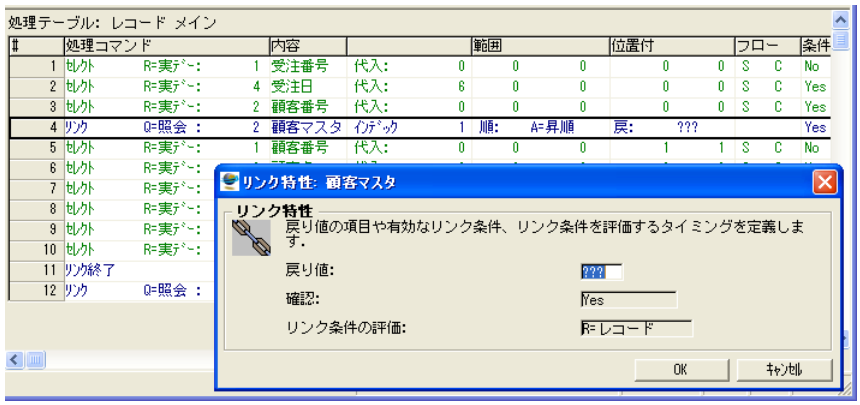


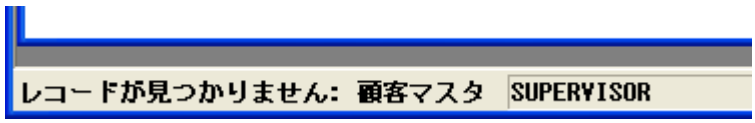
項目	V9Plus	V10
パークの可・不可	セレクトコマンドの **条件** で制御	コントロール特性の **パーキング可** により制御
TAB 移動方向によるパークの可・不可	セレクト コマンドの **フロー** パラメータで設定	コントロール特性の **TAB 移動方向** パラメータにより設定
TAB 順序	レコードメインでのセレクトコマンドの定義順	コントロール特性の **TAB 順序** パラメータにより制御。 モードとして、自動 TAB 順序 と 手動 TAB 順序 とがあります。

8.4.4. 照会リンクの移行

V9Plus では、**Q=照会 リンク**コマンドで **確認** 特性が Yes の場合、リンクが成功するかどうかがチェックされ、リンクに失敗した場合には、「レコードが見つかりません: テーブル名」というエラーメッセージが出て、フローはそこで中断されました。(右図)
これは簡便な機能ではありましたが、エラーメッセージが固定されて、結局はリンク結果を格納する論理型変数とエラーコマンドとを使って、プログラムで対応する、ということが多くありました。

V9Plus での、照会リンク（確認 特性）

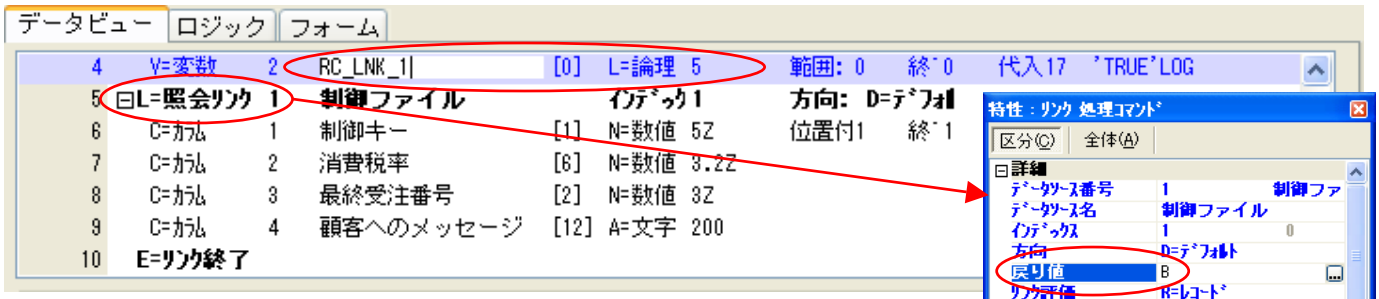




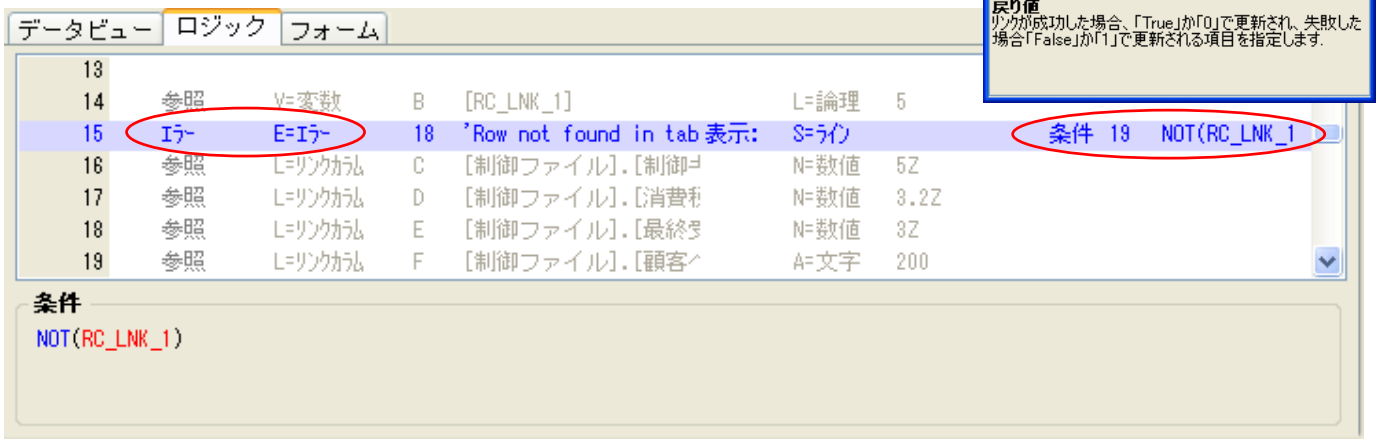
このようなことから、V10 では、リンクコマンドの **確認** 特性を廃止しました。V9Plus からの移行時に同じ動作をさせるために、V9Plus コンバータでは、照会リンクコマンドを次のような形で変換します。

1. リンク結果を格納する論理型変数を追加します。（以下の説明では、RC_LNK_1 という名前で呼びます）
2. リンクコマンドでは、戻り値 特性で、変数 RC_LNK_1 を設定します。
3. ロジックエディタの RM 互換 レベルで、RC_LNK_1 の直後にエラーコマンドを追加します。条件として、NOT(RC_LNK_1) とします。

データビューエディタ



ロジックエディタ



V9Plus コンバータでは、V9Plus での動きとなるべく同じにするために、RM 互換レベルにエラーコマンドを入れていますが、新しくタスクを定義する場合には、RM 互換レベルではなく、コントロールレベルのハンドラ（コントロール後処理、あるいはコントロール検証ハンドラ)にチェックロジックを入れるのが適当です。

このやりかたでは、データビューに論理型の変数が、リンクコマンドの個数分だけ新たに追加されます。このため、以下のような条件の場合、このプログラムにパラメータが正しく渡らなくなってしまうことがあります。

- パラメータを、**P=パラメータ** ではなく、**V=変数**で渡している。(V8.2 およびそれ以前のプログラムではこのようにしているので、V8.2 などから移行するアプリケーションでは要注意です)
- パラメータとして使っている V=変数 コマンドが、照会リンクコマンドよりも後ろに定義されている。

このふたつの条件がそろった場合には、パラメータとして使っている **V=変数** コマンドよりも前に、リンク結果を格納する論理型変数が定義されることになるので、パラメータの順序が変わってしまいます。その結果、パラメータが正しく渡らなくなることになります。

この問題の解決方法は、パラメータを **P=パラメータ** で定義しなおすことです。こうすれば、リンク結果を格納する論理型変数がどこに定義されていても、正しくパラメータと変数が区別されるので、順序が変わってしまうことはありません。

8.5. V9Plus コンバータのパラメータ

V9Plus コンバータは、コマンドラインのプログラムで、v9converter.exe という名前で Magic をインストールしたディレクトリにあります。
V9Plus コンバータは、以下のようなパラメータを受け付けます。

パラメータ	必須？	意味
-Export	必須	変換元である V9Plus のリポジトリ出力ファイル名を指定します。
-Project	必須	変換結果である V10 のプロジェクトファイルを格納するディレクトリ名をフルパス名で指定します。もしこの位置にすでに V10 プロジェクトファイルが存在していたら、変換処理は実行されず、警告メッセージが表示されます。
-LANG	必須	言語パラメータを設定します。日本語の場合には、-LANG=JPN と指定してください。
-LOCALS	必須	国・地域に固有な、パラメータを指定します。日本語の場合には、 -LOCALS=J,./: と指定してください。 文字の意味は、左から、日付モード(J)、数値の 3 桁区切り文字(,)、小数点文字(.)、日付セパレータ(/)、時刻セパレータ(:) です。
-LOG	任意	変換プロセスのログファイル名を指定します。
-PREFIX	任意	V9Plus のアプリケーションの識別子（2 文字）を指定します。テーブルリポジトリで、テーブル名が指定されていないテーブルのデフォルト名を決めるために使います。

下記は、V9Plus コンバータを利用する例です。(実際には行を分けず、1 行で記述します)

```
"C:¥Program Files¥Magic¥Studio 10.1¥Projects¥v9converter.exe"
       -LANG=JPN  -LOCALS=J,./:
       -EXPORT=D:¥Magic¥Project¥Petshop¥PetShop.exp
       -PROJECT=D:¥Magic¥Project¥Petshop -PREFIX=DM
```

注意：-LANG および -LOCALS パラメータは、日本語環境では必須です。

-LANG および -LOCALS パラメータをつけなくとも、変換プロセスは一見正常終了するように見えます。しかし、CONST 設定、日付形式の設定などが、日本語環境での設定と異なった状態で変換されるため、日付定数やイベントなどが正しく変換されません。

8.6. ログファイル

V9Plus コンバータは、変換処理の過程での種々の報告・警告・エラーメッセージを出力します。これらはすべてコマンドライン画面上にも表示されますが、ログファイルにも記録されます。
ログファイルは、v9converter.exe へのコマンドラインパラメータ –LOG= で指定します。このパラメータを指定しない場合には、デフォルトとして、作業ディレクトリの下に、（プロジェクト名）_cnv.log という名前で格納されます。

以下は、V9Plus のペットショップデモを変換したときのログの例です。

```
-+> Migrating table '制御ファイル'.
     Attribute of field '顧客へのメッセージ' changed from 'Memo' to 'Alpha'
-+> Migrating table '顧客ファイル'.
     Attribute of field 'メモ' changed from 'Memo' to 'Alpha'
-+> Migrating table '商品ファイル'.
-+> Migrating table '受注ファイル'.
-+> Migrating table '受注明細ファイル'.
Processing task 'メインプログラム'.
Task 'メインプログラム' migrated successfully.
Processing task ''.
Task '' migrated successfully.
-+> Migrating table '制御ファイル'.
     Attribute of field '顧客へのメッセージ' changed from 'Memo' to 'Alpha'
-+> Migrating table '顧客ファイル'.
     Attribute of field 'メモ' changed from 'Memo' to 'Alpha'
-+> Migrating table '商品ファイル'.
-+> Migrating table '受注ファイル'.
-+> Migrating table '受注明細ファイル'.
Processing task 'メインプログラム'.
Task 'メインプログラム' migrated successfully.
Processing task ''.
Task '' migrated successfully.
Processing task '制御ファイル更新 ONL'.
Task '制御ファイル更新 ONL' migrated successfully.

   （途中省略）

Processing task '照会 - 受注明細ファイル'.
Task '照会 - 受注明細ファイル' migrated successfully.
----> Internet development file root has been removed in this version.
----> Base Currency support has been removed in this version.

Saving application, please wait.
Application saved.

Conversion completed successfully.

Note:
1. Following functions have been renamed:
   MINMagic() => WINMinimize()
   MAXMagic() => WINMaximize()
   ResMagic() => WinRestore()
   Asc()      => ASCIIVal()
   CHR()      => ASCIIChr()
   CtrlName() => LastClicked()
   Sys()      => AppName()
   IOExist()  => FileExist()
   IOSize()   => FileSize()
   IODel()    => FileDelete()
```

```
   IORen()    => FileRename()
   IOCopy()   => FileCopy()

2. The 'Check object list'ACT & 'Save As MFF'ACT actions are deprecated.

3. The 'HTML' interface type for forms is deprecated.

4. Inheritance of the following GUI display table control properties is broken:
   'Style', 'Table in Window', 'Row Highlight style', 'Border Style',
   'Bottom Position Interval', 'Line Divider' and 'Column Divider'.

5. Inheritance of the all 'Style' properties of a blob field is broken.
```

8.7. GUI エンハンサ

8.7.1. GUI エンハンサとは？

V9Plus コンバータを使って V10 のプロジェクトを作れば、V10 で動くアプリケーションとなります。ここで、XP テーマが有効になっていれば、V9Plus のアプリケーションが、自動的に XP テーマ対応のユーザインターフェースとなります。(右図)

このユーザインターフェースでは、確かに、エディットコントロールやボタンコントロールが XP テーマ対応になっているものの、テーブルコントロールなどは V9Plus のままです。V10 の GUI 機能を生かしきっているとはいえません。

V10に移行直後の画面

V9Plus から移行してきたアプリケーションが、簡単に V10 の GUI の新機能を利用できるように、アプリケーション全体の GUI 設定を一括して V10 に最適なものに変更する **GUI エンハンサ** ウィザードが用意されています。

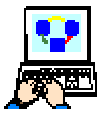
このユーティリティは、まず、オンラインフォーム上のコントロールを、すべて Windows スタイルにします。そして、ユーザの指定により、以下の設定をさせることができます。

- テーブルの **交互表示色** 特性
- テーブルの **区切り** 特性
- テーブルの **カラムの区切り線** 特性
- 平面コントロールを V10 のスタイルに変更

右図は、上のプログラムに GUI エンハンサを適用後、一部手作業で手直しをした画面です。テーブルが Windows スタイルになっているのがわかります。

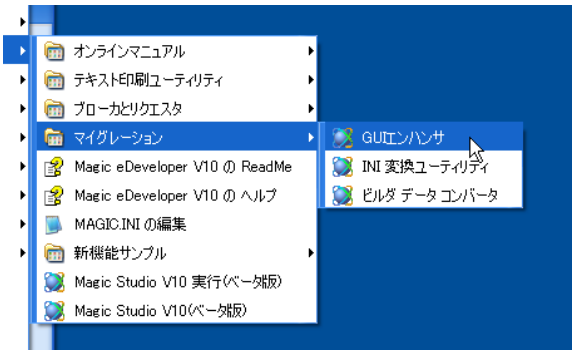
GUI エンハンサ適用後、一部手作業で修正した画面

8.7.2. GUI エンハンサの利用方法

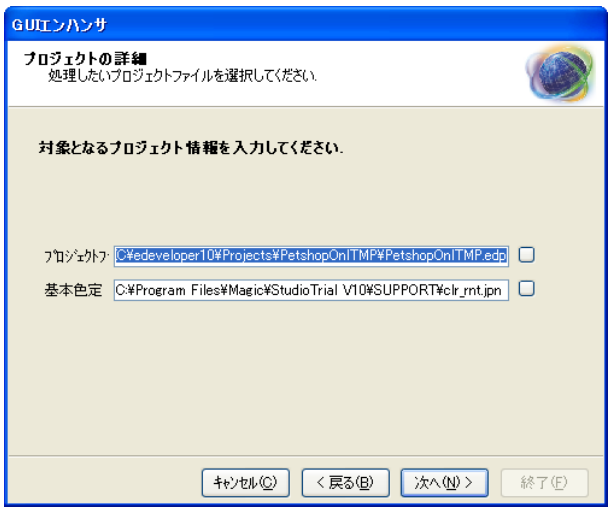


GUI エンハンサを利用するには・・・

1. スタートメニューから
   Magic Studio V10
   → マイグレーション
   → GUI エンハンサ
   を選びます。

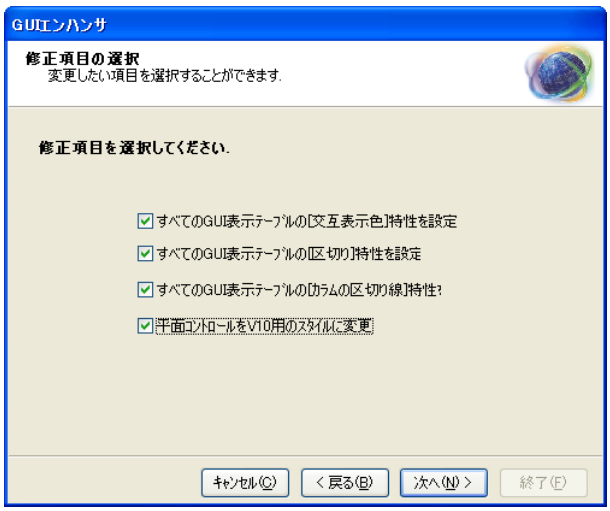


2. 変換を行いたいプロジェクトのプロジェクトファイルを指定します。
   このとき同時に、実行時の基本色定義ファイルも指定します。デフォルトでは、Magic V10 に添付されている、実行時基本色定義ファイルとなります。
   指定したら、**次へ(N)** ボタンを押します。

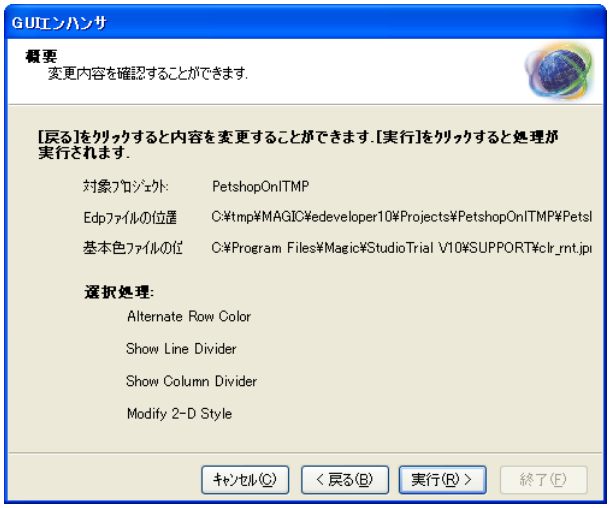


3. 修正項目を選択することができます。選択肢としては、以下のような項目をオン/オフできます。
   - すべての GUI 表示テーブルの[交互表示色]特性を設定
   - すべての GUI 表示テーブルの[区切り]特性を設定
   - すべての GUI 表示テーブルの[カラムの区切り線]特性を設定
   - 平面コントロールを V10 用のスタイルに変更

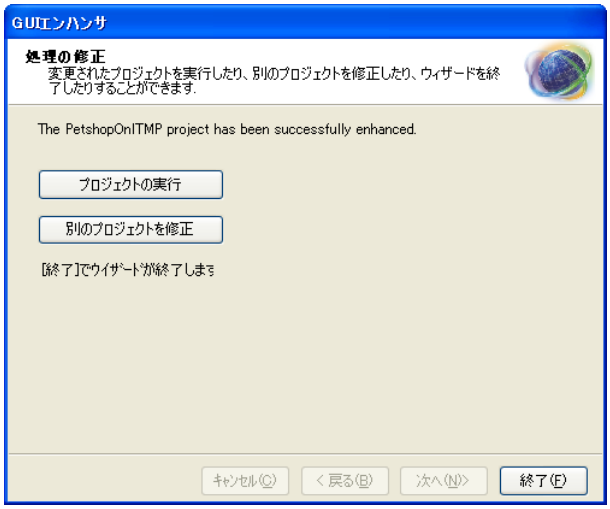
   指定したら、**次へ(N)** ボタンを押します。



4. 設定内容を確認するダイアログが表示されます。表示内容を確認し、正しければ **次へ(N)** ボタンを押します。処理が開始されます。
   まちがっているものがあれば、**戻る(B)** ボタンを押して、設定しなおします。



5. 処理が終了すれば、最終画面が表示されます。
   - ここで、今変換したプロジェクトを Magic スタジオで開きたい場合には、**プロジェクトの実行** ボタンを押します。
   - 別のプロジェクトを変換処理したい場合には、**別のプロジェクトを修正** ボタンを押して、2 から続けます。
   - 何もせずに終了する場合には、**終了ボタン(F)** を押します。



8.8. INI 変換ユーティリティ

8.8.1. INI 変換ユーティリティとは？

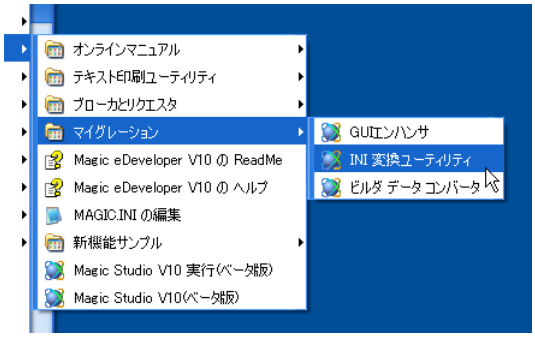
MAGIC.INI ファイルには、論理名、データベース定義、プリンタ定義、その他のアプリケーションの実行環境が数多く定義されています。アプリケーションをマイグレーションする場合、アプリケーション自体だけでなく、MAGIC.INI に定義されている環境設定も正しく移行しなければなりません。しかし、V9Plus と V10 とでは、MAGIC.INI の定義形式が異なっているため、V9Plus で使っていた MAGIC.INI をそのまま V10 で使うことはできません。
ひとつひとつを手作業で定義しなおす手間を省くために、Magic スタジオ V10 では V9Plus 形式の INI ファイルを V10 形式に変換するユーティリティを提供しています。これが **INI 変換ユーティリティ**です。

INI 変換ユーティリティは、V9Plus の INI ファイルを読み込み、V10 の標準 INI ファイルとマージするような形で処理を行います。このため、ユーティリティには次のようなパラメータを入力します。

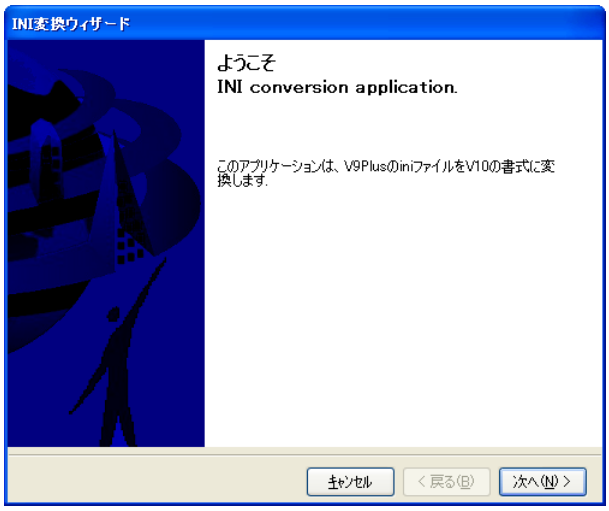
パラメータ	方向	（デフォルトで Magic をインストールした時の値）
V9Plus のインストールディレクトリ	入力	C:¥Program Files¥Magic¥eDeveloperPlus
V9Plus 形式の MAGIC.INI ファイル	入力	C:¥Program Files¥Magic¥eDeveloperPlus¥ Magic.ini
V10 形式の標準 MAGIC.INI ファイル	入力	C:¥Program Files¥Magic¥Studio V10¥Magic.ini
新規 Magic.INI ファイル	出力	C:¥Program Files¥Magic¥Studio V10¥MagicNew.ini

8.8.2. INI 変換ユーティリティの使い方

1. INI 変換ユーティリティは、
   スタート メニュー
   → Magic Studio V10
   → マイグレーション
   → INI 変換ユーティリティ
   で起動します。

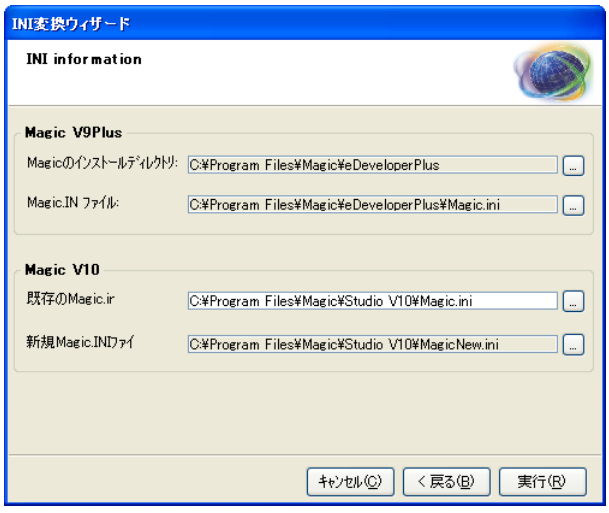


2. 最初の **ようこそ** 画面が出ます。
   次へ(N) を押します。

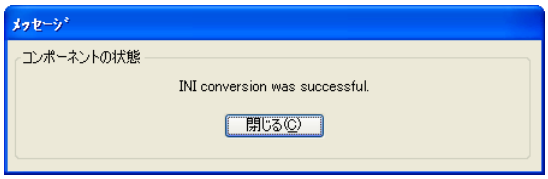


3. パラメータを入力します。
   - V9Plus のインストールディレクトリ
   - V9Plus 形式の MAGIC.INI ファイル
   - V10 形式の標準 MAGIC.INI ファイル
   - 新規 Magic.INI ファイル

   入力し終わったら、**実行(R)** を押します。



4. 正常に終わったら、右のようなメッセージが出ます。

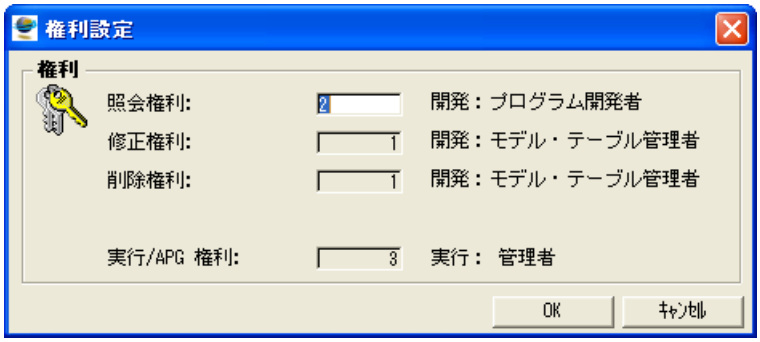


8.9. 権利設定とセキュリティファイル

8.9.1. 権利設定の変更

V9Plus での権利設定

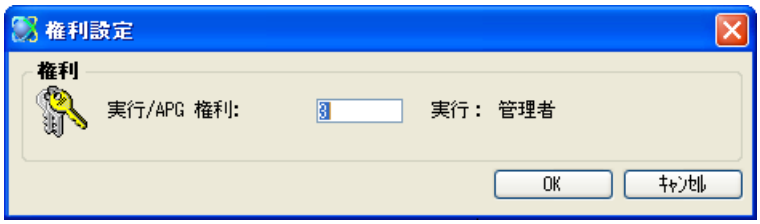
V9Plus までは、Magic のセキュリティ機能を使って、開発時あるいは実行時に有効な権利を、リポジトリ単位、プログラム単位で設定することができました。権利の設定は権利設定ダイアログ(右図)で行い、ここでは次のような権利を設定することができました。



権利設定内容	開発時/実行時？
照会権利	開発時権利
修正権利	
削除権利	
登録権利	
実行/APG 権利	テーブルリポジトリでは開発時権利。 プログラムリポジトリでは実行時権利。

V10 での権利設定

V10 では、実行時の権利設定は V9Plus と同じですが、開発時にはセキュリティをかけることができなくなりました。V10 の開発はプロジェクトベースの XML ソースファイルを使って開発するのですが、XML ソースファイルは通常のテキストファイルであるため、特定の開発者には開示し、別の開発者には隠蔽する、ということができないからです。このため、V9Plus から移行する場合、次のようになります。



- 権利設定ダイアログには、実行/APG 権利だけが残ります。その他の権利は削除されます。
- 権利設定ダイアログは、プログラムリポジトリでだけ開くことができます。

実行／APG 権利では、プログラムの実行または実行不可という基本的な設定しかできません。これ以上の細かなセキュリティ制御を行いたい場合には、Rights 関数と User 関数を使ってプログラムで制御するようにします。

8.9.2. セキュリティファイル

V10 ではセキュリティファイル（SUPPORT¥usr_std.jpn など）の内部構造が変わりましたので、V9Plus で使っていたセキュリティファイルの内容を継続して使用したい場合は、Magic スタジオ V10 で提供されているユーティリティ usrupd.exe を利用して、セキュリティファイルを変換する必要があります。

usrupd.exe は V10 のインストールディレクトリにあるコマンドラインツールです。使い方は、コマンドプロンプトを開き、V10 のインストールディレクトリに移動してから、次の形式で実行します。

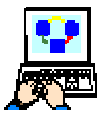
```
usrupd  <V9Plus セキュリティファイル名>  <V10 セキュリティ出力ファイル名>
```

次は、実行例です。(usrupd コマンドのパラメータは、1 行で書きます)

```
C:¥>cd "¥Program Files¥Magic¥Studio V10"
C:¥Program Files¥Magic¥Studio V10>usrupd "c:¥Program Files¥Magic¥eDeveloper
  Plus¥Support¥usr_std.jpn" support¥usr_v9plus.jpn
```

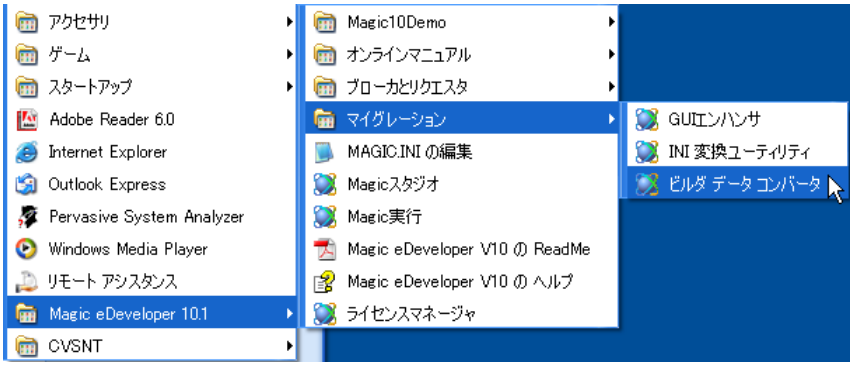
8.10. ビルダデータコンバータ

V9Plus には、Magic, COM、Web サービス、EJB などのビルダ ユーティリティがありました。
V10 でもこれらのユーティリティは提供されていますが、設定データ（Magic コンポーネントの設定など)の形式が異なるため、そのままでは流用することができません。
V10 のビルダ ユーティリティを使う際に、設定を最初からやりなおす手間を避けるため、V9Plus のビルダ ユーティリティの内容を V10 のユーティリティのフォーマットに変換する **ビルダ データ コンバータ** が提供されています。

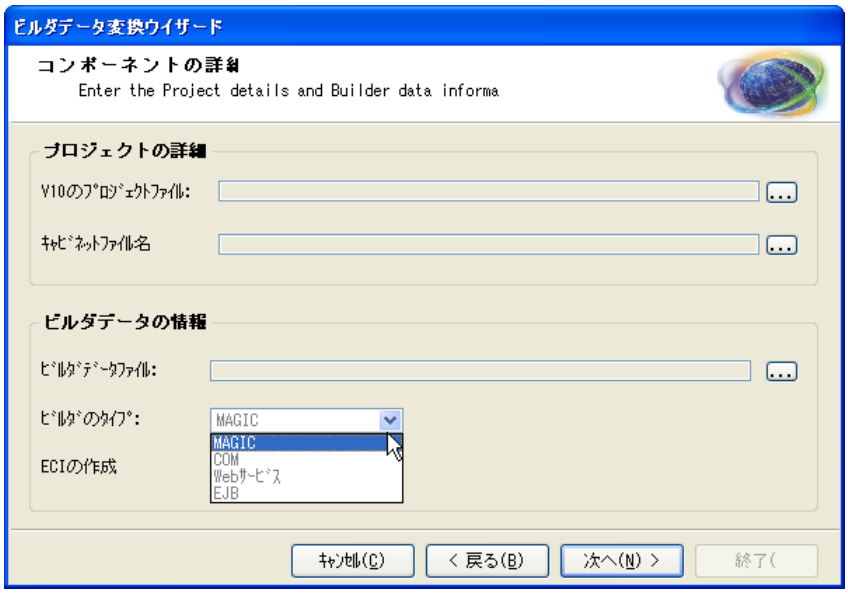


ビルダ データ コンバータでデータコンバートするには・・・

1. スタートメニューから起動します。



2. ウィザードが起動されるので、ウィザードの指示の従って処理してください。



8.11. 作業ディレクトリ

8.11.1. 作業ディレクトリの変更

V10 では、実行時の作業ディレクトリが V9Plus と異なりますので、注意が必要です。

V9Plus では、アプリケーション実行時の作業ディレクトリは、実行エンジン mgrntw.exe を起動したときの作業ディレクトリでした。起動時作業ディレクトリはショートカットのプロパティなどで設定することができます。
V10 では、プロジェクトファイル（.edp ファイル）、あるいはキャビネットファイル(.ecf ファイル)のあるフォルダが作業フォルダとなります。(プロジェクト/キャビネットファイルオープン時に変更されます)。
この変更のため、相対パスを指定して IO ファイルや Pervasive テーブルをオープンする場合には、異なる場所を参照するようになる可能性があるので注意が必要です。一般的に言って、ファイル名を指定する場合、プログラムの中では論理名を使い、論理名テーブル上では絶対パスで定義するのが一番トラブルが少ないといえます。
同様のことが、MAGIC.INI ファイル中の他の設定についても言えます。

8.11.2. 既定義の論理名

V10 では、Magic をインストールしたディレクトリと、現在の作業ディレクトリを示す論理名がエンジンで定義されています。

論理名	対応する実行名
EngineDir	Magic の実行エンジン eDevRnt.exe が存在するディレクトリ名。
WorkingDir	作業ディレクトリ名。

これらの論理名は、Magic のエンジンで自動的に定義されるので、論理名テーブルで定義する必要はありません。論理名テーブルで定義したとしても無視されます。

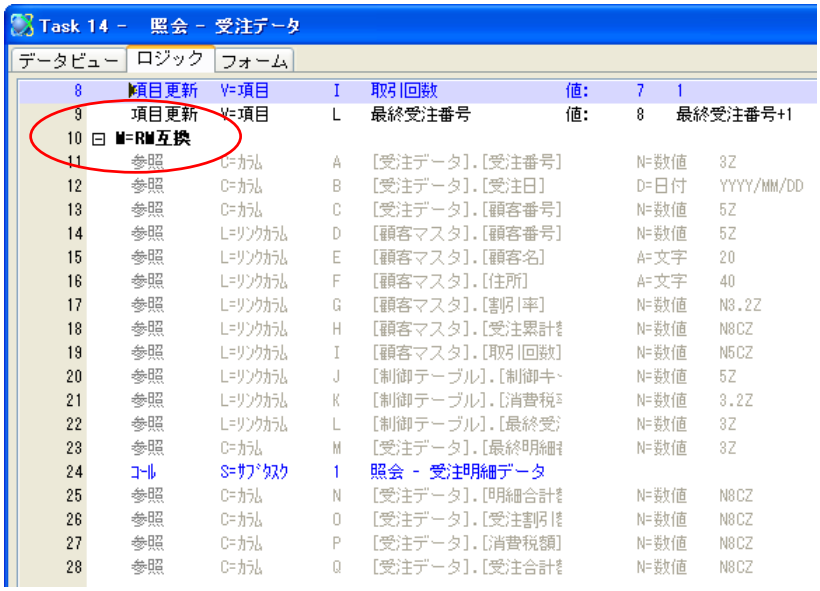
8.11.3. コンポーネントの場合

複数のコンポーネントからなるアプリケーションの場合、作業ディレクトリは一番最初にオープンされたキャビネットファイルの存在するディレクトリとなります。これは、コンポーネント中のテーブルやプログラムが呼び出され実行されている場合にも変わることはありません。
このため、コンポーネントとして使われている場合には、「キャビネットファイルの存在するディレクトリが、作業ディレクトリになる」という仮定は成り立たなくなります。キャビネットファイルを基準として相対パスでファイル名を指定している場合には、単体のテストでは正しく動くが、コンポーネントとした場合に間違った場所を参照する、ということになってしまいますので、注意してください。

8.12. RM 互換ロジックユニット

V9Plus のレコードメインには、データビューの定義とともにロジックを設定することができましたが、V10 ではレコードメインがなくなり、データビューの定義はデータビューエディタで、ロジックの記述はロジックエディタで行うようになりました。

V9Plus でレコードメインにロジックを記述していた場合には、V9Plus コンバータは、**レコードメイン互換（RM 互換)**という特殊なハンドラをロジックエディタに作成します。



8.12.1. RM 互換ハンドラの作成

V9Plus よりの移行

V9Plus コンバータは、V9Plus のアプリケーションを変換中に、レコードメインにロジックの書かれたプログラムがあると、V10 で RM 互換ハンドラを生成します。レコードメインにロジックのないプログラムには、RM 互換ハンドラを作成しません。

バッチタスクやブラウザクライアントタスクには、もともとレコードメインにロジックを書くことができないので、V10 に変換する際にも RM 互換ハンドラが作成されることはありません。

セレクトコマンドの条件式

V9Plus で、セレクトコマンドに設定されている **条件** は、カーソルのパーキングを制御しますが、V10 に変換されたプログラムでは、対応するコントロールの **パーキング** 関係の特性に反映されます。

パーキングについての詳細は、8.4.3 パーキング関係の設定 を参照してください。

8.12.2. 参照 コマンド

V9Plus では、レコードメインでデータ項目の定義にセレクトコマンドを使い、ロジックをその間に記述することができました。

V10 での RM 互換ハンドラでは、**セレクト** コマンドの変わりに、データビュー エディタで定義された項目を参照する **参照** コマンドが使われます。

V9Plus でレコードメインに定義してあったロジック（コールコマンド、エラーコマンド、項目更新コマンドなど)は、V10 の RM 互換ハンドラでは、参照コマンドの間に挿入されます。

8.12.3. データビューとの同期

参照 コマンドは、データビューでの定義と自動的に同期し、開発者が **参照** コマンドを追加・削除・修正することはできません。

また、次のような性質があります。

- データビューでの項目定義に従って自動的に作成・削除されます。
- 定義内容(データ型、書式、名前など)は、表示専用です。
- 順序もデータビューでの順序に従ったものです。
- データビューでの項目定義が追加・削除・変更、あるいは順序の変更がされると、**参照** もそれに従って自動的に追加・削除・変更されます。

8.12.4. TAB 順序

RM 互換 ハンドラがあるタスクでは、V9Plus での TAB 順序と同じ動作をさせるため、TAB 順序については、次のようになります。

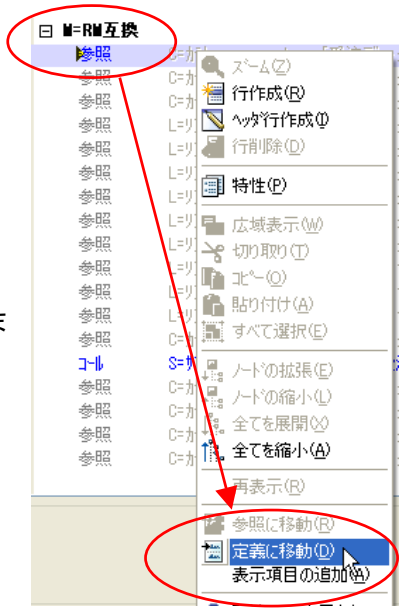
- 自動 TAB 順序は無効化されます。
- コントロールの **TAB 順序** 特性は無効化されます。
- TAB 順序は、項目がデータビューで定義されている順序になります。

8.12.5. 行間の移動

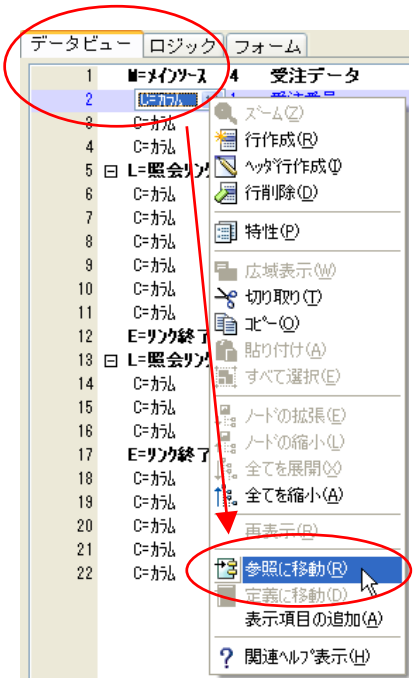
データビュー エディタと、ロジック エディタの RM 互換レベルの相互参照関係は、マウスクリックで簡単に移動することができます。

RM 互換レベルの 参照 コマンドから、対応するデータビュー定義に移動するには・・・

1. RM 互換レベルの **参照** コマンドにカーソルを起きます。
2. 右クリックでコンテキストメニューを開き、**定義に移動** を選択します。
3. データビュー エディタに移り、対応する項目定義の行にカーソルが移動します。



逆に、データビュー エディタ上での 項目定義の行から、対応する RM 互換 ハンドラの 参照 コマンドに移動するには ・・・

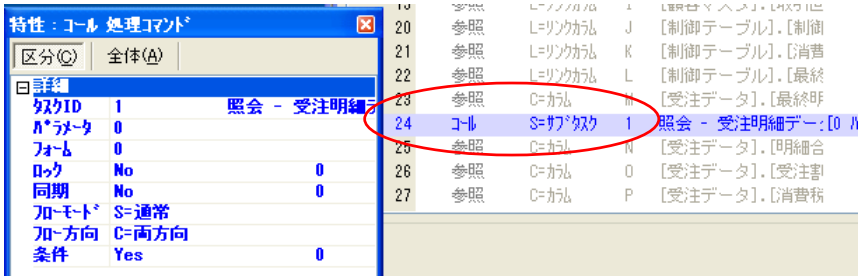


1. データビュー エディタで、項目を定義するコマンド（**C=カラム**、**V=変数**、**P=パラメータ**）にカーソルを置きます。
2. 右クリックでコンテキストメニューを開き、 **参照に移動** を選択します。
3. ロジックエディタに移動し、対応する **参照** コマンドの行にカーソルが移動します。

定義に移動、**参照に移動** は、メニュー **編集(E)** → **クイックアクセス(Q)** のサブメニューからも選択することができます。

8.12.6. コマンドの定義

V9Plus のレコードメイン中のロジックは、V10 の RM 互換ハンドラに移行されます。



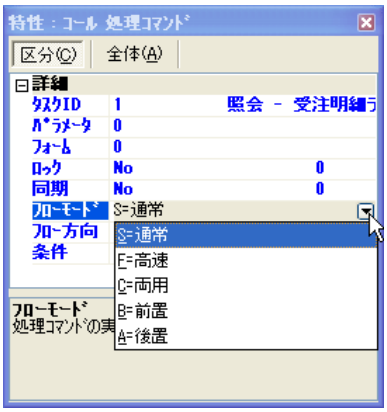
ただし、すべてのオペレーションが RM 互換 ハンドラ中で定義できるわけではありません。以下のような制限に留意してください。

- ブロック コマンドの **While** オプション（V9Plus の Loop オプション）は利用できません。
- **変数** 宣言（V9Plus の**セレクト** 変数 コマンド）は利用できません。
- **エラー** コマンドの **R=復帰** オプションは利用できません。

また、他のハンドラ中のコマンドと違い、RM 互換ハンドラのコマンドには次のように**フローモード** と **フロー方向** のパラメータがあります。 これらは、V9Plus でのフローの設定と同じ意味です。

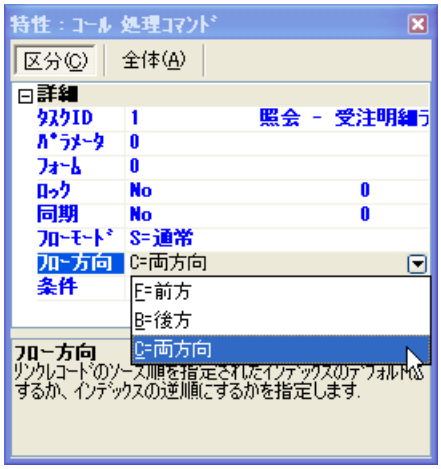
フローモード:

- S=通常
- F=高速
- C=両用
- B=前置
- A=後置



フロー方向:

- F=前方
- B=広報
- C=両方向



8.12.7. RM 互換ロジックユニットの実行時動作

RM 互換ハンドラは、V9Plus におけるレコードメインのように動作します。すなわち、カーソルがコントロールから別のコントロールに移動するときには、RM 互換ハンドラ中の対応する **参照** コマンドの間に定義されたロジックが、条件やフロー定義に従って実行されます。

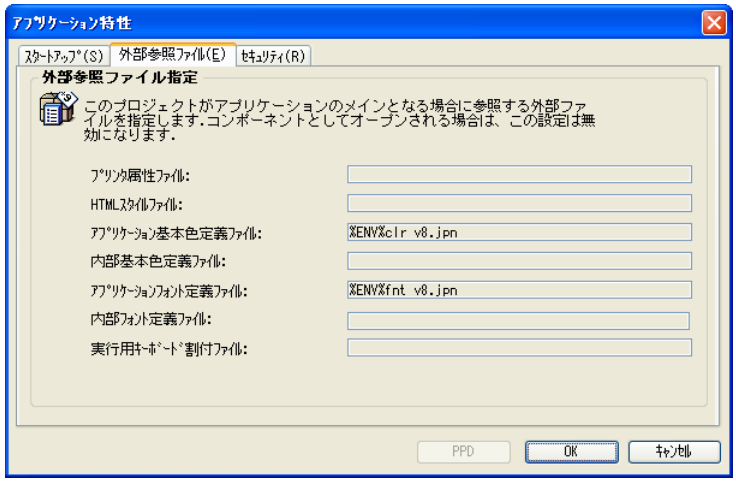
参照 コマンドは **ブロック** コマンドで囲んで条件付けすることもできます。この場合には、カーソルは、コントロール特性の **パーキング** 関係の設定と、ブロックコマンドの条件とをあわせた条件に従って、パーキングの可・不可が決定されます。

8.13. dbMAGIC V8.2 からの移行

dbMAGIC V8.2 およびそれ以前のバージョンから V10 に移行しようとしたとき、8.1.1 バージョン別の移行パスですでに説明したように、直接移行することはできないので、V8.2 → V9.4SP6a → V9Plus コンバータ というステップを踏んで移行することになります。
ここでは、V8.2 あるいはそれ以前のバージョンから移行する場合の若干のヒントを紹介します。

8.13.1. V9Plus での色/フォントテーブル変換は不要

V8.2 と V9Plus を比較したとき、V9Plus では色テーブルやフォントテーブルに変更があり、1～100 番までがシステム予約の番号となったため、V8.2 での色/フォントテーブルをそのまま使うことができなくなりました。そのため、V9Plus に移行する際には、V9Plus 開発版に添付されている色・フォントテーブル変換ユーティリティを使って変換しなければなりませんでした。
さて、V8.2 のアプリケーションを V10 に移行する際には、いったん V9Plus を経由して移行するのですが、V9Plus でアプリケーションを利用する予定がない場合には、色・フォントテーブルの変換は不要になります。これは、V10 では色、フォント、キーボード割付テーブルが、開発用、実行用、内部用と用途によって別々に定義できるようになったからです。このため、アプリケーション実行時用の色・フォントテーブルとして、V8.2 で使っていた色・フォント定義ファイルをそのまま設定することができるようになります。このときでも、V10 での開発時の画面には全く影響を与えることがありません。

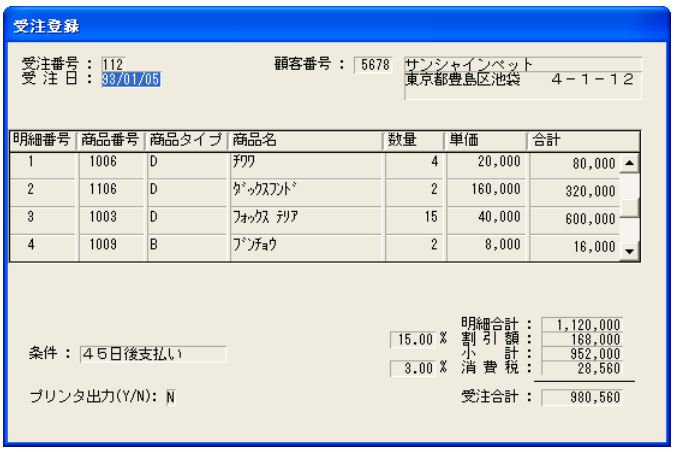


この場合、色・フォントテーブルは、**アプリケーション特性** ダイアログの **外部参照ファイル(E)** タブにパラメータに設定しておくのがよいでしょう。

8.13.2. テーブルのタイトルが消える

V8.2 のテーブルコントロールでは、タイトルはテキストコントロールで実現されていました。
右図は V8.2 で実行している画面ですが、明細の子タスクで表示しているテーブルコントロールのタイトル（明細番号、商品番号、商品タイプ・・・)は、すべて、それぞれ別々のテキストコントロールであり、テーブルコントロールとは親子関係を持って結合されています。

V8.2 での画面



このようにして作成されているプログラムを V9Plus を経由して V10 に移行し、さらに GUI エンハンサを適用すると、テーブルは Windows スタイルのものに変換されます。このときにプログラムを実行してみると、右図に見るように、明細テーブルのタイトルが消えてしまっています。

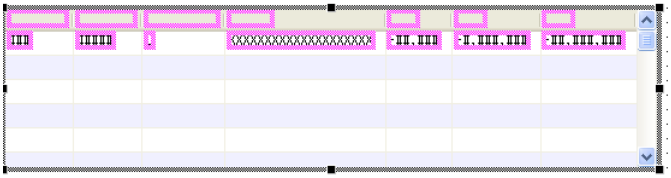
V10 での画面（GUI エンハンサ実行後）

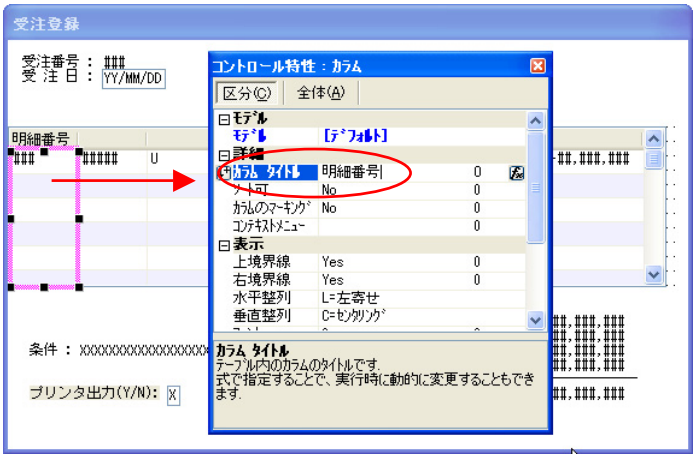
実際には、タイトルはテキストコントロールとして残っていますが、これらのテキストコントロールはテーブルコントロールの背面に隠れてしまっているために表示されない、というのが真相です。

試しに、V10 のフォームエディタを開いて、テーブルコントロールを選択してみると、右図に見るように、タイトルのところにテキストコントロールが存在するのが確認できます。ただし、背面にあるので見ることができません。

この問題に対しては、自動的に解決する方法はありません。V10 のフォームエディタ上で、各カラムを選択してカラムの **タイトル** 特性にタイトルをひとつひとつ入力していってください。

V10 フォームエディタ上でテーブルコントロールを選択





8.14. ファントムタスク

ファントムタスクとは、タスク特性の「ウィンドウのクローズ」を No に設定して、タスクが終了した後も画面が表示され続けるタスクです。オンラインの 1:N のタスクでよく使われる手法です。
Ver 9.40J SP3 において、Magic のイベントドリブンアーキテクチャとの親和性を向上させるため、マウスによるタスク間のフォーカスの移動方法が拡張されました。この機能拡張は V10 でも有効です。
その結果、Ver 9.40J SP3 よりも古いバージョンからアプリケーションを V10 に移行する場合、この機能拡張の影響を検討し、必要ならばプログラムの修正が必要になります。

8.14.1. Ver 9.40J SP3 での機能拡張とは？

Ver 9.40J SP3 より以前のバージョンでは、マウスクリックによりファントムタスクにフォーカスを移動するには、レコードメインにファントムタスクの呼び出しのためのコールコマンドが定義されている必要がありました。この場合、親タスクにカーソルがある状態でファントムタスク（子タスク）をクリックすると、現在止まっているカーソルに対応する項目のセレクトコマンドから、そのコールコマンドまでの間のコマンドが高速モードで実行され、コールコマンドが実行されて子タスクに制御が移ります。
Ver 9.40J SP3 からは、レコードメインだけでなく、他のハンドラレベル（コントロール前処理/後処理、イベントハンドラなど）にコールコマンドが定義されている場合でも、ファントムタスク呼び出しのために使われるようになりました。この場合、親タスクにカーソルがある状態でファントムタスク（子タスク）をマウスでクリックすると、親タスクの

1. レコードメイン
2. 有効なイベントハンドラ（下から上に向けて）

の順に、ファントムタスク呼び出しのコールコマンドがあるかを検索します。見つかったら、現在フォーカスのある項目から、そのコールコマンドまでの間に定義されているコマンドは、当該イベントハンドラ中で定義されているものも含め、実行されるようになります。

8.14.2. 機能拡張の波及と対応方法

この機能拡張の波及効果として、Ver 9.40J SP1a およびそれ以前の旧バージョンから移行を行うと、プログラムの構造によっては、この機能拡張によって、ファントムタスクにカーソルが移動してほしくないときに移動してしまう、という現象が起こる可能性があります。
このような場合には、次のようにファントムタスクを呼び出すコールコマンドに条件付けを行い、不要なときにコールコマンドが実行されないように、プログラムを修正してください。

- コントロール後処理でファントムタスクを呼び出している場合、コールコマンドに Level(0) = 'CS_コントロール名' というような条件を付けてください。
- プッシュボタンのイベントハンドラでファントムタスクを呼び出している場合には、LastClicked ()='ボタンのコントロール名' という条件を付けてください。

8.15. トラブルシューティング

v9Plus コンバータ実行中に発生するエラーの代表的なものは、以下のものがあります。

現象:	エラーメッセージ Missing Mandatory Parameters が出る。
原因:	-EXPORT と –PROJECT パラメータは必須です。V9Plus コンバータ実行時にこのパラメータを指定しなかった場合にこのエラーが出力されます。
対応:	これらのパラメータを指定します。

現象:	エラーメッセージ Invalid Export File が出る。
原因:	V9Plus のリポジトリ出力ファイルの形式が、V9Plus コンバータで処理できるものでなかった場合に出ます。V9Plus コンバータは、最初にリポジトリ出力ファイルのバージョンをチェックし、Ver 9.40J SP6 以降の開発版でリポジトリ出力したもののみ、処理します。
対応:	MCF を V9Plus Ver 9.40J SP6 以上の開発版でリポジトリ出力して、再度実行してください。

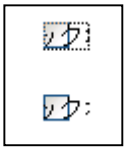
現象:	エラーメッセージ Project Directory Already Exists が出る。
原因:	-PROJECT パラメータで指定したディレクトリに、プロジェクトファイルがすでに存在している場合に出ます。
対応:	別のディレクトリを指定するか、既存のプロジェクトが不要であれば削除してから再実行してください。

現象:	内部イベントのリテラルが消えてしまう。(''EVENT という形になってしまう)。
原因:	V9 よりのマイグレーション時に ACT が消えてしまう問題は、ほとんどの場合、V9Plus でリポジトリ出力時に mgconstw.jp2 を使っていない場合に起こります。8.3.1 リポジトリ出力時の注意事項 を参照してください。また、V9Converter 実行時のパラメータを正しく設定していない場合、同じ問題が起こります。
対応:	● V9Plus でリポジトリ出力をするときには、mgconstw.jp2 を使ったかどうかを確認してください。不明の場合には、8.3.2 リポジトリ出力ファイルの確認 を参照してチェックしてください。もし mgconstw.jp2 を使っていない場合には、mgconstw.jp2 を使って再度リポジトリ出力してください。 ● v9converter.exe 実行時に、パラメータ「-LANG=JPN -LOCALS=J,./: 」が抜けていないことを確認して、再度実行してください。

現象:	実行時、プッシュボタンやキーなどが効かない。 カーソルの動きがおかしい。
原因:	内部イベント(アクション)が、移行時に消えてしまっていることが考えられます。
対応:	プログラムを開いて、内部イベントが ''EVENT などとなってしまっていないかを確認してください。もしそうなっていたら、上の「内部イベントのリテラルが消えてしまう。」の項に従って対応してください。

現象:	パラメータが正しく渡らないプログラムがある。

原因:	パラメータを受け取る場合に、**P=パラメータ**でではなく、**V=変数**で受け取っていて、かつ、照会リンクを使っているプログラムで起こることがあります。詳細は、8.4.4 照会リンクの移行 を参照してください。
対応:	パラメータは **P=パラメータ** で受け取るようにしてください。

現象:	 右図のように、チェックボックスの表示が乱れる（チェックボックスの上にテキストの一部が重複して表示されているように表示される）
原因:	チェックボックスにラベル(**テキスト** 特性)が設定されていて、このテキストが隠れるように幅を狭めている場合、V9Plus では表示されませんでしたが、V10 ではこれが前面に現れるようになり、このような表示になります。
対応:	ラベルを表示する必要がない場合には、**テキスト** 特性を削除してください。

現象:	GUI エンハンサ適用後、テーブルコントロールのタイトルが消えてしまう。
原因:	V8.2 あるいはそれ以前のバージョンから移行してきた場合、テーブルコントロールのタイトルはテキストコントロールとして定義されていますが、Windows スタイルのテーブルコントロールが前面に表示されるので、タイトルのテキストコントロールが隠れて見えなくなるのが原因です。8.13 dbMAGIC V8.2 からの移行 を参照してください。
対応:	タイトルは、**カラム タイトル** 特性で設定するように修正してください。

現象:	移行後、データソースリポジトリで定義されている SQL テーブル（Oracle、MS-SQL、DB2UDB など)のカラムの格納形式特性が変わり、データを正しくアクセスできなくなる。(例: ZString が String に変わってしまうなど)
原因:	移行中にリポジトリ入力をする際に、必要なデータベース定義がされていないために、Default Database をもとにした格納形式のデフォルトにリセットされてしまうためです。
対応:	移行中に利用する V9Plus や V8.2 開発版では、リポジトリ入力を行うに先立って、アプリケーションが利用しているデータベースをデータベーステーブルに定義してください。これは、そのバージョンでは実行を行わない場合にも必要です。

9. Datetime データ型のサポート

最近のリレーショナル型 DBMS では、Datetime 型というデータ型がサポートされています。Datetime 型は、ひとつのカラムに日付と時刻の両方のデータが格納されています。
Magic では、日付型と時刻型とは用意されていますが、Datetime 型はサポートされていませんので、DBMS の Datetime 型データを利用するのが不便でした。
V10 では、DBMS の Datetime 型カラムを利用するのに便利な機能が追加されています。

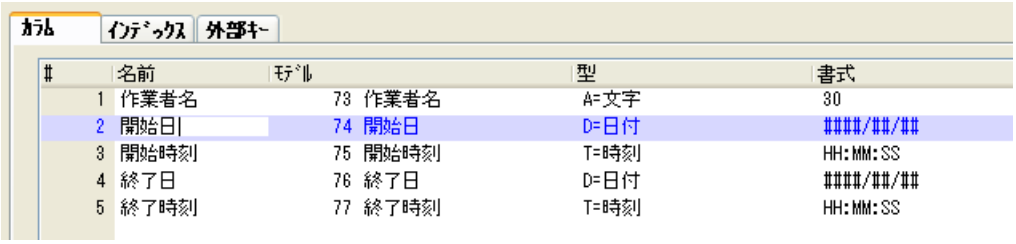
- Dateteime データ型のサポート機能は、SQL テーブル、ISAM AS/400 テーブル、および 埋め込み SQL タスクの変数項目でだけ有効です。その他の ISAM テーブルの日付、時刻型カラムのカラム特性には表示されません。
- TravelAgency のサンプルでは、DBMS として Pervasive を使っているため、本章で解説する Datetime サポート機能を使うことができません。
- 本章の内容を実際に試してみるには、Pervasive に定義されているデータソースを SQL DBMS に移行して、試してみてください。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。	
Datetime の定義	**データソース → データソースタイプ → テーブルソース → カラムテーブル → カラム特性 → SQL → Datetime の一部**
定義取得	**ユーティリティ → 定義取得 → Datetime データの定義取得**
Datetime 関連の関数	**式エディタ → 関数ディレクトリ → AddDateTime および DifDateTime**

9.1. Datetime 型のカラムをデータソースリポジトリに定義

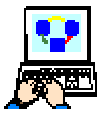
DBMS の Datetime 型のカラムには日付情報と時刻情報が含まれているので、Magic で利用するには、Magic の日付型と時刻型の二つの項目にマッピングしなければなりません。
このために、V10 のデータリポジトリでは、SQL データベースに定義されているデータソースの日付型と時刻型のカラムに、 **Datetime の一部** という特性があります。

TravelAgency アプリケーションのデータリポジトリを開くと、第 11 番目に **作業者時間** というデータソースが定義されています。ここには、以下のようなカラムが定義されています。



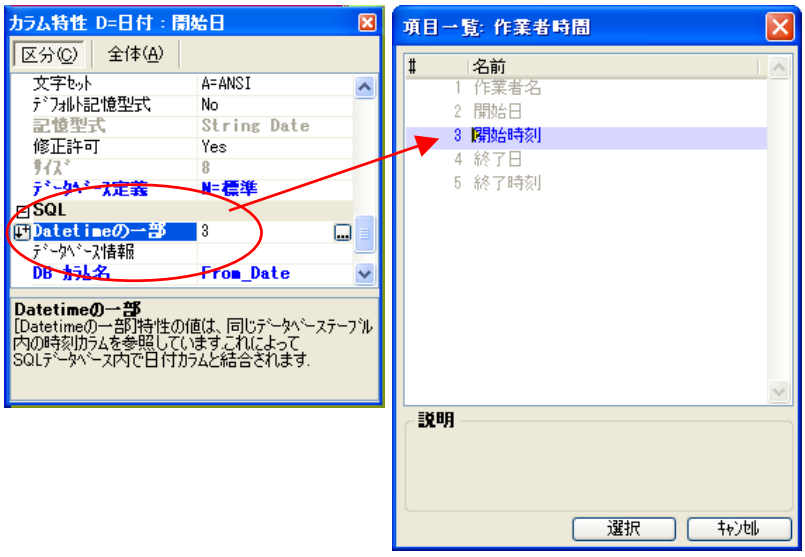
#	名前	モデル		型	書式
1	作業者名	73	作業者名	A=文字	30
2	開始日	74	開始日	D=日付	####/##/##
3	開始時刻	75	開始時刻	T=時刻	HH:MM:SS
4	終了日	76	終了日	D=日付	####/##/##
5	終了時刻	77	終了時刻	T=時刻	HH:MM:SS

ここで、開始日 と 開始時刻 カラムは、DBMS 上では、From_Date という、一つの Datetime 型のカラムにマッピングされています。Magic のデータソースリポジトリでこの関連付けを定義するために、カラム特性で **Datetime の一部** 特性を設定します。

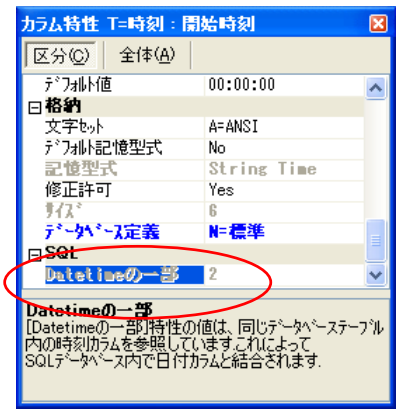


Magic の日付型と時刻型のカラムを、DBMS 上の一つの Datetime 型にマッピングするには:

1. 日付型のカラム（今の例では **開始日**）の **DateTime の一部** 特性には、対応する時刻型のカラムの番号（ここでは 3、すなわち 開始時刻 カラム)を指定します。この特性には、直接カラム番号を入力することもできるし、ズームして一覧から選択することもできます。



2. 一方、時刻型のカラム特性にある DateTime の一部 特性は、読込専用となっていて、開発者が設定することはできません。ここには、日付型の DateTime の一部 に指定された場合に、対応する日付型のカラム番号が自動的に設定されます。この特性は読込専用で、開発者が変更することはできません。



- 一つのテーブル上に定義できる 日付・時刻のペアの数には、制限はありません。
- Datetime の一部 として定義されている日付型カラムの格納形式としては、String Date のみがサポートされます。この場合 **格納形式** 特性は読込専用となり、開発者が変更することはできません。同様に、Datetime の一部として定義されている時刻型の格納形式としては、String Time のみがサポートされています。

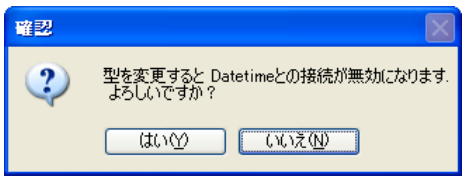
Datetime の一部として定義されている 日付・時刻型のペアは、DBMS 上では同一カラムに対応していますので、次の特性は自動的に共通な値に設定されます。

- Null 値可
- データベース デフォルト値
- データベース情報
- DB カラム名
- タイプ
- ユーザ タイプ

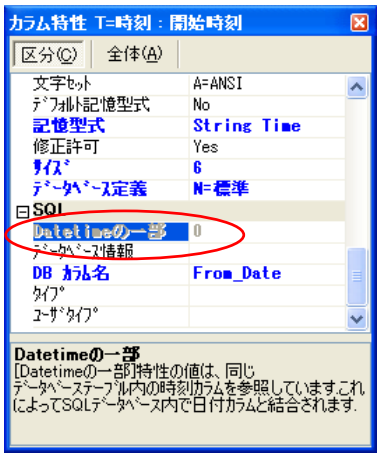
9.2. Datetime を構成する日付/時刻項目の変更

9.2.1. 変更時

Datetime の一部 として定義されている日付型、あるいは時刻型のカラムを変更しようとすると、Datetime 型のペアとしての関連が失われてしまいます。このような場合には、次のような確認ダイアログが表示され、開発者が **はい(Y)** で答えた場合のみ変更がなされます。



この場合には **Datetime の一部** の設定はリセットされて 0 になり、Datetime の一部 として読込専用となっていた特性は有効化されます。



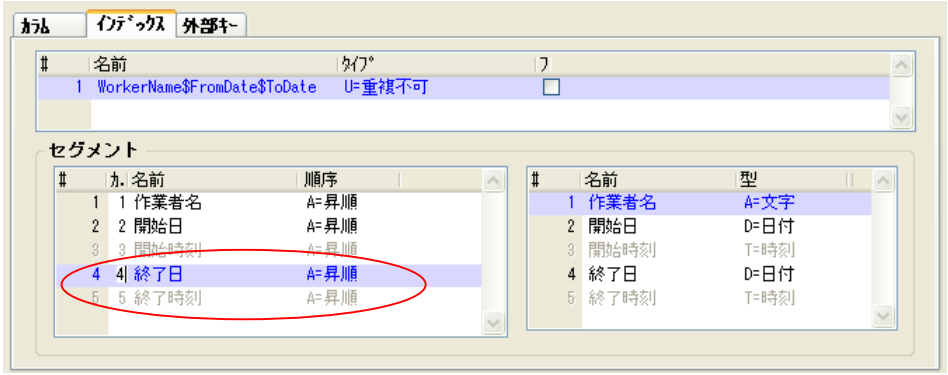
9.2.2. 削除時

Datetime の一部 として定義されている日付型、あるいは時刻型のカラムの一方だけを削除することは可能です。この場合にも、残された他方のカラムの **Datetime の一部** 特性は 0 にリセットされ、Datetime の一部として読込専用となっていた特性は有効化されます。

9.3. インデックス

9.3.1. インデックス定義

データリポジトリの インデックス に、Datetime 型に対応する日付と時刻のペアを、インデックスセグメントの一部として定義することもできます。



この場合、日付型と時刻型は必ずペアになってインデックスセグメントとなります。一方だけをインデックスセグメントに入れることはできません。
例えば上の場合、セグメント 4 (**終了日** カラム)を追加すると、対応する時刻型の **終了時刻** カラムも自動的に入ります。

削除時にも同様に、これらのカラムを単独で削除することはできません。上の例では、終了日を削除すると、終了時刻も自動的に削除されます。

9.3.2. 外部キー

外部キーの中に Datetime の一部 として設定されている 日付・時刻 ペアがあった場合には、参照されているテーブルのインデックスのセグメントでも **Datetime の一部** の定義が一致していなければなりません。定義に差異があった場合には、シンタックスチェックに引っかかります。

9.4. 定義取得

Datetime 型のカラムを含むテーブルを Magic に定義取得する場合、V9Plus では、日付型のカラムとして定義取得されていました。
V10 ではこのような場合、次のような確認ダイアログが表示されます。

ここで開発者がどちらを答えるかにより、定義取得の結果が異なってきます。

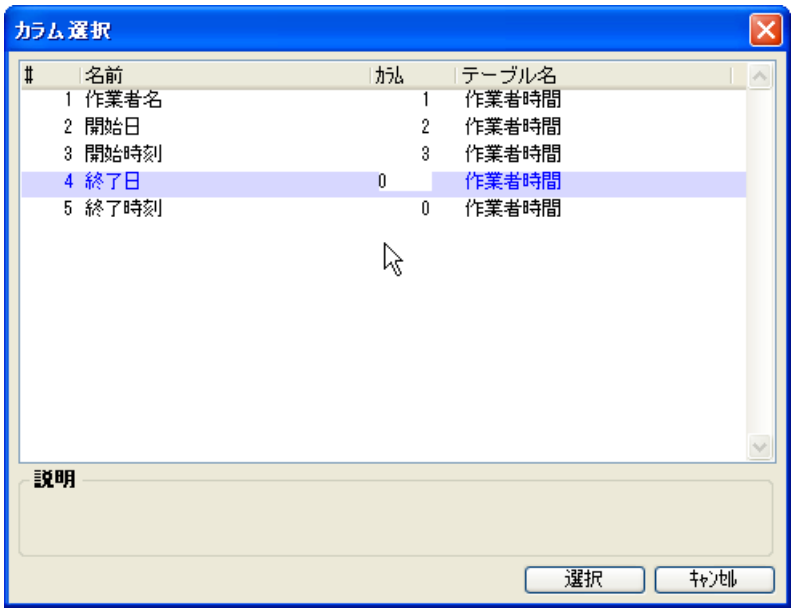
- **はい(Y)** を選んだ場合には、Datetime の一部 特性により関連付けられた 日付型と時刻型の二つのカラムとして定義取得されます。
- **いいえ(N)** を選んだ場合には、V9Plus と同様に 日付型カラムのみとして定義取得されます。

- 複数のテーブルに対して定義取得を行った場合、上記の確認ダイアログは 1 度だけ表示されます。Datetime 型のカラムが複数あった場合には、すべてに同じ設定が適用されます。Datetime の定義取得の方法を変えたい場合には、定義取得を 2 回にわけて行ってください。
- 上記のダイアログは、埋め込み SQL タスクで、SQL APG を行ったときに Datetime 型のカラムがあった場合にも表示されます。

9.5. タスクでの利用

データリポジトリで Datetime の一部 として定義した日付型・時刻型のカラムは、タスクの中で他のカラムと同様に利用することができますが、以下の点に留意してください。

- データビュー エディタ で **C=カラム** によりカラムを選択する場合には、日付型と時刻型の両方を選択しなければなりません。
- データビュー エディタで、一方のカラムのみを選択した場合には、他方のカラムも自動的に選択されます。
- データビュー エディタで、一方のカラムを削除した場合には、他方のカラムも自動的に削除されます。
- 範囲、位置付 などは、日付型、時刻型それぞれに独立して設定することができます。
- ソートには、日付型、時刻型いずれか単独で設定することもできるし、両方を設定することもできます。
- Datetime の一部 として定義された日付型と時刻型がインデックス セグメントとして定義されていて、そのインデックスが結合リンク、あるいは外部結合リンクで指定されている場合には、その日付型と時刻型は、両方選択しなければなりません。
- APG を行う際、カラム選択画面で、Datetime の一部となっている日付型・時刻型の一方を非選択にすると(すなわち、0 を設定すると)、他方のカラムも自動的に 0 となります。



9.6. Datetime 関連の関数

V10 では、Datetime 型を扱いやすくするために、以下の関数が追加されています。

- DifDateTime
- AddDateTime

9.6.1. DifDateTime

DifDateTime は、二つの 日付・時刻ペアから、その差異を計算します。このときに、日付の繰り下がり（時刻が 00:00:00 以下になる場合には、日付を1減らして時刻に 24 時間加える）を考慮します。結果は、日数(数値型) と時刻型の二つの変数に格納します。

形式: `DifDateTime (Date1, Time1, Date2, Time2, DestDays, DestTime)`

パラメータ: Date1、Time1: 計算の基準となる日付と時刻

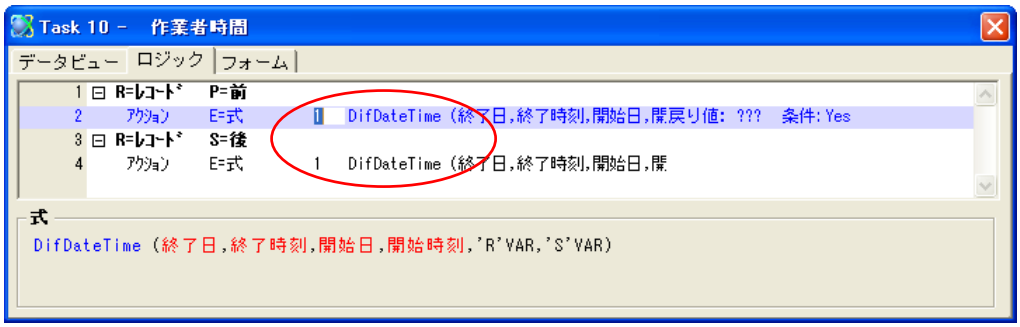
Date2、Time2: 差し引きとなる日付と時刻

DestDays: 計算結果のうち、日数の部分を格納する数値型変数。'x'VAR の形式で指定します。

DestTime: 計算結果のうち、時間の部分を格納する時刻型変数。'x'VAR の形式で指定します。

戻り値: 成功した場合には True、失敗した場合には False が返ります。

プログラム 10 番 **作業者時間** に、DifDateTime を使った例があります。



9.6.2. AddDateTime

AddDateTime は、日付・時刻ペアを基準として、年数、月数、日数、時間数、分数、秒数 をそれぞれ数値型データとして与え、加算します。このとき、月、日、および時刻の繰上げを考慮します。

形式: `AddDateTime (Date, Time, Years, Months, Days, Hours, Minutes, Seconds)`

パラメータ: Date、Time: 計算の基準となる日付と時刻。結果もここに格納されます。いずれも、'x'VAR の形式で指定します。

Years, Months, Days, Hours, Minutes, Seconds: それぞれ、加算する年数、月数、日数、時間数、分数、秒数を表す数値型データ

戻り値: 成功した場合には True、失敗した場合には False が返ります。

10. ビュー事前読込

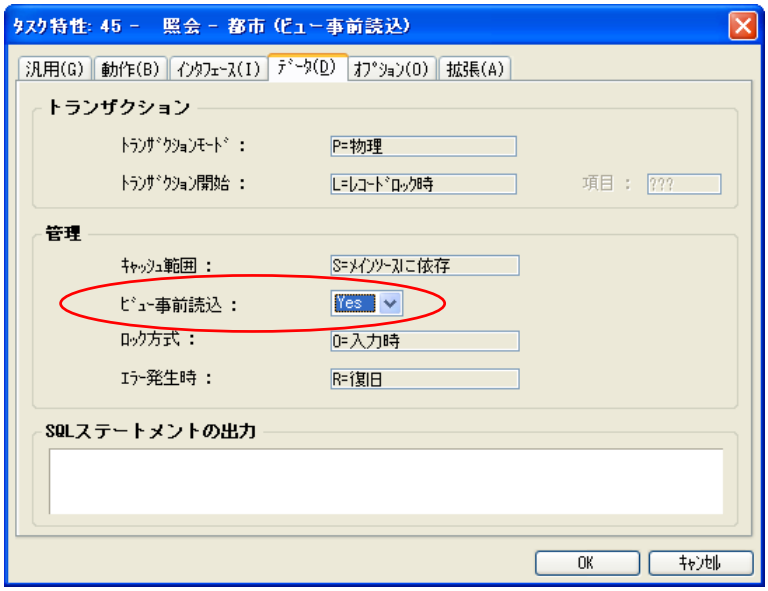
Magic でテーブルをオンライン画面に表示させるとき、テーブルコントロールのスクロールバーのつまみの位置を正確に表示してほしい、という要望がありましたが、V9Plus までは対応することができませんでした。
V10 では、**ビュー事前読込** という新機能を用いて、これに対応することができます。

本章では、ビュー事前読込の使い方と、注意点について説明します。

本章の内容については、リファレンスマニュアルの以下の箇所も参考にしてください。

ビュー事前読込 特性	**プログラム → タスク特性 → データ → ビュー事前読込**
DbView… 関数	**式エディタ → 関数ディレクトリ → データベース関数**

10.1. ビュー事前読込特性



ビュー事前読込機能は、タスク単位で、タスク特性で設定します。タスク特性を開き、**データ(D)** タブを開くと、**管理** カテゴリに **ビュー事前読込** 特性があります。ここには、Yes あるいは No を設定します。デフォルトは No です。

この特性が No の場合には、デフォルトの動作となり、V9Plus までと同じ動作になります。すなわち、Magic エンジンはタスクオープン時に、初期画面を表示するに足るレコード数だけ、レコードを読み込みます。それ以上のレコードは読み込みません。画面をユーザがスクロールすると、Magic エンジンは画面を表示するのに必要最小限のレコードのみを随時読み込みます。このため、範囲設定条件を満たす全体のレコード数が正確にはいくつあるのかがわかりませんので、スクロールバーの表示は正確でないことになります。

この特性が Yes の場合には、Magic エンジンはタスクオープン時にタスクに設定されている範囲設定条件を満たすレコードをいったんすべて読み込み、キャッシュします。この時点でデータビュー全体に含まれるレコード数が決定されるので、スクロールバーはそれに基づいて正しい表示をできるようになります。

10.2. ビュー事前読込関連の関数

ビュー事前読込に関連して、次の関数が追加されています。

目的　現在のデータビュー内のレコード数を返します。

シンタックス　DbViewSize(世代番号)

パラメータ　世代番号 … タスクの階層位置を表す番号。カレントのタスクが 0、親タスクが 1、その親タスクが 2 などとなります。

戻り値　数値 … キャッシュされたレコード数を表す数値

補足　メインソースが定義されていなかった場合は、0 が返ります。
ビュー事前読込 特性が Yes に設定されている場合は、ビュー定義によって取得された全てのレコード数が返ります。

目的　タスクのデータビューがキャッシュされている場合、キャッシュ内のレコード番号を取り出します。この関数を使用することで、どの行に現在パークしているかが確認できます。

シンタックス　DbViewRowIdx (世代番号)

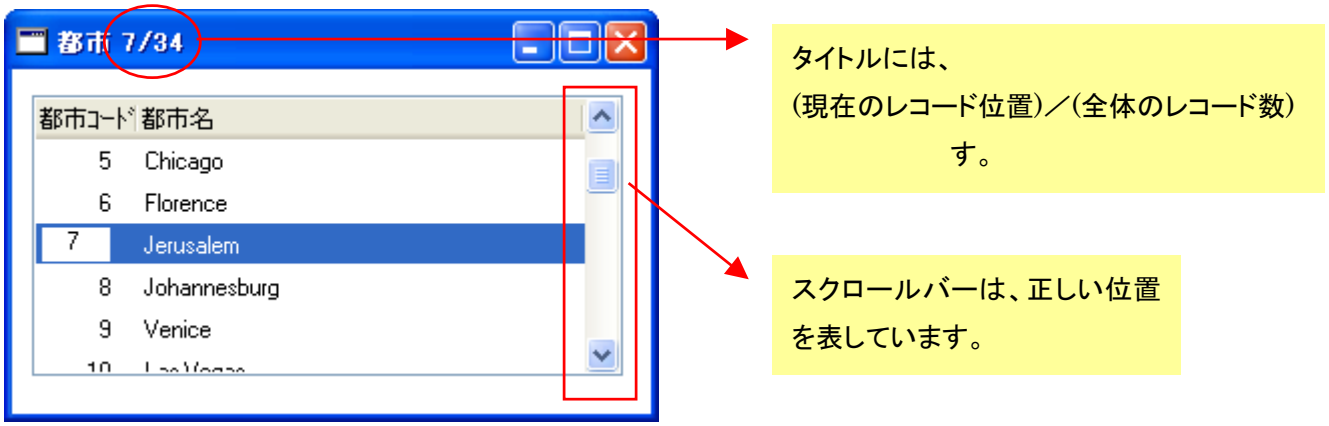
パラメータ　世代番号 … タスクの階層位置を表す番号。カレントのタスクが 0、親タスクが 1、その親タスクが 2 などとなります。

戻り値　数値 … キャッシュされたビュー内での、現在レコードの連番を表す数値

補足　この関数は、タスクの ビュー事前読込 特性が、Yes に設定された場合のみ有効です。メインソースが定義されていなかったり、ビュー事前読込 特性が No に設定されている場合は、0 が返ります。

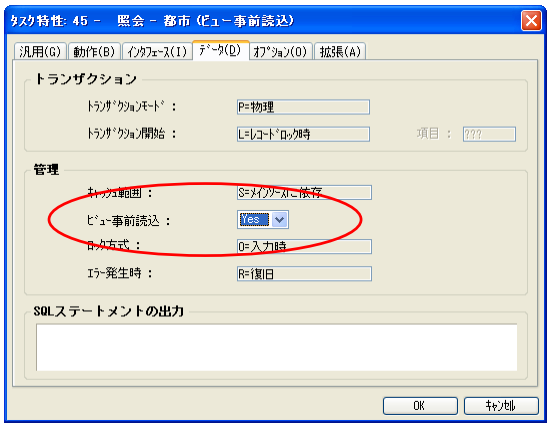
10.3. ビュー事前読込機能のサンプル

TravelAgent サンプルのプログラム 45 **照会 - 都市（ビュー事前読込）**でビュー事前呼出を利用しています。このプログラムでは、スクロールバーが正確に表示されるほかに、ウィンドウのタイトルに、全体のレコード数と現在のレコード位置を表示するようにしています。

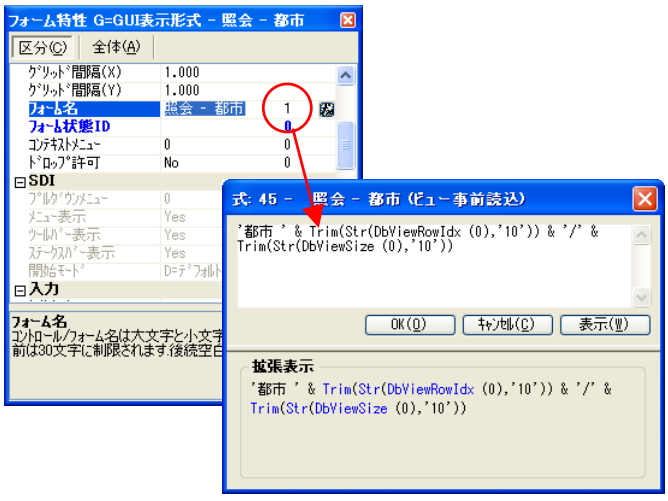


このプログラムは、APG で作成されるオンライン照会プログラムに、次のような変更を加えて作成できます。

スクロールバーで正しい位置を表示させるには：単に、タスク特性で **ビュー事前読込** 特性を **Yes** にするだけで実現できます。



タイトルに、「(現在のレコード位置)／(全体のレコード数)」を表示させるには：フォームの **タイトル** 特性で、DbViewSize()、DbViewRowIdx() 関数を使って、タイトルを式で設定します。サンプルでは次のような式を使っています。




```
'都市 ' & Trim(Str(DbViewRowIdx (0),'10')) & '/' & Trim(Str(DbViewSize (0),'10'))
```

10.4. ビュー事前読込利用時の注意事項と補足事項

ビュー事前読込機能を使う上での注意事項

- **ビュー事前読込** 特性が Yes になっている場合、その動作仕様上、範囲設定条件を満たすレコードがタスクオープン時にすべて読み込まれます。対象レコード数が大きい場合には、それに比例して時間がかかります。実用的なスピードを得るために、キー項目で範囲を絞り込んで、対象レコード数がせいぜい数百程度になるように設計することをお勧めします。

ビュー事前読込に関する補足事項

- **ビュー事前読込** 特性は、**キャッシュ範囲** 特性が、**位置とデータ** または **メインソースに依存** に設定されている場合のみ有効になります。**キャッシュ範囲** 特性が **メインソースに依存** の場合には、さらに、データソースリポジトリでのデータソース特性で、**キャッシュ範囲** 特性が **位置とデータ** に設定されていなければなりません。それ以外の場合にはシンタックスチェックでエラーになります。
- **ビュー再表示** イベントを実行すると、再度テーブルの内容が事前読み込みされます。テーブルのつまみは、常にレコード数に反映されます。レコードが追加されたり、削除されると、つまみ用のレコードサイズも増えたり減ったりします。
- タスクの **ビュー事前読込** が Yes に設定された場合、ツリーコントロールに対するこの特性も Yes と見なされます。

11. XML サポート

XML（Extensible Markup Language）は、非常に柔軟なデータ表現を可能とする、国際標準をベースとしたデータ形式です。この柔軟性のために、インターネット、イントラネット、その他の媒体を通してソフトウェア間のデータ共有を行うのに大変便利なものです。
V9Plus では、ファイル IO テーブルに XML ファイルを定義し、組み込み関数を使って XML の要素にアクセスすることができました。
V10 では、XML のサポートが更に容易になりました。V10 では XML をデータソースとして、データリポジトリに定義することができるようになり、XML ファイルを他の DBMS のテーブルと同じように取り扱うことができます。
本章では、V10 で新たにサポートされるようになった 次のような XML 機能について説明します。

- XML ファイルをデータリポジトリで、データソースとして定義する。
- 同一の XML ファイルを、異なるマッピングにより異なるデータソースとして定義する。
- XML ファイルをメインソース、リンクソースとするタスクを作成する。
- XML ファイルの特定の要素・データにアクセスする。
- XML ファイルを作成し、データを登録する。

V10 では、XML ファイルの正当性を、スキーマファイル（XSD)を使って検証します。その結果を調べるための新しい関数が定義されています。
また、V10 の XML サポートでは、名前空間が自動的に取り扱われます。

V9Plus での XML サポート機能は、V10 でも有効ですが、この機能と V10 で新たに導入された XML サポート機能とを混在して利用することはお勧めできません。

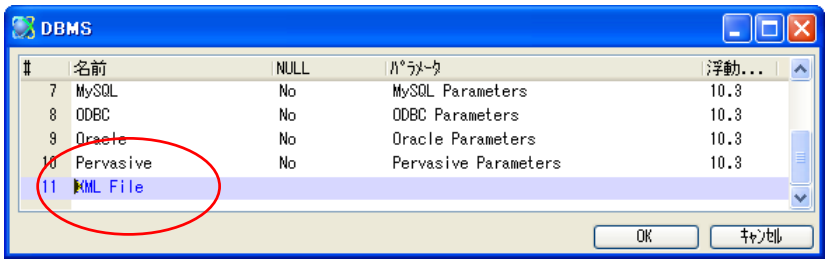
本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

XML サポート全般	**XML 連携** の章
データリポジトリでの定義	**データソース** の章
XML データソースの定義	**データソース → データソースタイプ → XML ソース**
XML データソースの特性	**データソース → データソース特性 → [XML]タブ**
データビュー エディタでの、XML データソース固有の機能	**データビューエディタ → メインソース特性** 以下の XML 関連項目
XMLDBMS の設定	**設定 → DBMS → DBMS 特性 → デフォルトデータタイプ**
XML データベースの設定	**設定 → データベース**

11.1. 設定

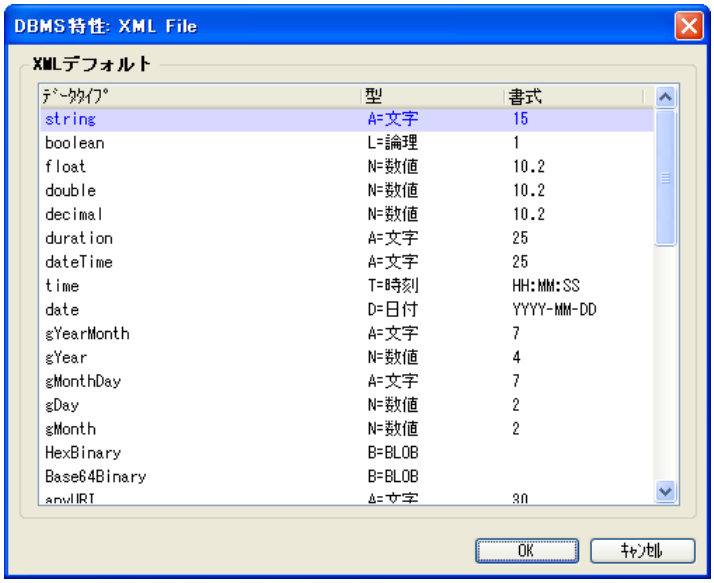
ここでは、V10 の XML サポートに関する設定について説明します。

11.1.1. DBMS リポジトリ



DBMS リポジトリには、XML File というエントリが常に存在します。このエントリは削除することはできません。

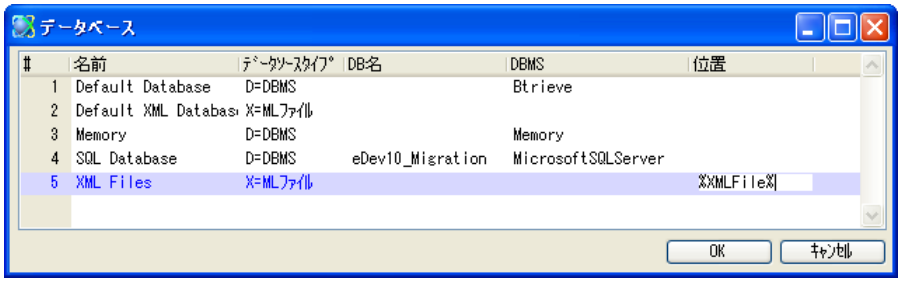
このエントリから、コンテキストメニューで **特性** を開くと、DBMS 特性が表示されます。XML File の DBMS 特性では、XSD でのデータ型と、Magic でのデータ型のデフォルトのマッピングを定義します。



データタイプ 欄は、XSD 中の type= で指定するもので、開発者が追加・削除することはできません。
型 および **書式** 欄は、それに対応する Magic のデータ型および書式を定義します。この設定は開発者が変更することができます。変更された型、および書式は、デフォルトとは異なった色で表示されます。

11.1.2. データベースリポジトリ

XML ファイルに対しても、データベースリポジトリで定義をします。
XML ファイルを扱う場合には、データベース リポジトリの各欄は、次のように設定します。



欄名	設定内容
名前	適当な名前をつけます。
データソースタイプ	**X=ML ファイル** を選択します。
DB 名、および DBMS	空欄のまま。(設定できません)
位置	XML ファイルのあるディレクトリの名前、あるいは URL を設定します。実行時には、ここでの値が、データリポジトリに設定された データソース名 に連結されて、実際のデータソースの名前となります。

- XML データソースを、**動作環境** ダイアログで、 **デフォルト データベース** として設定することは可能です。
- XML データソースを、**ソート/一時データベース** として設定することはできません。

データベース特性

XML データベースを定義した行から、コンテキストメニューで 特性 を選ぶと、XML ファイルの **データベース特性** ダイアログが開きます。

ここにある **一時データソース** には、XML ファイル中のデータを一時的にテーブル形式で読み込むときに使うデータベースを指定します。通常は Memory データベースが適当です。

11.2. XML ファイルをデータリポジトリに定義する

ここでは、XML ファイルをデータリポジトリに定義するための手順を説明します。

XML ファイルをデータリポジトリに定義するには・・・

11.2.1. XML データソースファイル定義から定義取得する

スキーマファイル（XSD）

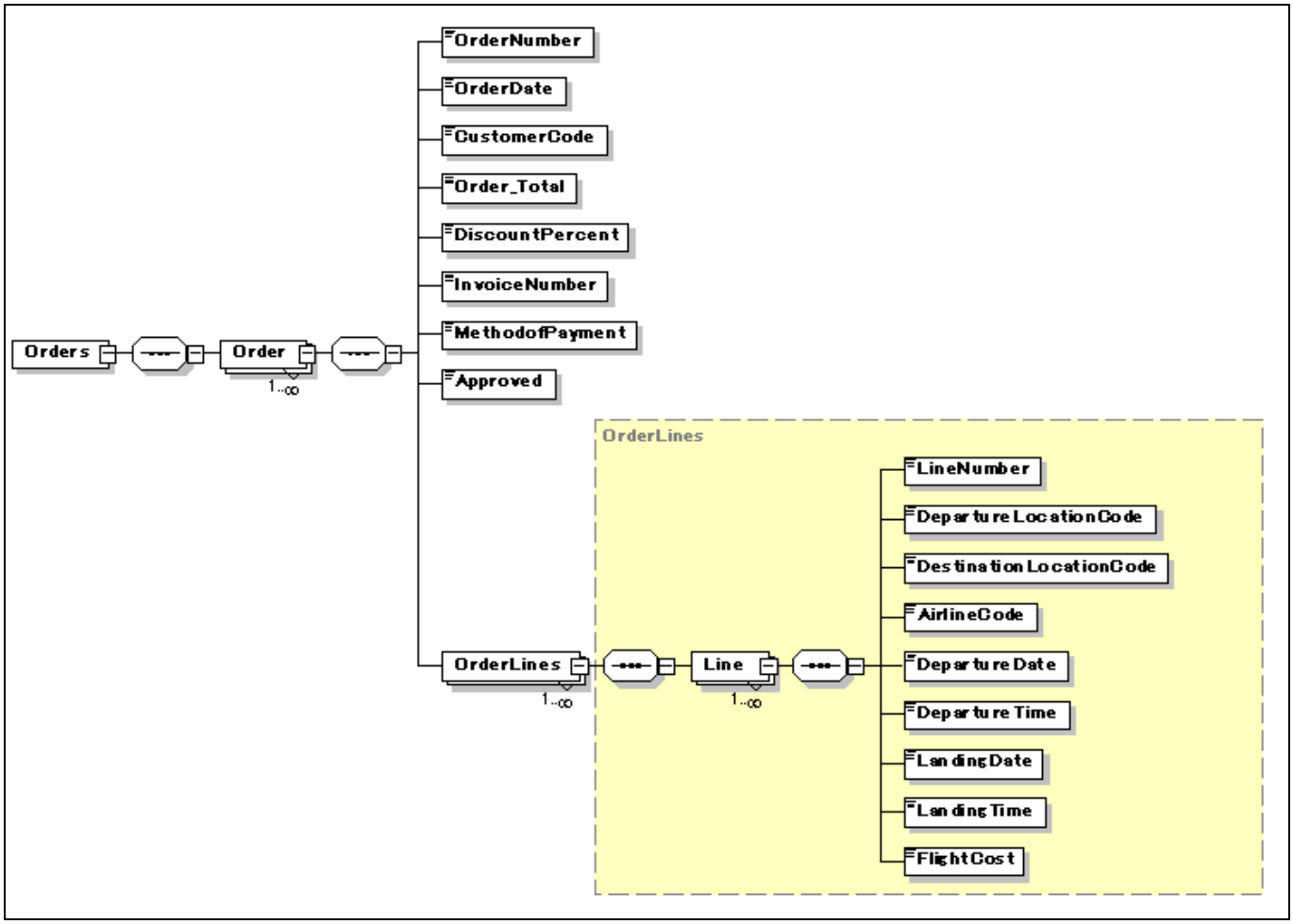
XML データソースをデータリポジトリに定義するには、大前提として、スキーマファイル（XSD）が必要です。

TravelAgent アプリケーションでは、プロジェクトファイルのサブディレクトリ XML の中に、Orders.xsd というスキーマファイルがあります。

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:annotation>
        <xsd:documentation xml:lang="en">
    Order schema for Example.com.
    Copyright 2000 Example.com. All rights reserved.
  </xsd:documentation>
    </xsd:annotation>
    <xsd:element name="Orders">
        <xsd:complexType>
            <xsd:complexContent>
                <xsd:extension base="OrderType">
                    <xsd:sequence>
                        <xsd:element name="Order" maxOccurs="unbounded">
                            <xsd:complexType>
                                <xsd:sequence>
                                    <xsd:element name="OrderNumber" type="xsd:integer"/>
                                    <xsd:element name="OrderDate" type="xsd:date"/>
                                    <xsd:element name="CustomerCode" type="xsd:integer"/>
                                    <xsd:element name="Order_Total" type="xsd:decimal"/>
                                    <xsd:element name="DiscountPercent" type="xsd:integer"/>
                                    <xsd:element name="InvoiceNumber" type="xsd:integer"/>
                                    <xsd:element name="MethodofPayment" type="xsd:string"/>
                                    <xsd:element name="Approved" type="xsd:boolean"/>
                                    <xsd:element name="OrderLines" type="OrderLines" maxOccurs="unbounded"/>
                                </xsd:sequence>
                            </xsd:complexType>
                        </xsd:element>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="OrderType"/>
    <xsd:complexType name="CustomerDetails">
        <xsd:sequence>
            <xsd:element name="Customer_Name" type="xsd:string"/>
            <xsd:element name="PhoneNumber" type="xsd:string"/>
            <xsd:element name="street" type="xsd:string"/>
            <xsd:element name="HomeNumber" type="xsd:integer"/>
            <xsd:element name="city" type="xsd:string"/>
            <xsd:element name="BirthDate" type="xsd:date"/>
            <xsd:element name="zip" type="xsd:string" minOccurs="0"/>
            <xsd:element name="Gender" type="xsd:string"/>
            <xsd:element name="GoldCustomer" type="xsd:boolean"/>
            <xsd:element name="Marital" type="xsd:string"/>
            <xsd:element name="Favorites" type="xsd:string"/>
            <xsd:element name="FavoritesReading" type="xsd:string"/>
            <xsd:element name="FavoritesMusic" type="xsd:string"/>
```

```
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="OrderLines">
        <xsd:sequence>
            <xsd:element name="Line" maxOccurs="unbounded">
                <xsd:complexType>
                    <xsd:sequence>
                        <xsd:element name="LineNumber" type="xsd:integer"/>
                        <xsd:element name="DepartureLocationCode" type="xsd:integer"/>
                        <xsd:element name="DestinationLocationCode" type="xsd:integer"/>
                        <xsd:element name="AirlineCode" type="xsd:integer"/>
                        <xsd:element name="DepartureDate" type="xsd:date"/>
                        <xsd:element name="DepartureTime" type="xsd:time"/>
                        <xsd:element name="LandingDate" type="xsd:date"/>
                        <xsd:element name="LandingTime" type="xsd:time"/>
                        <xsd:element name="FlightCost" type="xsd:decimal"/>
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

このスキーマ定義は、下図のような構造を表現しています。



このスキーマファイルに基づいた XML ファイルの例として、XML ディレクトリに Order.xml という XML ファイルがあります。

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<Orders>

  <Order>
    <OrderNumber>100</OrderNumber>
    <OrderDate>2005-12-01</OrderDate>
    <CustomerCode>7</CustomerCode>
    <Order_Total>7187.4</Order_Total>
    <DiscountPercent>15</DiscountPercent>
    <InvoiceNumber>1</InvoiceNumber>
    <MethodofPayment>Credit Card</MethodofPayment>
    <OrderLines>
      <Line>
        <LineNumber>1</LineNumber>
        <DepartureLocationCode>12</DepartureLocationCode>
        <DestinationLocationCode>6</DestinationLocationCode>
        <AirlineCode>6</AirlineCode>
        <DepartureDate>2006-03-20</DepartureDate>
        <DepartureTime>12:30:04</DepartureTime>
        <LandingDate>2006-03-21</LandingDate>
        <LandingTime>01:01:30</LandingTime>
        <FlightCost>2400</FlightCost>
      </Line>
      <Line>
        <LineNumber>2</LineNumber>
        <DepartureLocationCode>9</DepartureLocationCode>
        <DestinationLocationCode>13</DestinationLocationCode>
        <AirlineCode>6</AirlineCode>
        <DepartureDate>2006-04-20</DepartureDate>
        <DepartureTime>03:45:00</DepartureTime>
        <LandingDate>2006-04-21</LandingDate>
        <LandingTime>23:04:09</LandingTime>
        <FlightCost>2998.1</FlightCost>
      </Line>
      <Line>
        <LineNumber>3</LineNumber>
        <DepartureLocationCode>16</DepartureLocationCode>
        <DestinationLocationCode>12</DestinationLocationCode>
        <AirlineCode>9</AirlineCode>
        <DepartureDate>2006-05-09</DepartureDate>
        <DepartureTime>10:02:00</DepartureTime>
        <LandingDate>2006-05-09</LandingDate>
        <LandingTime>15:06:00</LandingTime>
        <FlightCost>1789.3</FlightCost>
      </Line>
    </OrderLines>
  </Order>

  <Order>
    <OrderNumber>101</OrderNumber>
    <OrderDate>2006-01-10</OrderDate>
    <CustomerCode>9</CustomerCode>
    <Order_Total>2320.5</Order_Total>
    <DiscountPercent>30</DiscountPercent>
    <MethodofPayment>Credit Card</MethodofPayment>
    <OrderLines>
      <Line>
        <LineNumber>1</LineNumber>
        <DepartureLocationCode>16</DepartureLocationCode>
        <DestinationLocationCode>3</DestinationLocationCode>
        <AirlineCode>9</AirlineCode>
        <DepartureDate>2005-12-29</DepartureDate>
        <DepartureTime>01:55:00</DepartureTime>
        <LandingDate>2005-12-30</LandingDate>
        <LandingTime>04:00:00</LandingTime>
        <FlightCost>456</FlightCost>
      </Line>
      <Line>
        <LineNumber>2</LineNumber>
        <DepartureLocationCode>3</DepartureLocationCode>
        <DestinationLocationCode>4</DestinationLocationCode>
        <AirlineCode>1</AirlineCode>
        <DepartureDate>2006-03-01</DepartureDate>
        <DepartureTime>23:45:00</DepartureTime>
```

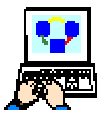
```
          <LandingDate>2006-03-02</LandingDate>
          <LandingTime>04:21:00</LandingTime>
          <FlightCost>689.5</FlightCost>
        </Line>
        <Line>
          <LineNumber>3</LineNumber>
          <DepartureLocationCode>4</DepartureLocationCode>
          <DestinationLocationCode>14</DestinationLocationCode>
          <AirlineCode>1</AirlineCode>
          <DepartureDate>2006-03-14</DepartureDate>
          <DepartureTime>02:30:00</DepartureTime>
          <LandingDate>2006-03-14</LandingDate>
          <LandingTime>08:06:00</LandingTime>
          <FlightCost>769</FlightCost>
        </Line>
        <Line>
          <LineNumber>4</LineNumber>
          <DepartureLocationCode>14</DepartureLocationCode>
          <DestinationLocationCode>15</DestinationLocationCode>
          <AirlineCode>4</AirlineCode>
          <DepartureDate>2006-04-01</DepartureDate>
          <DepartureTime>10:50:00</DepartureTime>
          <LandingDate>2006-04-01</LandingDate>
          <LandingTime>13:32:00</LandingTime>
          <FlightCost>406</FlightCost>
        </Line>
      </OrderLines>
    </Order>

    <Order>
      <OrderNumber>102</OrderNumber>
      <OrderDate>2005-12-23</OrderDate>
      <CustomerCode>90</CustomerCode>
      <Order_Total>700</Order_Total>
      <DiscountPercent>35</DiscountPercent>
      <InvoiceNumber>0</InvoiceNumber>
      <MethodofPayment>Credit Card</MethodofPayment>
      <OrderLines>
        <Line>
          <LineNumber>1</LineNumber>
          <DepartureLocationCode>4</DepartureLocationCode>
          <DestinationLocationCode>1</DestinationLocationCode>
          <AirlineCode>12</AirlineCode>
          <DepartureDate>2006-06-10</DepartureDate>
          <DepartureTime>10:06:00</DepartureTime>
          <LandingDate>2006-06-10</LandingDate>
          <LandingTime>16:00:00</LandingTime>
          <FlightCost>700</FlightCost>
        </Line>
      </OrderLines>
    </Order>

</Orders>
```

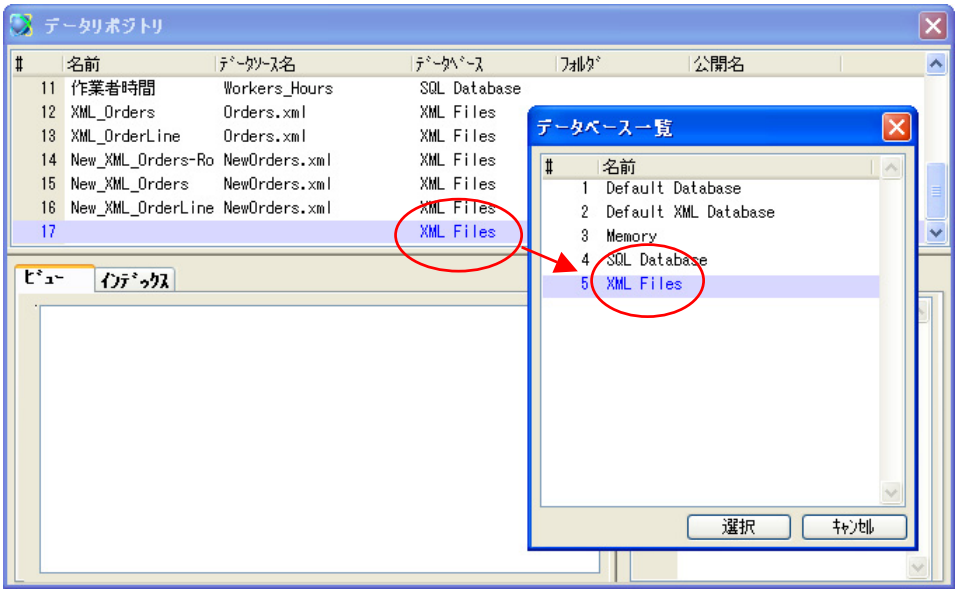
XSD から定義取得

XML データソースの定義は、データリポジトリに スキーマファイルから定義取得する、という形で行います。
TravelAgent のプロジェクトを開き、次のようにして定義取得します。

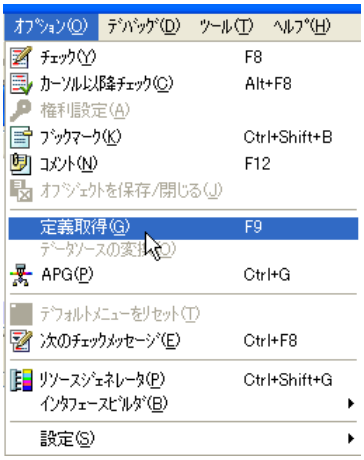


XSD から定義取得するには・・・

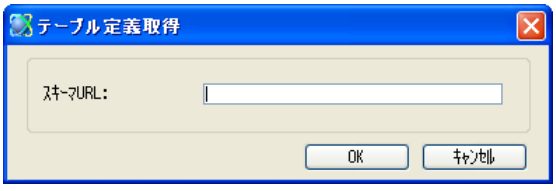
1. データリポジトリに新しい行を追加します。
2. データベース 欄で **XML ファイル** を選択します。



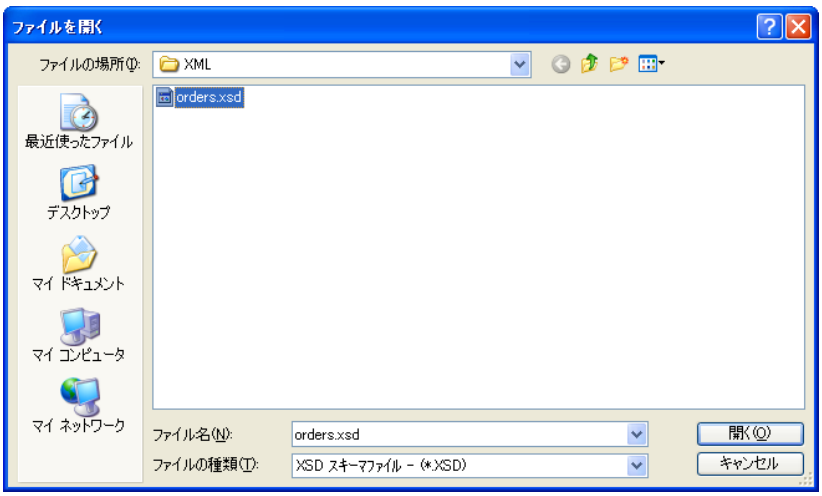
3. メニュー **オプション(O)** → **定義取得(G)** を選びます。



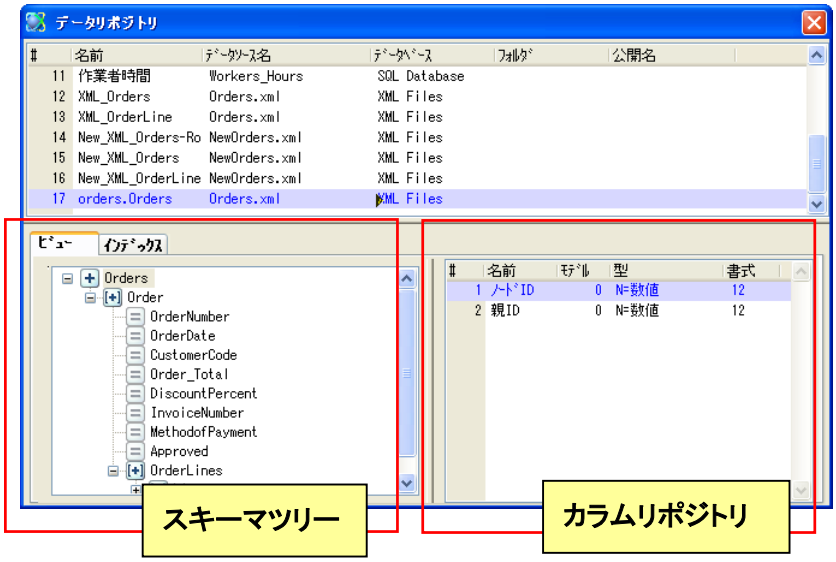
4. テーブル定義取得 ダイアログが開きます。



5. スキーマ URL をズームすると、**ファイルを開く** ダイアログが出ますので、スキーマファイル名を指定します。



6. **テーブル定義取得** ダイアログに戻るので、ここで OK ボタンを押せば、スキーマが解析され、データリポジトリに定義内容が設定されます。

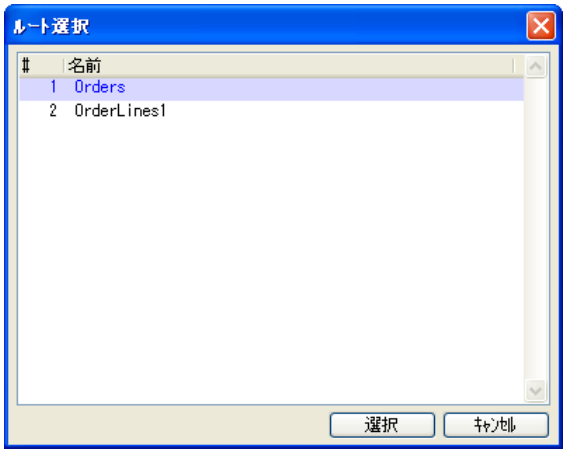


XML ファイルの場合、データリポジトリの画面の左下は **スキーマツリー** 画面、右下は **カラムリポジトリ** と呼ばれます。

ルート要素が複数ある場合

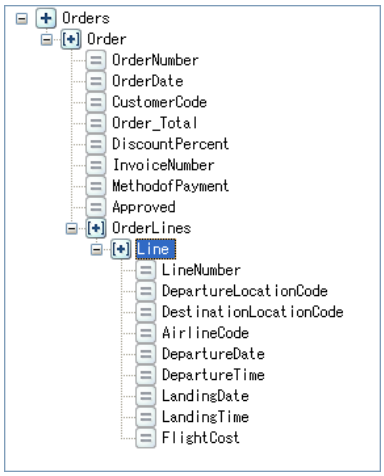
サンプルプロジェクトに添付されているスキーマファイル Orders.xsd は、要素が一つしか定義されていないので、その要素 Orders がデフォルトでルート要素となりましたが、一般には、一つのスキーマファイルに複数の要素が定義されていますので、定義取得時にどの要素がルート要素なのかを、開発者が指定してやる必要があります。

このような場合には、定義取得時に、次のように、スキーマファイル中に定義されている要素名の一覧が表示されます。開発者はこのうちから、ルート要素とする要素を選択します。



11.2.2. スキーマツリーの表示

スキーマツリーは、XSD に定義されている構造をツリー形式で表示しているものです。



ノードの種類

スキーマツリーのノードは、XSD の定義に従って、以下のような種類の表示がされます。

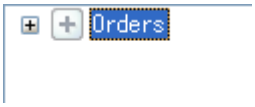
要素の種類	通常状態	選択不可状態	選択済状態
単純要素			
複合要素			
繰り返し可能な単純要素			
繰り返し可能な複合要素			

ノードの展開と縮小

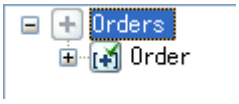
サブツリーのあるノードは**展開・縮小**ができます。

スキーマツリーのノードの展開・縮小の方法

- 縮小されている(閉じている)ノードは、左側に ⊞ 記号が表示されます。すべて閉じている状態では、右図のように表示されます。

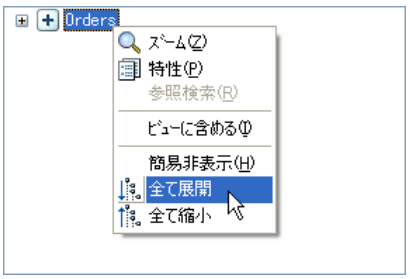


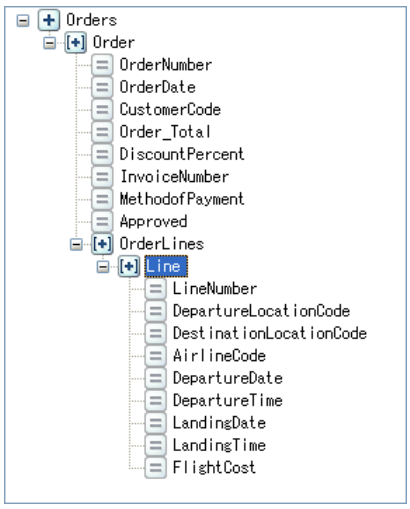
- ⊞ 記号をマウスでクリックすると、ノードが展開されます。展開されている(開いている)ノードは、左側に ⊟ 記号が表示されます。1レベルだけ開いた状態では、右図のように表示されます。
  逆に、⊟ 記号をマウスでクリックすると、ノードが縮小されます。



- コンテキストメニューから、すべてのノードを展開、あるいは縮小することもできます。

  すべてのノードが展開された状態では、右図のように表示されます。



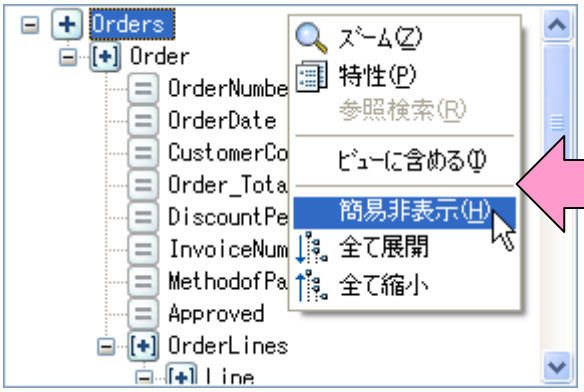


簡易表示

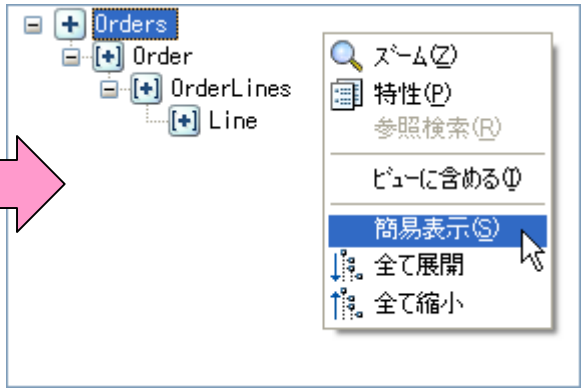
XML の構造は、ネストの深い複雑な構造になることがありますが、このときにはスキーマツリーが大きなものになって、データリポジトリ上で全体を見るのが難しくなります。このような場合には、**簡易表示** で表示すると、全体を見やすくなります。

簡易表示 にするには、スキーマツリー上でコンテキストメニューから **簡易非表示** を選びます。簡易非表示の状態では、単純要素が表示されなくなります。

簡易表示状態



簡易非表示状態



11.2.3. データ ソース ビューを定義する

XML ファイルは階層的な構造を持ったデータですが、Magic のデータソースは、リレーショナルデータベースのテーブルのように、2 次元のテーブル形式のデータです。このため、XML ファイルのデータ構造を、2 次元のテーブル形式にマッピングする必要があります。

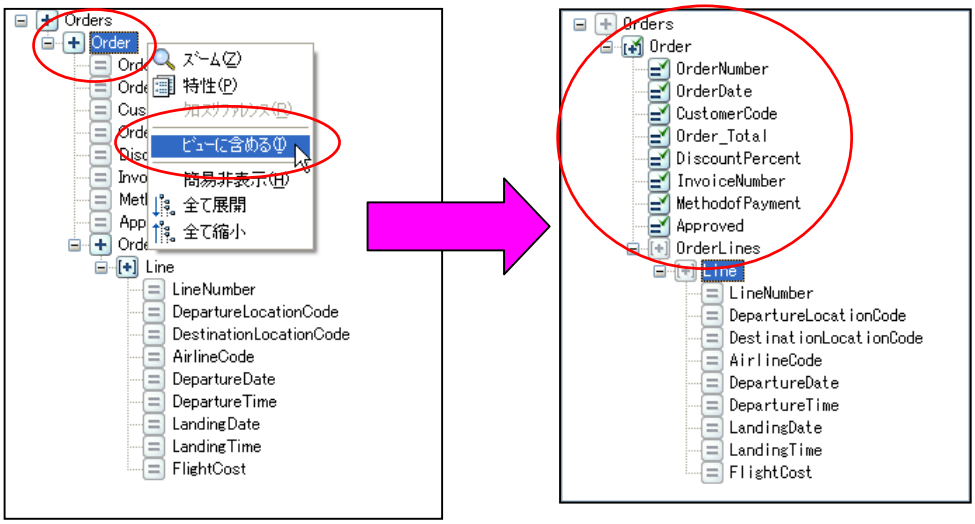
データのマッピングは、データソースのビューにどの要素を含めるかを指定することにより行われます。

要素をビューに含める

XML ファイルの特定の要素をデータソース ビューに含めたい場合には、スキーマツリー 画面でコンテキストメニューを出し、**ビューに含める** を選択します。

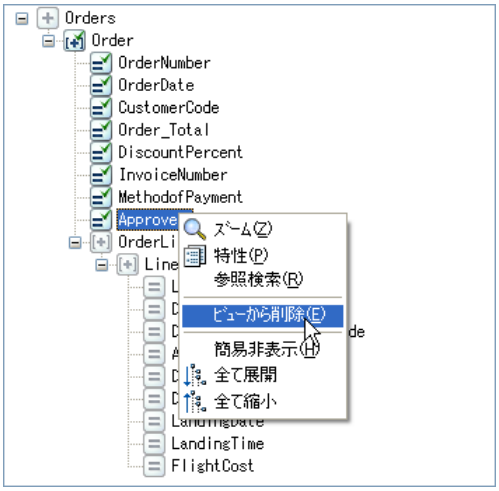
例えば、Order およびその直下の要素をビューに含めたい場合には、Order ノードでコンテキストメニューを出し、**ビューに含める** を選びます。

ビューに含むように選択されたノードは、 や などのように、チェック記号がつきます。



ビューから削除

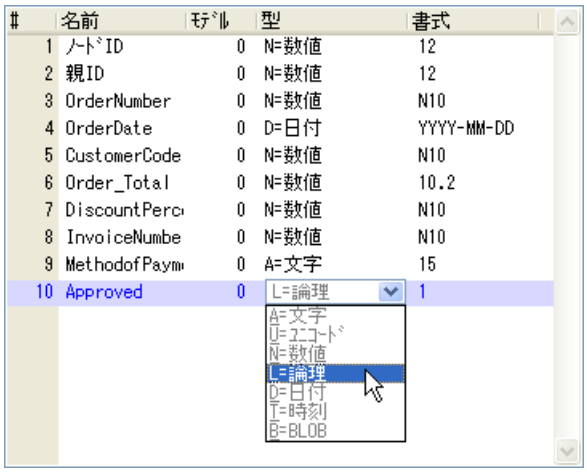
ビューから特定の要素を除きたい場合には、スキーマツリー上で、除きたいノードにカーソルを合わせて、コンテキストメニューを開き、**ビューから削除(E)** を選択します。



カラムリポジトリ

画面右下の **カラムリポジトリ** には、選択されたノードに対応して、カラムが追加されます。

各カラムのデータ型および書式は、XSD の type= 属性をもとにして、デフォルトの値が設定されます。開発者は、モデル、型、書式の設定を変更することができますが、名前を変更することはできません。



ただし、カラムリポジトリの先頭の2要素は、**ノード ID** と **親 ID** で固定となっていて、開発者が変更・削除することはできません。

- **ノード ID** は、XML データ構造の特定のノードを一意に識別する内部番号です。この値は Magic 実行エンジン内部で作成されるもので、Magic プログラムから修正・作成することはできません。また、この値は、この XML ファイルをオープンしてからクローズするまでは一定ですが、再度オープンすると別の値になっている可

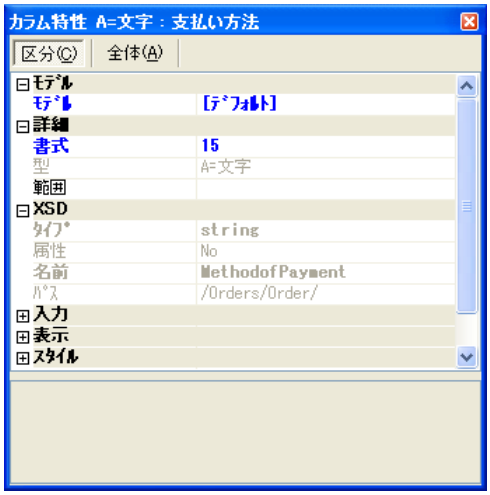
能性があります。従って、ノード ID は、XML ファイルがオープンされている間だけ有効な、内部識別番号として扱ってください。

- **親 ID** は、XML のデータ構造において、現在のノードの親ノードの、ノード ID です。この値も Magic 実行エンジン内部で生成されるものです。ノード ID と同じく、これも XML ファイルがオープンされている間だけ有効な、内部識別番号です。

カラム特性

XML データソースのカラムの特性には、以下のような、DBMS のカラム特性にはない特性があります。これらの特性は、**XSD** セクションの下に格納されており、読込専用で、開発者が修正することはできません。

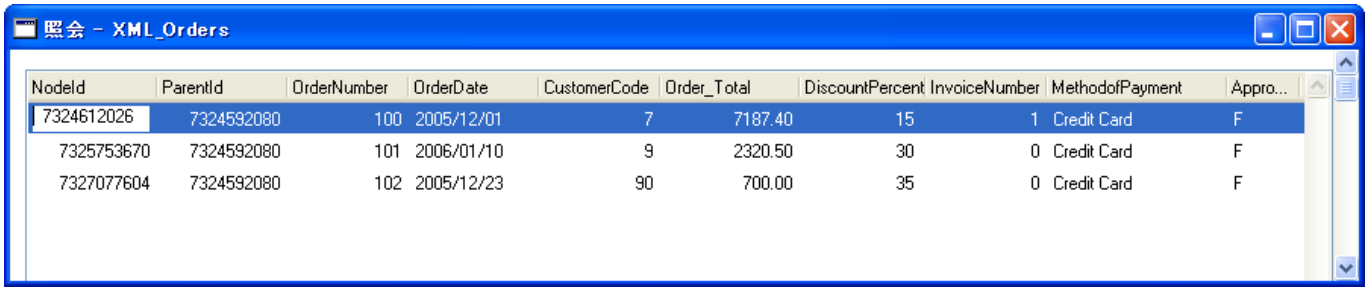
右図は、MethodofPayment カラムのカラム特性の例です。



特性名	説明
タイプ	このカラムに対応する XML 要素の、XSD タイプを示します。
属性	Yes の場合、このカラムが、XML の要素の属性に対応することを意味します。No の場合には、XML の要素に対応することを意味します。
名前	XSD 中での名前です。
パス	このカラムを取得するための、XML のパスです。

カラム特性は、単純要素の場合にだけ表示されます。複合要素上にカーソルがある場合には表示されません。

以上定義したデータソースを APG で開くと、次のようなテーブルとして見ることができます。

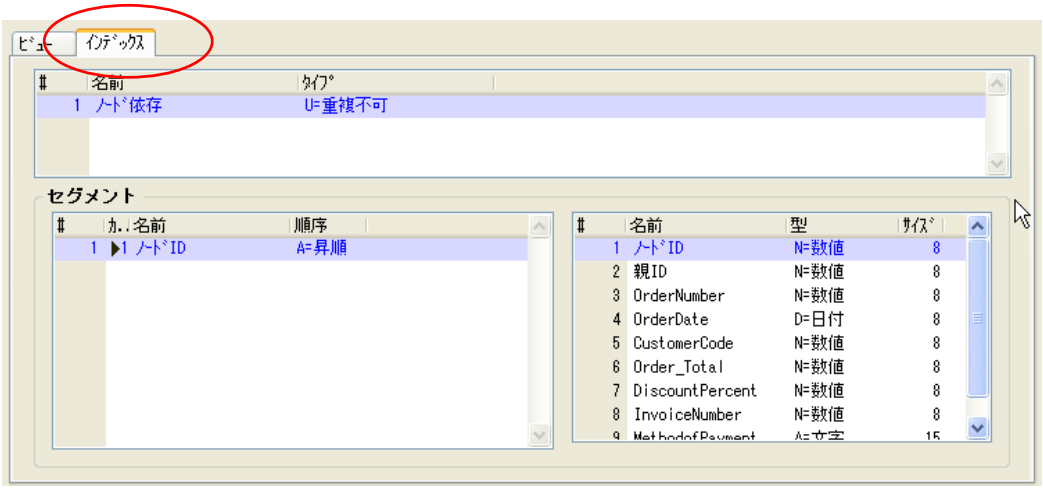


これを見てわかるように、このデータソースでは、Order レベルの要素のみが表示され、明細行（OrderLines 以下)はビューに含められていません。

11.2.4. インデックス定義

XML データベースにも、DBMS データベースと同様、インデックスを定義することができます。

インデックスの定義は、**インデックス** タブで表示・定義されます。



デフォルトで、**ノード依存** という名前のインデックスが定義されています。これは、ノード ID をセグメントとするものです。このインデックスは修正・削除することができません。

開発者は、アプリケーションの必要に応じ、インデックスを追加定義することができます。

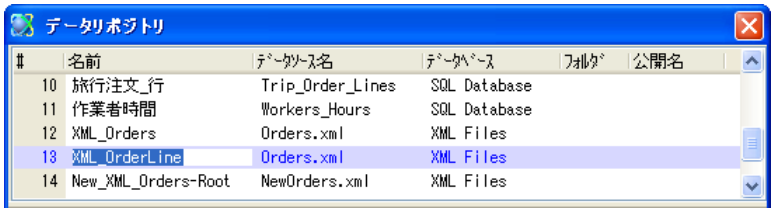
11.2.5. 同じ XML ソースファイルから異なる XML ビューを定義する

XML データは階層的な構造を持つデータですので、異なるレベルのデータを扱うために、異なる XML ビューを定義する必要があります。

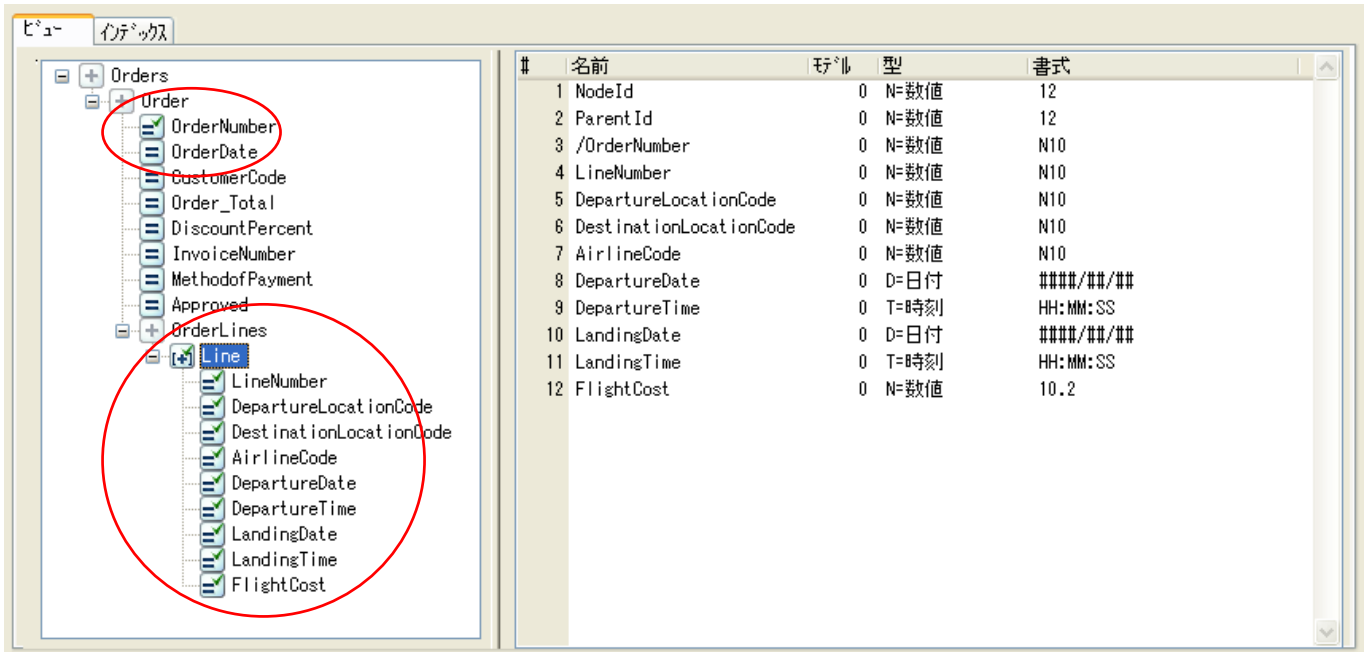
例えば、「11.2.3 データソース ビューを定義する」では、Order.xml というXMLファイルから、Order というタグの直下にある要素(OrderNumber、OrderDate など)を扱いましたが、OrderLines 以下の明細行については扱うことができませんでした。

明細行を Magic で取り扱えるようにするには、データリポジトリで、同じ Order.xml という XML ファイルから、OrderLines 以下の要素を含む XML ビューを定義してやります。

データリポジトリを開いて、データソース 13 番 XML_OrderLine を見てください。これは、データソース 12 番の XML_Orders と同じく、Orders.xml というXMLファイルに対応するデータソースです。



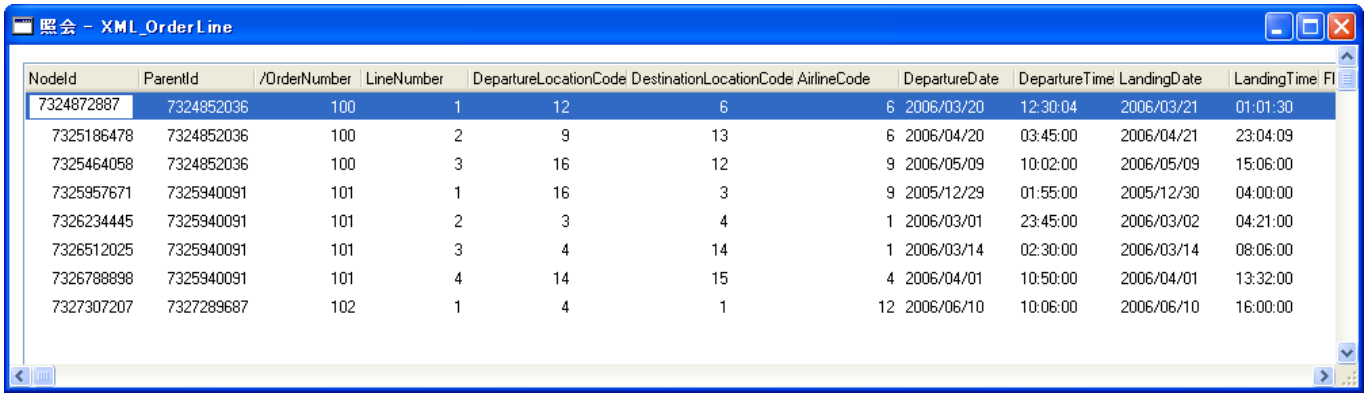
次にデータソース 13 番のビューを示します。



これを見てわかるように、データソース 13 番 XML_OrderLine では、Line とその直下の要素がビューに含められています。従って、明細行(Line 要素以下)を Magic で扱う場合には、データソース 13 番を扱います。

ここで注意すべきことは、Line は Order に従属する要素であるので、各 Line は、どの Order に従属するのかを識別できるようになっていなければなりません。そのために、データソース 13 番 XML_OrderLine のビューには、Order レベルの OrderNumber が識別のためにビューに加えられています。

このように定義されたデータソース 13 番で、APG を実行すると、以下のようなデータが表示されます。



NodeId	ParentId	/OrderNumber	LineNumber	DepartureLocationCode	DestinationLocationCode	AirlineCode	DepartureDate	DepartureTime	LandingDate	LandingTime	Fl
7324872887	7324852036	100	1	12	6	6	2006/03/20	12:30:04	2006/03/21	01:01:30	
7325186478	7324852036	100	2	9	13	6	2006/04/20	03:45:00	2006/04/21	23:04:09	
7325464058	7324852036	100	3	16	12	9	2006/05/09	10:02:00	2006/05/09	15:06:00	
7325957671	7325940091	101	1	16	3	9	2005/12/29	01:55:00	2005/12/30	04:00:00	
7326234445	7325940091	101	2	3	4	1	2006/03/01	23:45:00	2006/03/02	04:21:00	
7326512025	7325940091	101	3	4	14	1	2006/03/14	02:30:00	2006/03/14	08:06:00	
7326788898	7325940091	101	4	14	15	4	2006/04/01	10:50:00	2006/04/01	13:32:00	
7327307207	7327289687	102	1	4	1	12	2006/06/10	10:06:00	2006/06/10	16:00:00	

OrderNumber カラムは、名前が **/OrderNumber** となっており、先頭にスラッシュ文字「/」が付けられていますが、これは上のレベルの要素であることを示しています。

このようにデータリポジトリに定義しておけば、Magic プログラムからは、通常の DBMS と同じようにテーブルとして扱うことができ、Magic プログラムでレコードの追加、修正、削除などを行った場合には、対応する XML ファイル上でもデータが追加、修正、削除されます。

11.3. XML ファイルの作成

XML ファイルや、XML レコードを作成する場合には、通常の DBMS テーブル上に定義されるデータソースにはない、以下のような注意が必要になります。

- XML ビューで定義したデータソース名で定義される XML ファイルが存在しなかった場合、Magic はその XML ファイルを作成しますが、そのノードまでは作成しません。従って、XML のルートとなる要素は、必ず一つだけ、プログラムで作成しておかなければなりません。
- XML は構造を持ったデータですので、従属する要素(明細行のデータなど)は、必ず上位の要素が存在している状態でなければ、作成することができません。従って、同一 XML ファイルを別の XML ビュー定義により、別のデータソースとして定義した場合にも、まず、上位のノードを作成してから、下位のノードを作成していく、という順序でレコード追加していく必要があります。この制約は、DBMS で参照制約が設定されている場合と似ています。
- 下位ノードがどの上位ノードに従属する要素であるかがわかるように、上位ノードの要素(「/OrderNumber」など)に、値を設定する必要があります。

- 親子タスクで、上位ノード、下位ノードを扱うようなタスクの構造にする場合、上位ノードのレコードが作成されてから下位ノードのレコードが作成されるようにするには、子タスクを呼び出す場合のコールコマンドで、**同期** パラメータを Yes にしておくと便利です。
- **同期** = Yes を利用するには、遅延トランザクションを利用することが必要になります。

11.4. XML 関数

XML データソースを通して XML ファイルを作成・更新する場合、Magic は XML ファイル構造を、対応するスキーマファイル XSD に照らし合わせて、正当性を検証します。しかし、一部の XSD 定義（制約条件など）については検証がされません。
XML データ構造の正当性を正確に検証したい場合には、組み込み関数 XMLValidate を使うことができます。

11.4.1. XMLValidate

XMLValidate 関数は、スキーマ定義に照らし合わせて XML データを検証します。スキーマ定義は XML データ自身の中に、schemaLocation あるいは noNameSpaceSchemaLocation を用いて参照先を定義しておくこともできるし、あるいは、XMLValidate 関数の第二引数として URI を指定することもできます。

シンタックス XMLValidate (XML [, XSD schema URI])

パラメータ
1. XML: XML データを格納している BLOB 値
2. XSD schema URI: スキーマ定義ファイルを参照する URL (http またはファイルシステム)。このパラメータはオプションです。指定されない場合には、XML ファイル中の SchemaLocation 指定が参照されます。指定された場合には、XML ファイル中の SchemaLocation 指定は無視され、こちらが優先されます。

戻り値 XML の検証が正しく行われれば、真が返ります。エラーの場合には偽が返ります。

参考
- スキーマが指定されなければ、XML ファイルは正当であると解釈されます。
- 警告のみが見つかった場合には、XML ファイルは正当であると解釈されます。

例
```
XMLValidate (File2Blb('C:¥eDeveloper2¥XML¥Orders.XML'),
    'C:¥eDeveloper2¥XML¥Orders.XSD')
```

11.4.2. XMLValidationError

XMLValidate 関数を実行したときにエラーが見つかった場合、エラー内容を Unicode 型のベクトル値として返します。

シンタックス XMLValidationError ()

戻り値 エラーの詳細を説明する Unicode 文字列のベクトル。ベクトルの各要素には、XML パーサ(Xerces)が検出したエラーメッセージが格納されています。

12. UNICODE

Unicode というのは、現在世界で使われている文字をコンピュータ上で統一的に扱うことを目的とした文字コード体系です。

Unicode 以前では、コンピュータ上では言語ごとに異なった文字コード体系が使われており、複数の異なる言語の文字を一つのファイルや文字データで混在させて利用することができませんでした。しかしながら、ビジネスや文化の国際化が進み、一つのデータ内で複数の言語を利用することが強く求められてきた結果、Unicode が国際標準として制定されました。

現在においても、コンピュータ上では、言語地域ごとの文字コード体系と、Unicode とが混在されて利用されているのが現状です。

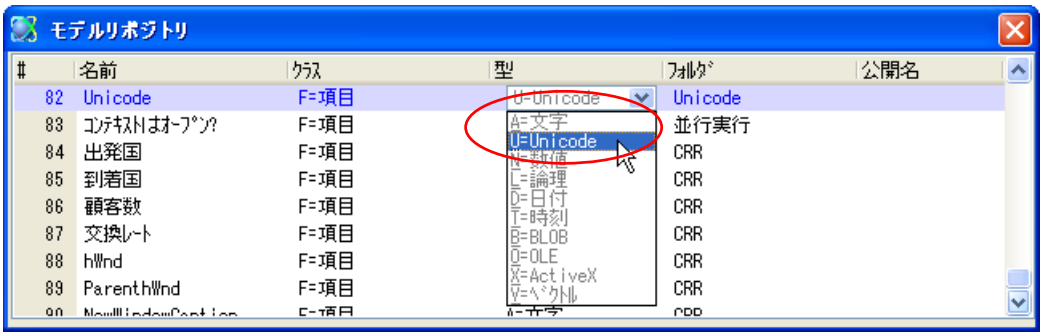
Magic においても、V9Plus までは、ANSI/OEM 文字コード（日本語の場合には Shift-JIS コード)のみを扱っていましたが、対象アプリケーションのビジネスが国際化していく流れを踏まえて、Unicode サポートを V10 で導入し、従来の ANSI/OEM との互換性を保ちながら、Unicode も利用できるようになりました。本章では、V10 で新しく追加された Unicode サポート機能について説明していきます。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

Unicode 全般	**Unicode サポート** の章
外部コードページの設定	**設定 → 動作環境 → [国別設定]タブ → 外部コードページ**
Unicode モデル	**モデル → データ項目 → 型 → Unicode 型**
データソースの Unicode カラム	**データソース → データソースタイプ → テーブルソース → カラムテーブル → 型**
プログラムでの Unicode 項目	**データビューエディタ → 項目特性 → 型**
プログラムの入出力ファイル	**プログラム → 入出力ファイル → 入出力特性 → 使用する文字セット**
変換用関数	**式エディタ → 関数ディレクトリ** 以下の UnicodeChr, UnicodeFromANSI, UnicodeToANSI, UnicodeVal 関数
メニューリポジトリでの Unicode	**メニューとヘルプ → メニュー → メニュー定義テーブル → ユーザメニュー名**
ヘルプリポジトリでの Unicode	**メニューとヘルプ → ヘルプ → ヘルプと Unicode**

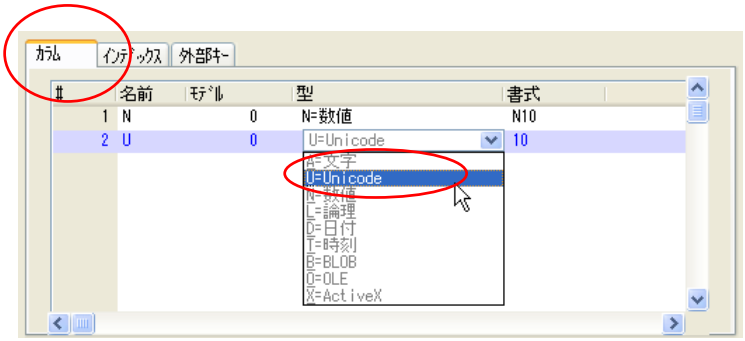
12.1. Unicode 型のモデル・項目を定義する

Unicode 型のモデルは、モデルリポジトリで **クラス** は **F=項目**、**型** は **U=Unicode** を選択することで定義することができます。

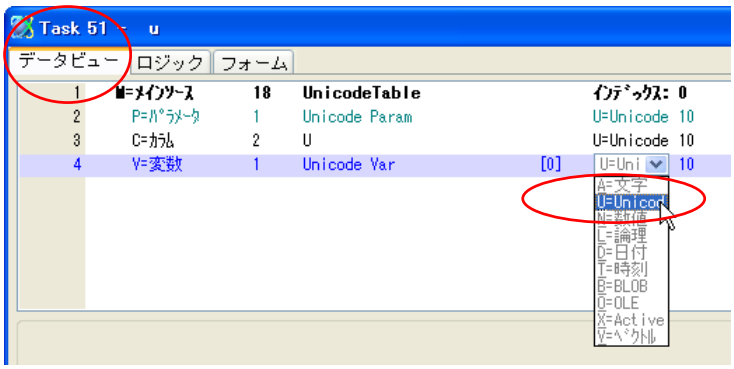


モデルリポジトリのほかに、Unicode 型のデータは、次の箇所でも定義することができます。

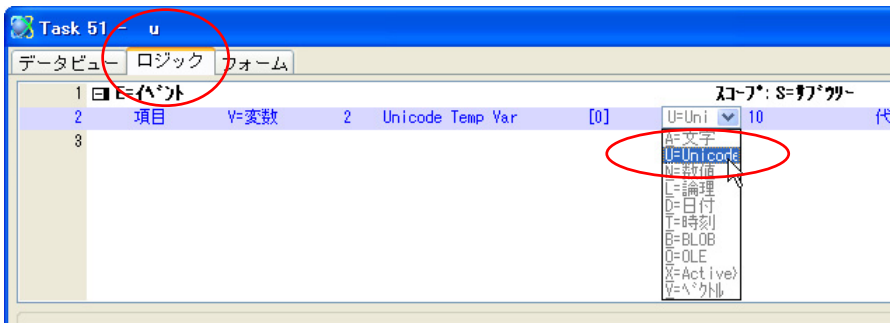
- データリポジトリ の カラム



- プログラムリポジトリのデータビュー タブの **V=変数** や **P=パラメータ** 行



- プログラムリポジトリの **ロジック** タブの **項目** 行



12.2. Unicode 型モデル・項目の特性

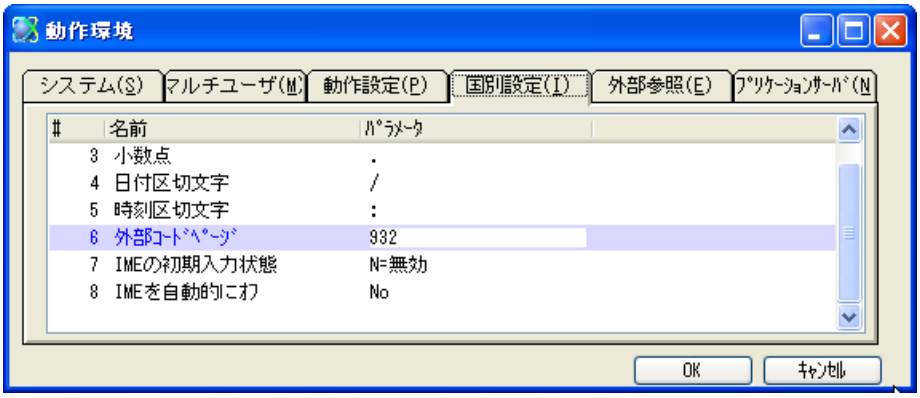
Unicode 型のモデル・項目には、右図のような特性があります。Unicode 型の特性は、文字型の項目とほとんど同じです。

Unicode 型の **記憶形式** は、文字型と異なり、次の二つが定義できます。

- **Unicode**： 固定長の Unicode 文字列。データベースには、各 DBMS がサポートする固定長 Unicode のカラムとして定義されます。
- **ZUnicode**： 最後が NULL 文字で終端されている可変長 Unicode 文字列。各 DBMS がサポートする可変長 Unicode のカラムとして定義されます。

12.3. 変数間でのデータ変換

Unicode 型と文字型データとは混在して利用することができます。項目への代入、式の評価などで必要な場合には、内部で自動的に文字コード変換が行われます。この際、文字型データのコードページとしては、動作環境 → 動作設定 タブ → 外部コードページ パラメータに設定されているコードページが利用されます。日本語の場合には インストーラによりデフォルトで 932 が設定されています。



利用可能なコードページの一覧は、「付録 A. 利用可能なコードページの一覧」を参照してください。

12.4. データ変換用関数

V10 では、Unicode 型と文字型との間のデータ変換を明示的に行う組み込み関数が用意されています。

文字型項目から Unicode 型項目、または Unicode 型項目から文字型項目への代入、または項目更新処理を行う場合、内部で変換を行いますので、開発者が明示的に UnicodeFromANSI 関数や UnicodeToANSI 関数を使う必要はありません。

12.4.1. UnicodeFromANSI

シンタックス UnicodeFromANSI (*ANSI String*, *Code Page*)

パラメータ ANSI String – 文字型のデータ。
Code Page – 第一パラメータの文字データの文字コードを表すコードページ値(整数)。この値が 0 の場合には、システムデフォルトのコードページが使われます。

戻り値 変換後の Unicode 文字列が返されます。
ANSIString が Null だったり、コードページ値が不正だった場合には、NULL 値が返されます。

12.4.2. UnicodeToANSI

シンタックス UnicodeToANSI (*Unicode String*, *Code Page*)

パラメータ Unicode String – Unicode 型のデータ。
Code Page – 変換後の文字データの文字コードを表すコードページ値(整数)。この値が 0 の場合には、システムデフォルトのコードページが使われます。

戻り値 変換後の文字型データが返されます。
Unicode String が Null だったり、コードページ値が不正だった場合には、NULL 値が返されます。

注意 Unicode String 中に、Code Page で定義されていない文字データが入っていた場合には、変換が正しく行われません。

12.4.3. UnicodeChr

シンタックス UnicodeCHR (*Number*)

パラメータ Number – Unicode 文字のコード値。 0 - 65,535 の間の整数。

戻り値 Number をコード値とする Unicode 文字 (1 文字)が返ります。Unicode のコード値として不正な値が指定された場合には、Null が返ります。

12.4.4. UnicodeVal

シンタックス　UnicodeVAL (Unicode CHR)

パラメータ　Unicode CHR - Unicode 型の文字列。この文字列に 2 文字以上ある場合には、先頭の 1 文字だけが対象となります。

戻り値　与えられた Unicode 文字の文字コードを整数値として返します。
文字列が空の場合には、0 が返ります。また、パラメータの値が Null の場合には、Null が返ります。

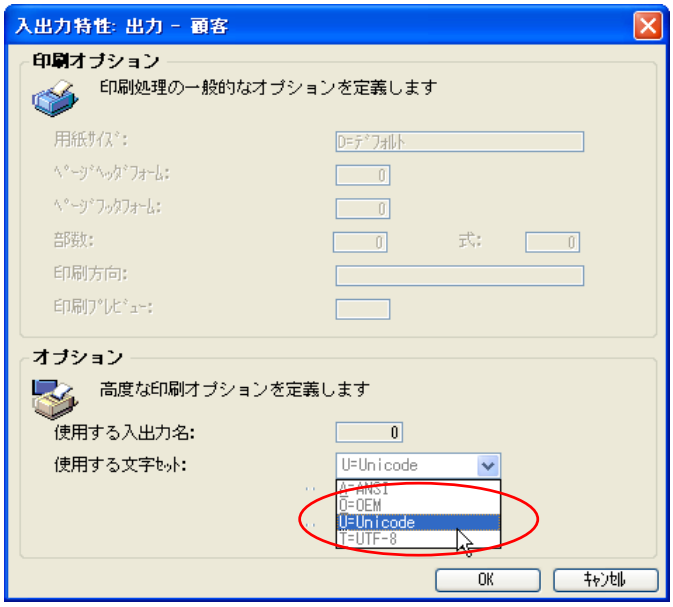
V9Plus の Chr および Val 関数は、ASCIIChr と ASCIIVal と改名されました。V10 で Ansi (Shift-JIS) のコードを扱う場合には、こちらの関数を使ってください。

12.5. Unicode サポート機能

モデル、カラム、変数などのほかに、次のようなところで Unicode 文字を使うことができます。

12.5.1. I/O ファイル

プログラムリポジトリの入出力テーブルで、外部ファイルとのデータ入出力処理を行う際の文字コードを指定することができます。
プログラムを開いて、入出力テーブルを開き、更に入出力特性を開くと、 **使用する文字セット** というパラメータがあり、ここで以下のような文字コード指定を行うことができます。



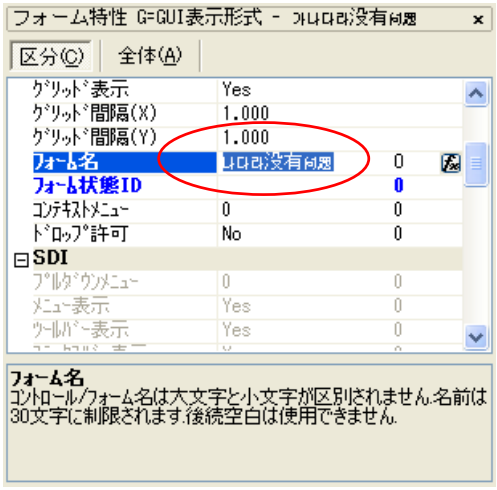
文字コード指定	意味
A=ANSI	ANSI 文字セットを使います。日本語の場合には、Shift-JIS です。
O=OEM	プラットフォームに依存した文字セットを使います。日本語の場合、PC 上では Shift-JIS ですが、他のハードウェア/オペレーティングシステム上では別の文字セットとなる場合があります。
U=Unicode	Unicode (UTF-16) を使います。
T=UTF-8	UTF-8 を使います。

BOM (Byte Order Mark)文字について

- データ出力時には、BOM が出力されます。
- データ入力時には、BOM があるかどうかがチェックされます。BOM があって、入出力特性の **使用する文字セット** の設定と異なっている場合には、データの読込は失敗します。

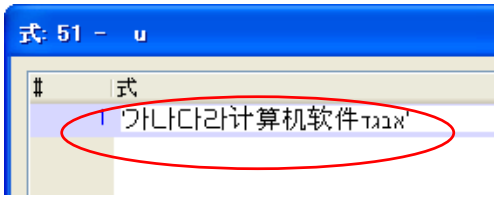
12.5.2. フォーム・コントロール特性

フォームエディタでのフォーム特性やコントロール特性など、タスクのデータ表示に関わる箇所の多くでは Unicode 文字を使うことができます。



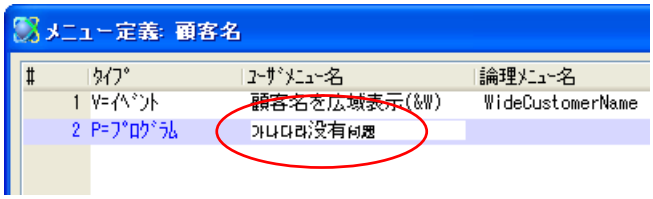
12.5.3. 式

V10 の式エディタは、Unicode ベースで実装され、リポジトリにも Unicode で格納されます。



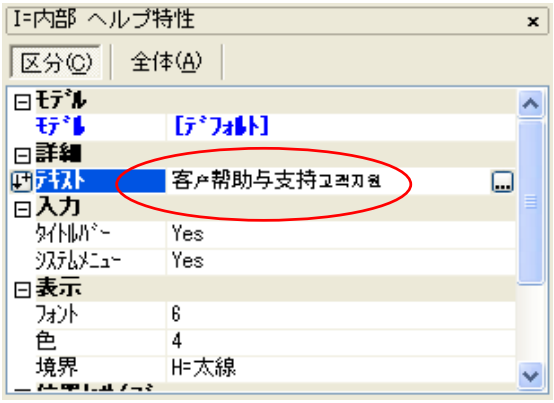
12.5.4. メニュー

メニューリポジトリの ユーザメニュー名 欄には、Unicode 文字を指定することができます。



12.5.5. ヘルプリポジトリ

ヘルプリポジトリのテキストにも、Unicode 文字列を使うことができます。



上記以外のところでは、Shift-JIS にない Unicode 固有のデータを使うことはできません。例えば、次のようなところでは使えません。

- モデル、データソース、プログラム等の各リポジトリ名、
- データソースリポジトリ内のカラム名、インデックス名、外部キー名
- プログラムリポジトリの変数名、パラメータ名、ユーザイベント名、フォーム内のコントロール名、など

12.6. 実行エンジンの動作

V9Plus までの Magic 実行エンジンは、内部データの文字セットとして Shift-JIS を使っていました。V10 からは、内部データとして Unicode を使うようになっています。このため、実行エンジンの動作が変わってくるところが出てきます。
ここでは、V10 でアプリケーションを設計・開発する上で注意すべき点について説明します。

12.6.1. データの照合順序

Shift-JIS と Unicode は、格納される文字コードが異なりますので、それぞれをキー項目としたデータの処理順序が異なることにより、プログラムの範囲抽出処理やバッチのグループ処理に影響がでる可能性があります。
また、SQL データベースを利用する場合は、そのデータベースマネージャ固有の機能や設定による影響があります。
文字コードを比較した場合の文字種類別の順序は次の通りです。

Shift-JIS(文字型項目)のコード順序	Unicode(Unicode 型)のコード順序
1.半角数字	1.半角数字
2.半角英	2.半角英
3.半角カナ	3.ひらがな
4.全角数字	4.カタカナ
5.全角英	5.漢字
6.ひらがな	**6.全角数字**
7.カタカナ	**7.全角英**
8.漢字	**8.半角カナ**

漢字については、個々の漢字のコード順序が異なります。 例えば、「亜」と「一」では Shift-JIS と Unicode で、次のようにコードの順序が異なります。

	亜	一	比較結果
Shift-JIS	0x889F	0x88EA	亜 が 一 より先
Unicode	U+4E9C	U+4E00	一 が 亜 より先

12.6.2. 関数

文字型項目をパラメータに持つ関数に対して、文字型項目の代わりに Unicode 型項目を設定した場合、次の点に注意が必要です。

一つの文字データをパラメータに持つ関数

Len 関数など、1つの文字列をパラメータにもつ関数の場合、パラメータが文字型か Unicode 型かで、文字列の長さを数える単位が変わります。

#	パラメ	長さの単位
1.	文字型	バイト単位
2.	Unicode 型	文字単位

例えば、

- 項目 A が文字型項目で 'あ' を格納した場合、Len(RTrim(A))の結果は 2 を返します。
- 項目 B が Unicode 型項目で 'あ' を格納した場合、Len(RTrim(B))の結果は 1 を返します。

一つの文字パラメータを持ち、戻り値も文字データの場合

MID 関数など、一つの文字データをパラメータに持ち、戻り値も文字データの関数については、文字データの長さの単位は、

1. パラメータの型、
2. 戻り値の型

の順で決定されます。

#	パラメータの型	戻り値の型	文字データの長さの単位
1.	文字型	文字型	バイト単位
2.	文字型	Unicode 型	バイト単位
3.	Unicode 型	文字型	文字単位
4.	Unicode 型	Unicode 型	文字単位
5.	固定値 (*1)	文字型	バイト単位
6.	固定値 (*1)	Unicode 型	文字単位

*1 式に直接、'あいうえお' などの文字列を指定する場合

2つ以上の文字データをパラメータに持つ関数

REP関数（文字列中の位置と文字数を指定し、その部分を別の文字列に置き換える関数）では、パラメータに二つの文字データがあり、戻り値も文字データです。

このような関数の場合、文字列の長さを数える単位は、次の順に決定されます。

1. 二つのパラメータの型
2. 戻り値の型

#	パラメータ 1 の型	パラメータ 2 の型	戻り値の型	長さの単位
1.	文字型	文字型	文字型	バイト単位
2.	文字型	文字型	Unicode 型	バイト単位
3.	文字型	Unicode 型	文字型	バイト単位
4.	文字型	Unicode 型	Unicode 型	文字単位
5.	Unicode 型	文字型	文字型	バイト単位
6.	Unicode 型	文字型	Unicode 型	文字単位
7.	Unicode 型	Unicode 型	文字型	文字単位
8.	Unicode 型	Unicode 型	Unicode 型	文字単位
9.	文字型	値(*1)	文字型	バイト単位
10.	文字型	値(*1)	Unicode 型	バイト単位
11.	Unicode 型	値(*1)	文字型	文字単位
12.	Unicode 型	値(*1)	Unicode 型	文字単位
13.	値(*1)	文字型	文字型	バイト単位
14.	値(*1)	文字型	Unicode 型	バイト単位
15.	値(*1)	Unicode 型	文字型	文字単位
16.	値(*1)	Unicode 型	Unicode 型	文字単位

*1 式に直接、'あいうえお'などの文字列を指定する場合

12.6.3. その他の制限事項

V10 での Unicode 対応については、その他にも細かな制限事項があります。また、データベースゲートウェイにもデータベースに固有な制限事項があります。これらの制限事項については、Readme や弊社ホームページの技術情報などを参照して、確認してください。

13. 開発エンジンと実行エンジンの分離

V9Plus までは、開発版 mggenw.exe には開発機能と実行機能と両方が備わっており、ひとつのプロセスで開発とテスト実行とを行っていました。
V10 では、開発機能のみを持つ開発エンジンと、実行機能のみを持つ実行エンジンとが分離されました。分離することにより、次のような利点が生まれます。

- 実行エンジンがなんらかの理由で異常終了した場合でも、開発エンジンはそのまま動作しています。これにより、異常終了によるアプリケーションの破損などの問題が起こらなくなります。
- プログラムミスなどで実行エンジンが止まらなくなってしまった場合などには、実行エンジンのみを強制終了させることができるようになります。
- 開発エンジンと実行エンジンとは別ウィンドウ上で表示されるようになるので、デバッガの取り扱いが容易になります。(V10 で強化されたデバッガについては、15 章デバッガ で説明します。)
- V9Plus では、開発版での実行機能と、実行版あるいはサーバ版での実行機能とで、微妙に動作が異なる場合がありましたが、V10 では実行は常に実行エンジンで行われるので、動作の違いがなくなります。

本章では、V10 での開発エンジンと実行エンジンの分離について説明していきます。

開発エンジンと実行エンジンとは、次に示すような実行可能モジュールです。下表では、比較のため、V9Plus でのモジュールファイル名も示しておきました。これらのファイルは、いずれも Magic をインストールしたディレクトリにあります。

モジュール区分	V10 モジュール名	V9Plus モジュール名	備考
開発エンジン	eDevStudio.exe	mggenw.exe	V9Plus までは、開発機能・実行機能両方が一緒になっていた。
実行エンジン	eDevRTE.exe	mgrntw.exe	

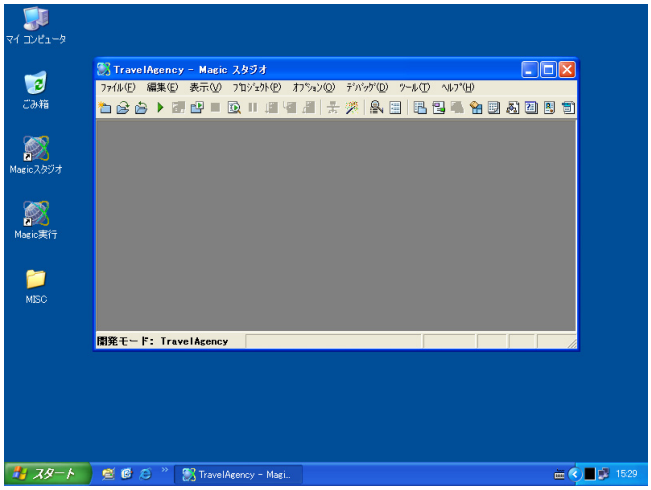
本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

開発エンジンと実行エンジンの分離	**アプリケーションのテスト → 開発エンジンと実行エンジンの分離**
デバッガ	**アプリケーションのテスト → デバッガ**
キャビネットファイル	**プロジェクトとアプリケーション → プロジェクト開発 → プロジェクトのキャビネットファイル**
アプリケーションの実行	**プロジェクトとアプリケーション → アプリケーション実行**
アプリケーションの動作環境	**設定 → 動作環境 → [システム]タブ**

13.1. 開発エンジンでの操作

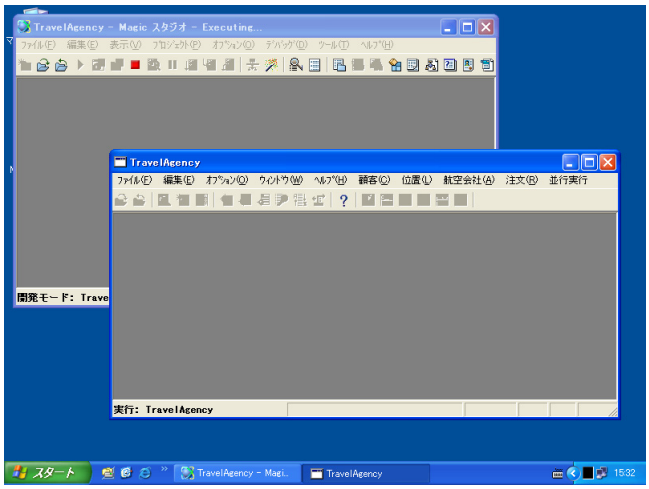
13.1.1. 開発エンジンからプロジェクトを実行する

開発エンジン（Magic スタジオ)を起動し、プロジェクトを開いた状態では、画面には開発エンジンのウィンドウだけが表示されています。
このときに実行エンジンは、表示はされていませんが、背後ですでに起動されています。



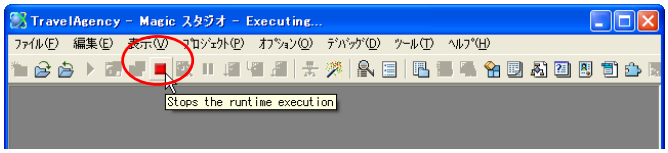
現在開いているプロジェクトを実行するには、V9Plus までは実行モードにするために Ctrl+T を押していましたが、V10 では、**デバッグ メニュー → プロジェクトの実行**（あるいは、**Ctrl+F7** キー）を選択します。

すると、背後で隠れていた実行エンジンのウィンドウが表示されるようになります。
下図では、実行エンジンのウィンドウが前面に現れて、開発エンジンのウィンドウが背後になったようすを示しています。



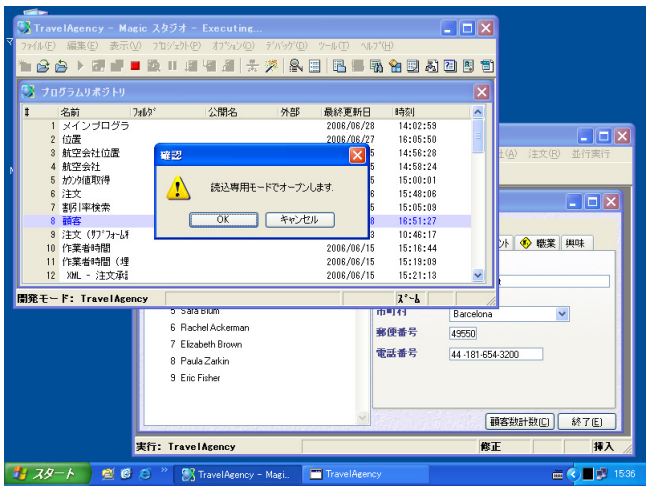
このとき、開発エンジンは次のように表示されます。

- タイトルバーには「**プロジェクト名 – Magic スタジオ – Executing...**」と表示されます。
- 停止ボタンが四角い赤色 で表示されます。



このときには、実行エンジンと開発エンジンとは独立したプロセスとして動作しています。このため、実行エンジンでプログラムを実行しながら、開発エンジンでプログラムを開くことが可能です。ただし、この場合にはプログラムを修正することはできず、読込専用モードで開くことになります。

下図は、**顧客** プログラムを実行している状態で、開発エンジンでプログラムを開き、**顧客**プログラムを開こうとしているところです。



13.1.2. RunMode 関数

V9Plus でも RunMode 関数がありましたが、V10 では、開発エンジンと実行エンジンが分離されたことにより、RunMode 関数の戻り値の意味が次のように変更されました。

戻り値	意味
-1	以下のいずれかの場合です。 ● 最初にメインプログラムが実行された場合 ● プロジェクトがクローズされ、他のプロジェクトが最初のコンテキストでオープンされた場合 ● プロジェクトがマルチスレッドで最初にオープンされた場合 サブタスクや、サブプログラムで実行された場合は、0 になります。
0	プログラムが実行版で動作している。フォアグラウンドまたは、バックグラウンドで動作している場合 開発エンジンがバックグラウンドで動作している場合
1	(V10 では無効です)
2	プログラムが開発版で動作しており、プロジェクトが実行モードに切り替わった場合
3	プログラムが開発版で動作しており、プログラムリポジトリ上から実行されている場合

13.1.3. 実行中のアプリケーションを閉じる

実行エンジンは、次のような事象によりアプリケーションの実行が終了した場合に閉じられます。

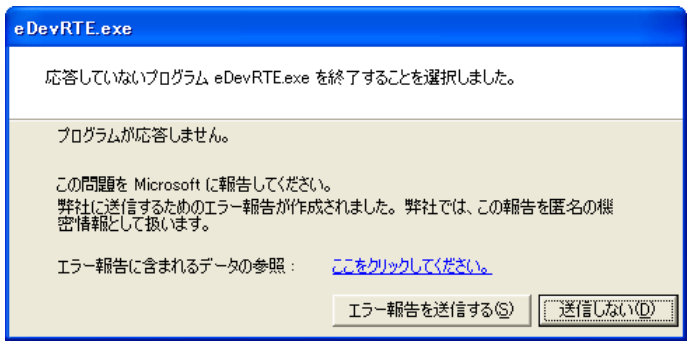
- **アプリケーションを閉じる** イベントが発行された。
- ウィンドウタイトルバーの ボタンがクリックされた。
- メインプログラムのタスク終了条件が真になった。

開発エンジンからアプリケーションが開始された場合、アプリケーションの終了と共にアプリケーションは閉じられ、実行エンジンのウィンドウは再びデスクトップから隠れます。しかし、実行エンジンは背後で動作したままで、後に再度実行モードになったときに現れます。

13.1.4. 実行エンジンの異常終了時の動作

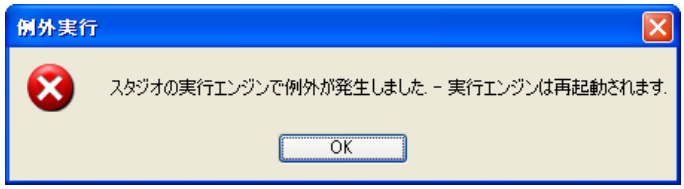
Magic プログラムのエラーなどにより、実行時に不正な動作を行った場合、実行エンジンは異常終了することがあります。また、タスクマネージャなどから、実行エンジン eDevRTE.exe を強制的に終了させることもできます。

このような場合には、オペレーティングシステムは次のようなダイアログを表示します（この表示は、オペレーティングシステムの設定によって変わります）。



また、開発エンジンは、実行エンジンが異常終了したことを検知し、次のようなエラーメッセージを出します。

その後、開発エンジンは実行エンジンを再起動します。

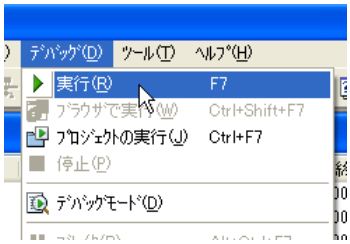


この場合にも、実行エンジンは再起動されますが、ウィンドウは隠れています。次に実行モードになったときにウィンドウが表示されます。

13.1.5. 開発エンジンからプログラムを実行する

開発エンジンから、特定のプログラムのみを実行することができます。

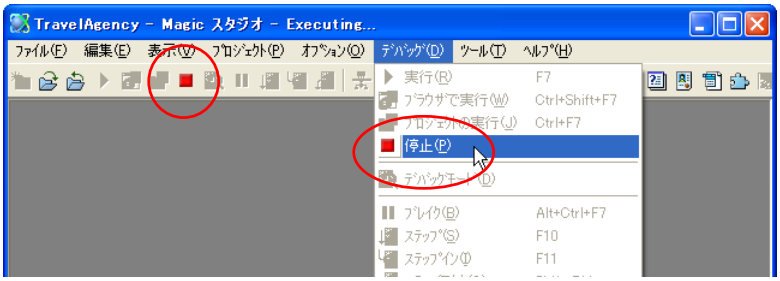
特定のプログラムを実行するには、プログラムリポジトリを開き、実行したいプログラムにカーソルを置いてから、**デバッグ** メニュー → **実行**（あるいは **F7** キー）を選択します。



このときには、実行エンジンは次のような動作を行います。

1. 実行エンジンのウィンドウが前面に表示されます。
2. プロジェクトが開かれます。
3. メインプログラムのタスク前処理が実行されます。
4. 指定されたプログラムが起動されます。
5. プログラムが終了したら、メインプログラムのタスク後処理が実行されます。
6. プロジェクトがクローズされ、実行エンジンのウィンドウが隠されます。

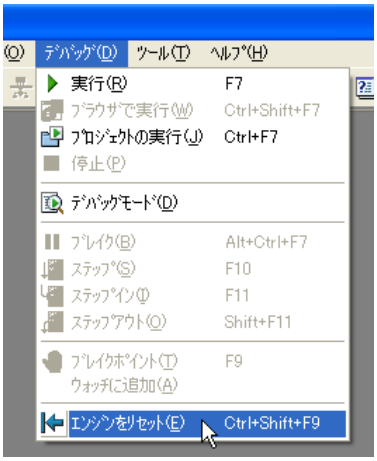
13.1.6. 実行エンジンの強制停止



プログラムのエラーなどにより、実行エンジンが終了しなくなってしまったり、あるいは非常に時間のかかる実行を強制的に中断させたい場合があります。
このときには、**デバッグ** メニューから **停止** を選びます。ツールバーの停止ボタン から停止させることもできます。

これにより、現在実行中の実行エンジンのプロセスが強制的に終了させられ、実行エンジンが再起動させられます。実行エンジンは再起動しますが、ウィンドウは隠されたままです。また、モードは開発モードに戻ります。

13.1.7. 実行エンジンをリセットする



通常、開発エンジンの背後で動いている実行エンジンは、実行モードが終わって開発モードに戻ったときにもそのまま実行を続けています。
しかし、プログラムのエラーなどで、実行エンジンの状態がおかしくなっていることが疑われる場合には、実行エンジンを再起動してリセットしたい場合があります。
このような場合には、**デバッグ** メニューから **エンジンをリセット** を選択します。
これにより、コンテキスト情報等が初期化されます。

13.2. アプリケーションの配備

本書の今までの説明では、プログラムを実行するのに、開発エンジンから起動するようにしていました。
本節では、開発エンジンを使わず、実行エンジンで直接アプリケーションを実行する場合の説明をします。

開発エンジンからプロジェクトを実行する場合とは異なり、実行エンジン eDevRTE.exe で直接アプリケーションを実行するには、**キャビネットファイル** を作成しなければなりません。キャビネットファイルというのは、プロジェクトの全リポジトリの内容を、Magic の内部形式に変換して、コンパクトかつ高速にアクセスできるようなバイナリ形式にしたものです。V9Plus でのフラットファイル(MFF)と似ています。

V9Plus では、実行エンジンはコントロールファイル MCF もフラットファイル MFF も実行することができましたが、V10 では、実行エンジンはキャビネットファイルのみを実行することができます。プロジェクトファイルをそのまま実行することはできません。

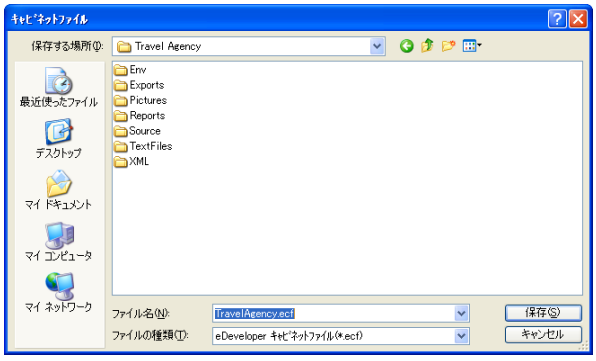
13.2.1. キャビネットファイルを作成する

キャビネットファイルは、開発エンジンで作成します。キャビネットファイルを作成するのは、V9Plus でフラットファイルを作成したのとほぼ同様の手順で行います。

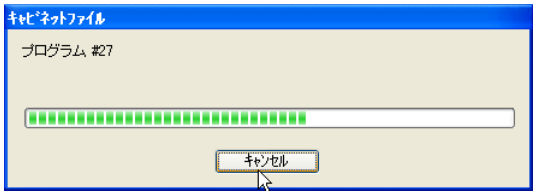
キャビネットファイルの作成方法

1. 開発エンジンを起動します。
2. プロジェクトを開きます。
3. ファイル メニューから、**キャビネット作成** を選びます。

4. キャビネットファイルの位置を指定します。



5. **保存** ボタンを押すと、キャビネットファイルの作成を開始します。



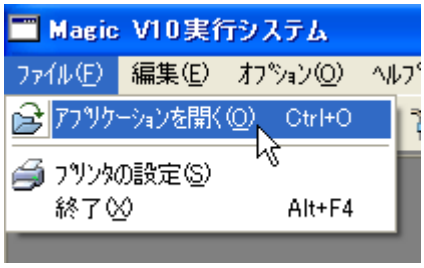
キャビネットファイルの拡張子は、.ecf (eDeveloper Cabinet File の略)です。

13.2.2. キャビネットファイルをオープンする

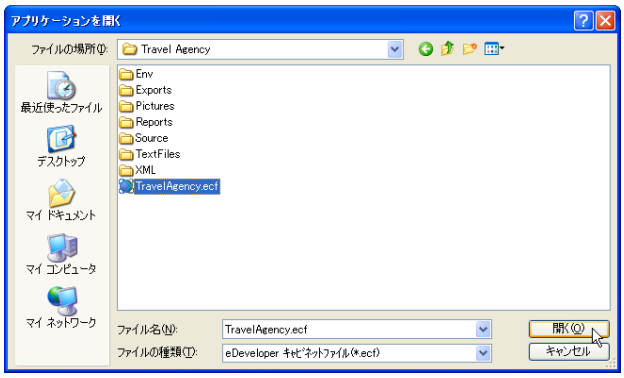
キャビネットファイルを作成したら、実行エンジンから開いて実行させることができます。

実行エンジンからキャビネットファイルを実行させるには・・・

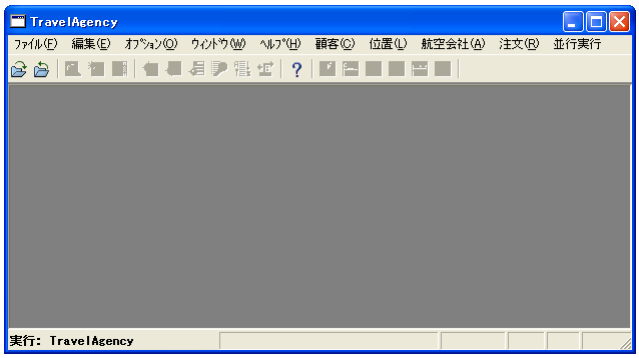
1. 実行エンジンを起動します。
2. **ファイル** メニュー から **アプリケーションを開く** を選択します。



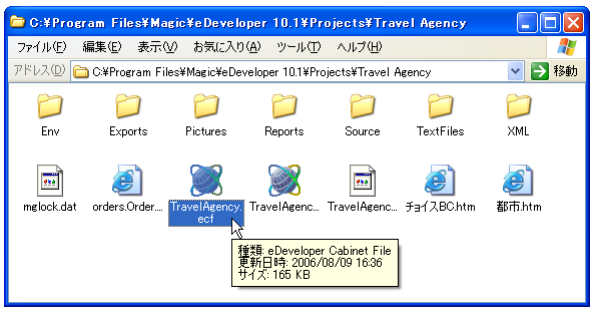
3. 実行するキャビネットファイルを選択します。



4. **開く** ボタンを押すと、アプリケーション実行が開始されます。



- 実行エンジンのデフォルトの作業ディレクトリは、キャビネットファイルの存在するディレクトリになります。
- エクスプローラなどから、キャビネットファイル をダブルクリックしても、実行エンジンが起動されます。



13.2.3. アプリケーションを閉じる

アプリケーションを終了したい場合には、アプリケーションを閉じます。

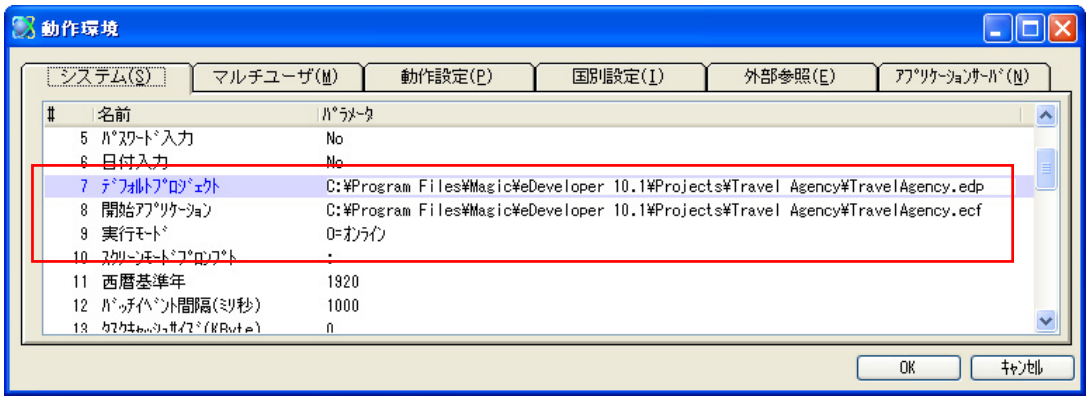
実行エンジンで実行しているアプリケーションを閉じるには、次のいずれかを行います。

- 全てのプログラムが閉じている状態で、**ESC** キーを押す、あるいはウィンドウツールバーの ボタンを押して、実行エンジンを閉じる。
- 全てのプログラムが閉じている状態で、**閉じる** イベントを実行する。
- **アプリケーションを閉じる** イベントを実行する。
- **ファイル** メニューから **アプリケーションを閉じる** を選ぶ。

13.3. 動作環境の設定

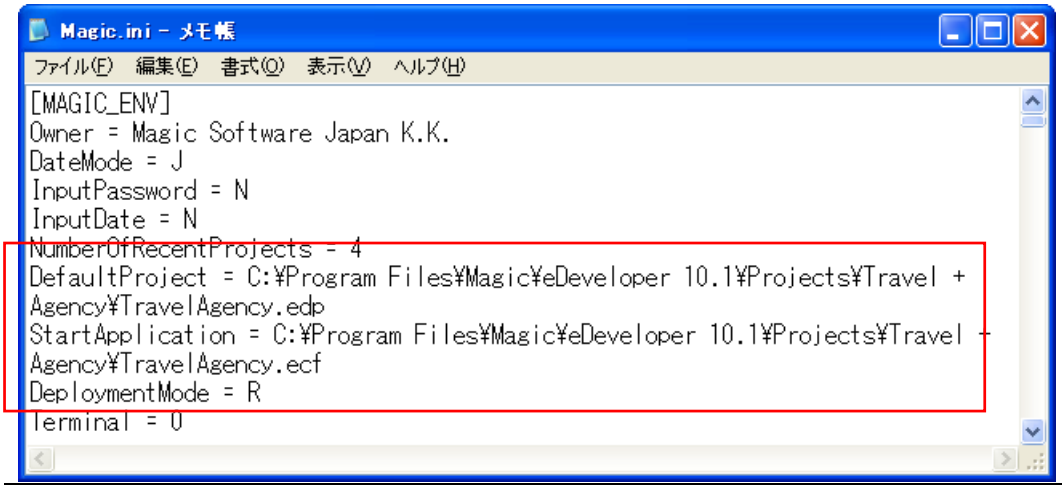
動作環境 ダイアログにおいて、アプリケーション実行に関し、次のようなパラメータがあります。

- デフォルトプロジェクト
- 開始アプリケーション
- 実行モード



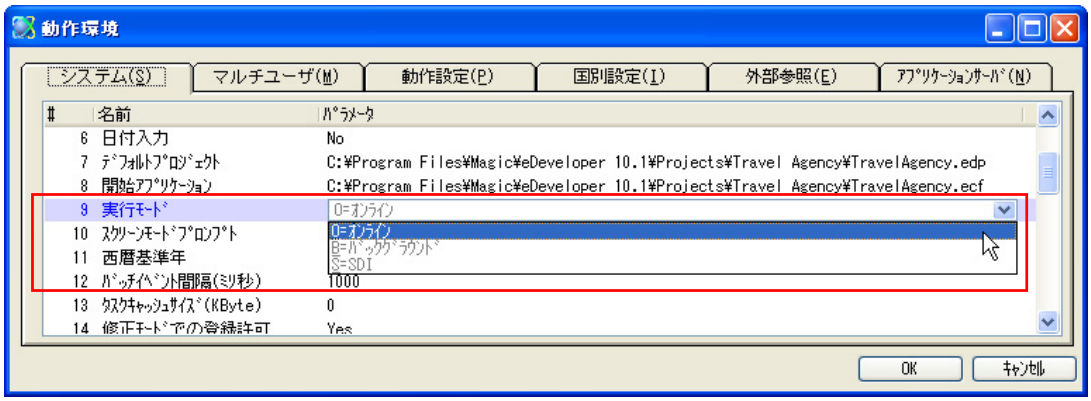
これらのパラメータの意味は次の通りです。

パラメータ	意味	MAGIC.INI のキーワード
デフォルトプロジェクト	開発エンジンが起動したときに、自動的に開かれるプロジェクトファイル名を指定します。	DefaultProject
開始アプリケーション	実行エンジンが起動したときに、自動的に開いて実行されるキャビネットファイル名を指定します。	StartApplication
実行モード	開始アプリケーションが指定されたとき、どのような形で実行がされるのかのモードを指定します。	DeploymentMode



実行モード パラメータについては、次のような値が指定できます。

実行モードパラメータ値	意味	MAGIC.INI での設定値
O=オンライン	通常の実行モード（実行エンジンのウィンドウがディスプレイ上に表示される）	O
B=バックグラウンド	実行エンジンのウィンドウは表示されず、バックグラウンドのマルチスレッド アプリケーションサーバとして動作する。	B
S=SDI	実行エンジンのウィンドウは表示されず、その代わりに、SDI スタイルのウィンドウが初期画面として表示される。 参考: SDI スタイルのウィンドウについては、「14.8 SDI」を参照。	S



14. 複数のタスクを同時に実行する

V9Plus では、Magic アプリケーションサーバはマルチスレッドで動作するものであり、一つのインスタンス(プロセス)の中で、同時に複数のバッチタスク・ブラウザタスクが並行して実行することができました。しかし、V9Plus では、フォアグラウンドのクライアントで、複数のオンラインタスクを並行して同時に実行することはできませんでした。
Magic V10 では、フォアグラウンドのクライアントで、複数のオンライン プログラムを並行して同時に実行することができるようになりました。
プログラムの並行実行は、コンテキスト を使って実装されています。あるプログラムが並行実行を開始すると、Magic 内部では新しいコンテキストが作成されます。コンテキストには、メインプログラム、グローバル変数、メモリテーブル、実行スタック、その他のプログラム実行状態に関する情報がすべて格納されています。コンテキストはお互いに独立して存在するもので、普通はお互いが干渉しあうことはありませんが、プログラムで明示的にコンテキスト名を指定してイベントなどを使ってお互いに通信することができます。

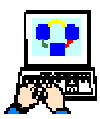
本章では、Magic V10 の並行実行プログラムについて解説します。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

並列実行の設定	**プログラム → タスク特性 → [拡張]タブ**
コールコマンド	**ロジックエディタ → 処理コマンド → コールプログラム**
プログラム再呼出 イベント	**Magic エンジン → エンジン実行レベル → イベントレベル → 内部イベント一覧 → プログラム再呼出**
MDI と SDI 一般	**表示フォーム → MDI と SDI**
MDI と SDI の実行時動作	**表示フォーム → MDI と SDI → 実行時の動作**
コンテキスト間のイベント	**表示フォーム → MDI と SDI → コンテキスト間のイベント**
コンテキスト関数	**式エディタ → 関数ディレクトリ** 以下の ctx 関連関数
ウィンドウメニュー	**表示フォーム → MDI と SDI → MDI サポート → ウィンドウメニュー**
コンテキスト関係イベント	**Magic エンジン → エンジン実行レベル → イベントレベル → 内部イベント一覧**
イベントメニューの出力コンテキスト名	**メニューとヘルプ → メニュー → メニュー特性 → 特性タブ** **ロジックエディタ → 処理コマンド → イベント実行 → イベント実行特性**

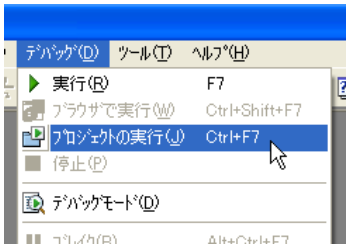
14.1. 並行実行プログラムのいろいろ

まずは、サンプルを使って、並行実行プログラムの基本的な動作を見ていきます。

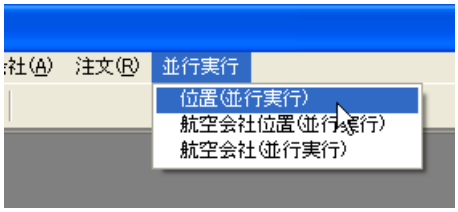


TravelAgency にある並行実行プログラムの実行例

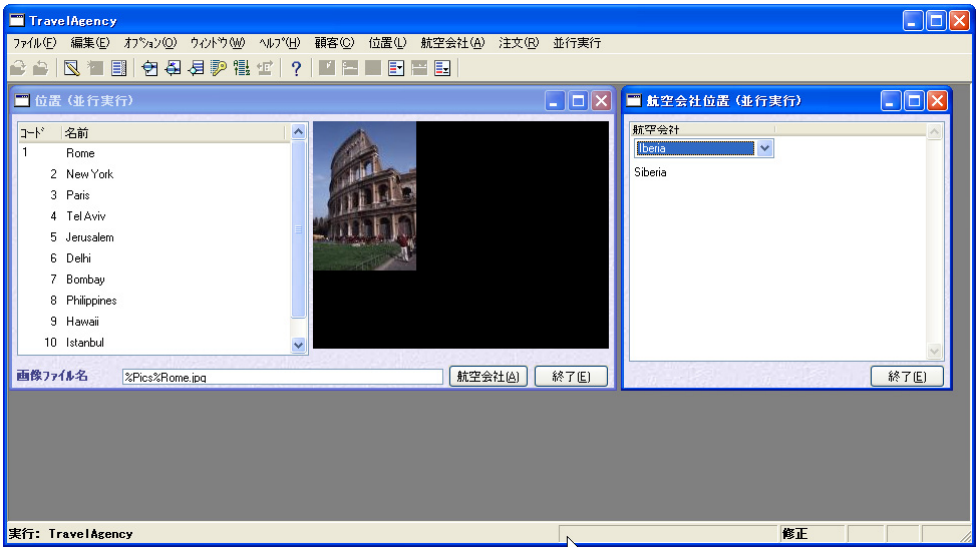
1. 最初に、プロジェクトを実行モードにします。これは、**デバッグ** メニューから **プロジェクトの実行** を選びます。



2. 実行エンジンが現れるので、 **並行実行** メニューから **位置(並行実行)** を選択します。



位置（並行実行）というタスクと、**航空会社位置(並行実行)**というタスクとが表示されます。



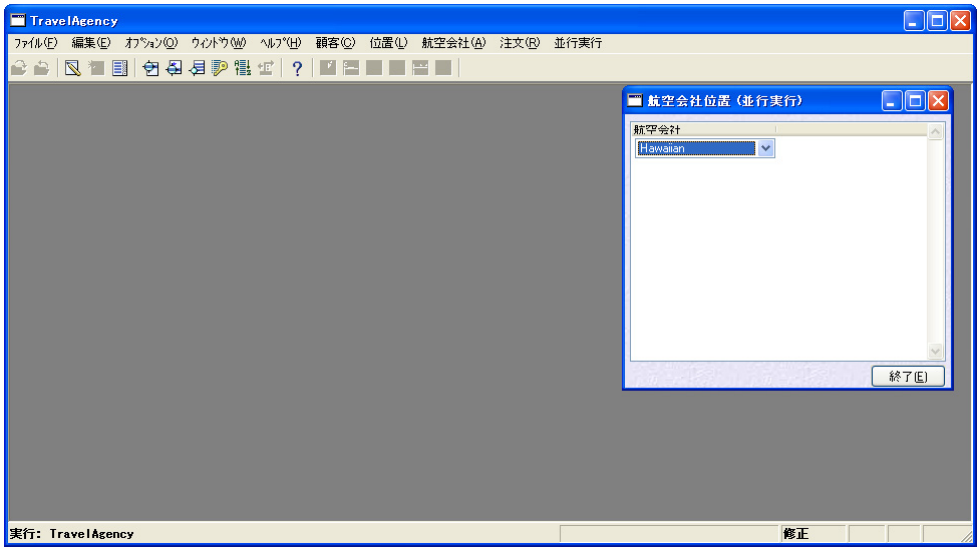
左側の **位置(並行実行)** のテーブルには都市の名前が表示され、右側の **航空会社位置(並行実行)** には、その都市に乗り入れている航空会社名が表示されます。

3. 都市のカーソルを移動すると、それに合わせて右側の 航空会社名の表示も変わります。

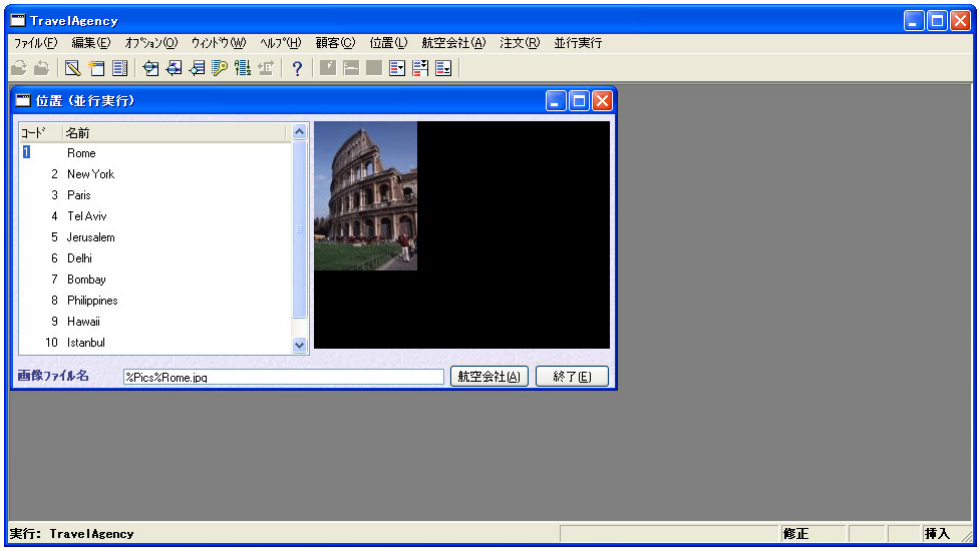
14.1.1. 並行実行の確認

これだけでは、通常の親子関係を持ったファントムタスクと似ています。この二つのタスクが並行に実行していることを確かめるには次のことを確認してください。

- **位置(並行実行)** の ボタンを押して、タスクを閉じてください。**位置(並行実行)** のウィンドウが閉じますが、**航空会社位置(並行実行)** のウィンドウはそのままです。



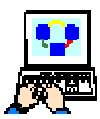
- **航空会社位置(並行実行)** のウィンドウも閉じ、再度 **並行実行** メニューから **位置(並行実行)** を選択してください。そして右側の **航空会社位置(並行実行)** のウィンドウを先に閉じてください。今度は、左側の **位置(並行実行)** ウィンドウが残ります。



このことから、この二つのタスクは親子関係をもっているのではなく、並行に平等に存在していることがわかります。

14.2. 並行実行の設定

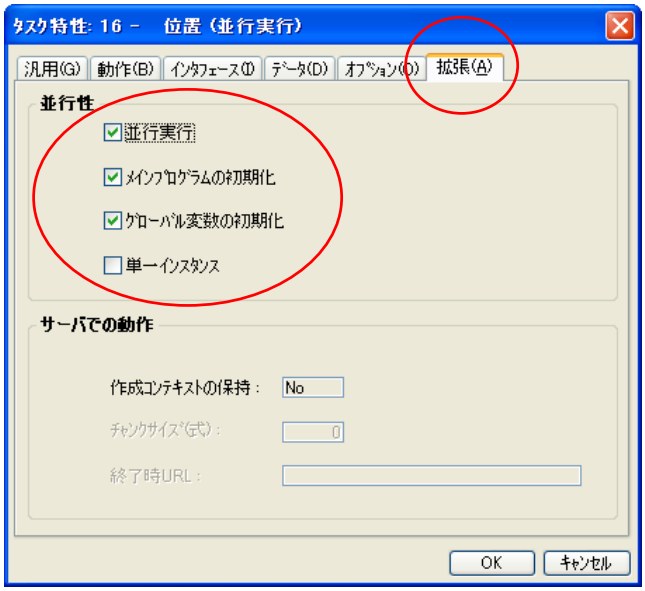
タスクは、デフォルトでは並行実行を行いません。並行実行をさせるには、タスク特性で指定します。



並行実行を行うように設定するには・・・

並行実行をさせたいプログラムを開き、**タスク特性** の **拡張(A)** タブを開きます。ここで、 **並行実行** というパラメータがありますので、これにチェックを入れます。

下図は、プログラム 16 番 **位置(並行実行)** のタスク特性です。



並行実行 にチェックが入っているプログラムが呼び出された場合には、Magic 実行エンジンは次のことを行います。

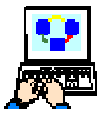
1. 新たにコンテキストを作成します。このプログラム、およびこのプログラムから呼び出されるタスクやプログラム(**並行実行** パラメータにチェックが入っていないもの)は、すべてこのコンテキストの中で実行されます。
2. **グローバルパラメータを複写** にチェックが入っていた場合には、呼び出し元のコンテキストから、グローバルパラメータ(SetParam で設定される値)をこの新しいコンテキストにコピーします。
3. **メインプログラムを初期化** にチェックが入っている場合には、新しいコンテキストのメインプログラムのタスク前処理が実行されます。チェックが入っていない場合には、メインプログラムのタスク前処理は実行されず、メインプログラムのグローバル変数の値は、呼び出し元のコンテキストのグローバル変数からコピーされます。
   なお、いずれの場合にも、コンテキストが終了する場合には、メインプログラムのタスク後処理は実行されます。

ここには **単一インスタンス** というパラメータもありますが、これについては次節で説明します。

先に見たサンプルでは、**位置(並行実行)** プログラム(プログラム 16 番) に、**並行実行** パラメータがセットされています。また、このプログラムのレコード前処理で、**航空会社位置(並行実行)** プログラム(プログラム 17 番)を呼び出していますが、このプログラムにも **並行実行** パラメータがセットされています。従って、後者は前者から呼び出されるようにはなっていますが、親子関係ではなく、平等に並行に実行されるようになります。

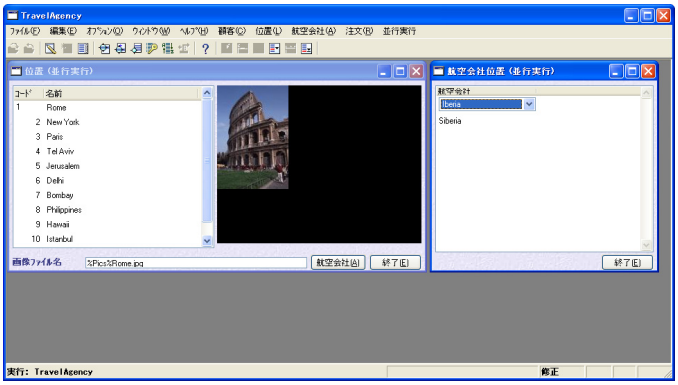
14.3. 単一インスタンス

単一インスタンス パラメータは、このプログラムが複数回呼ばれた場合、それに応じて複数のコンテキストを作成するか、あるいは単一のコンテキストのみが許可されるかを指定します。**単一インスタンス** パラメータにチェックが入っている場合には、単一のコンテキストのみが許可されます。

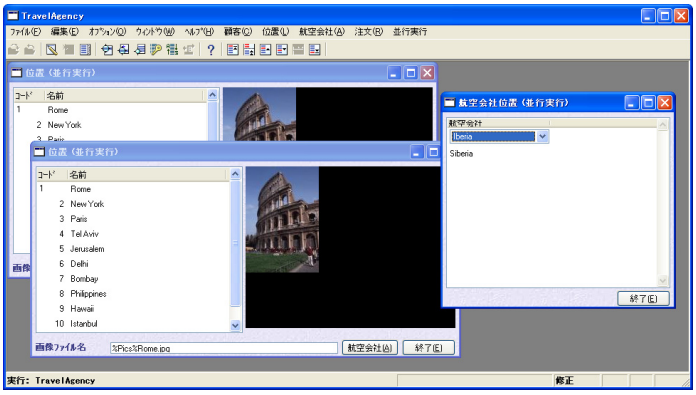


単一インスタンス パラメータの効果を、サンプルで見てみるには・・・

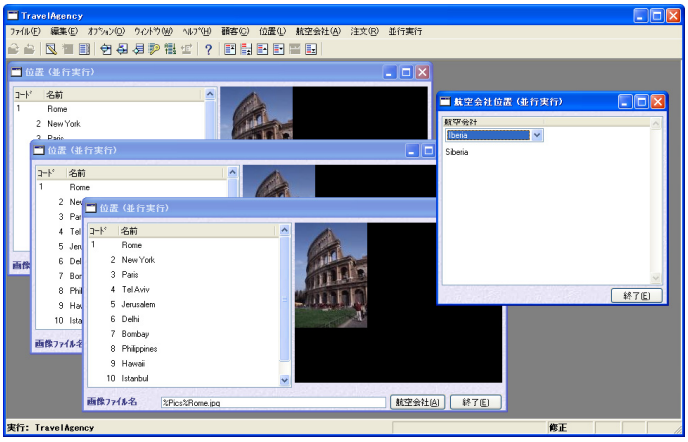
1. **デバッグ メニュー** から **プロジェクトの実行**（あるいは Ctrl+F7）を選んで、実行モードにしてください。
2. **並行実行** メニューから、**位置(並行実行)** を選びます。**位置(並行実行)** と **航空会社位置(並行実行)** プログラムが表示されます。



3. もう一度、**並行実行** メニューから、**位置(並行実行)** を選びます。すると、**位置(並行実行)** プログラムは二つ表示されますが、**航空会社位置(並行実行)** プログラムはひとつだけしか表示されません。(ウィンドウが重なっているので、ウィンドウの位置をずらして、確認してください)

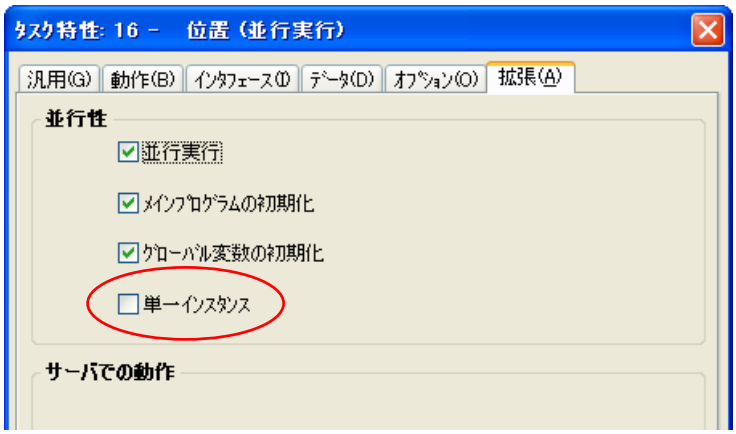


4. 念のため更にもう一度、**並行実行** メニューから、**位置(並行実行)** を選びます。同様に、**位置(並行実行)** プログラムは三つ表示されますが、**航空会社位置(並行実行)** プログラムはやはりひとつだけしか表示されません。

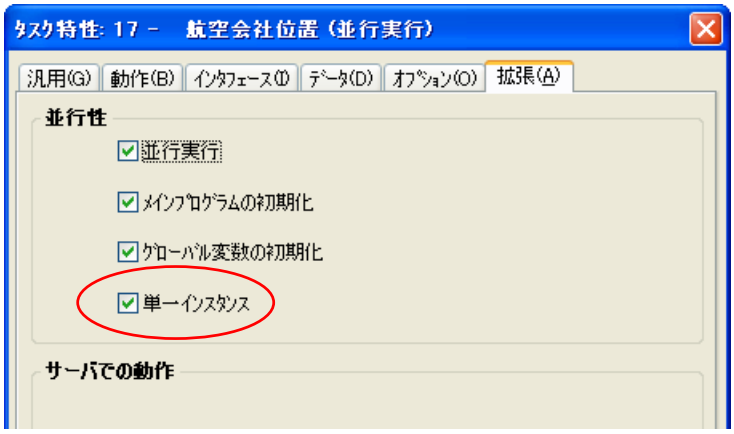


この違いは、この二つのプログラムの **単一インスタンス** パラメータの設定の違いによるものです。

位置(並行実行)プログラム（プログラム 16 番)のほうは、**単一インスタンス** にチェックが入っていません。従って、呼び出された回数分だけのコンテキストが作成されます。



一方、**航空会社位置(並行実行)**（プログラム 17 番)のほうには、**単一インスタンス** にチェックが入っています。このため、何回呼び出されても、コンテキストはただ一つだけしか作成されません。



14.4. プログラム再呼出 イベント

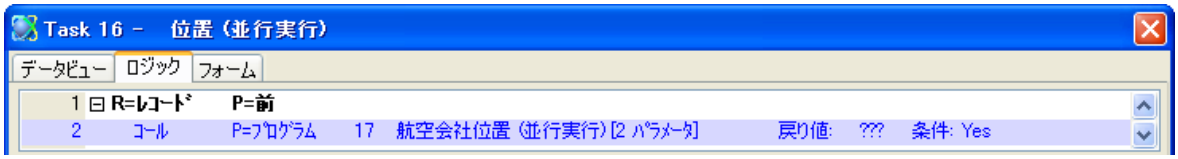
単一インスタンス にチェックが入っているプログラムでは、何回呼び出しがあってもコンテキストは一つしか作成されません。
しかし、2 回目以降の呼び出し時には、コールコマンドで与えられたパラメータや最新のデータベースレコード値などをもとに、最新の表示に更新したいことがあります。

例えば、**航空会社位置(並行実行)** プログラムでは、**位置(並行実行)** プログラムでカーソルのある都市に合わせて、その都市に乗り入れている航空会社の一覧を表示します。カーソルが動いたら、それに応じて、航空会社も再表示しなければなりません。

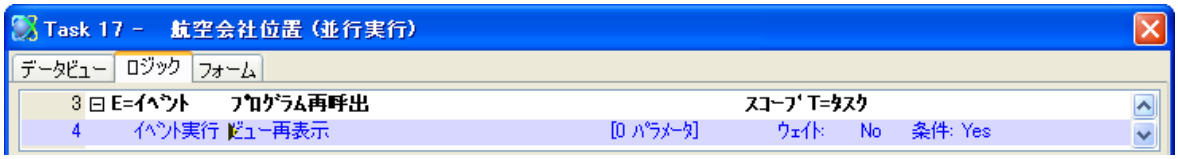
これを実現するために、単一インスタンスの並行実行プログラムが再度呼び出された場合に、Magic 実行エンジンは、そのプログラムに対して **プログラム再呼び出し** イベントを発行します。プログラムはそのイベントに対するハンドラを定義して、表示を最新にするなどの適当な処理を行います。

このイベントを使って、次のように実現することができます。

- **位置(並行実行)**プログラムでは、レコード前処理で**航空会社位置(並行実行)**プログラムを呼び出します。このときに、パラメータとして、現在の都市のコードを渡します。



- **航空会社位置(並行実行)**プログラムでは、**プログラムの再呼び出し** イベントのハンドラを定義し、そこで **ビュー再表示** イベントを発行します。



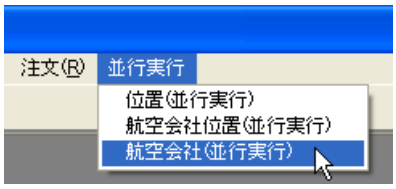
14.5. 並行実行プログラム間の相互作用

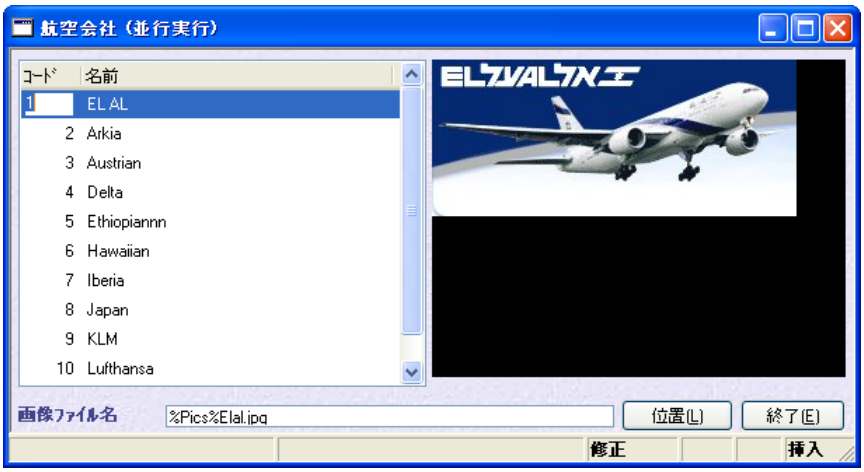
並行に実行しているプログラムの間で相互作用を行いたい場合(表示の同期をとりたい場合など)は、前節で説明したように、コールコマンドと プログラム再呼出 イベントとをもちいて実現することもできますが、別の方法として、ここで示すように、ユーザ定義イベントを使うこともできます。

14.5.1. 航空会社プログラムの例

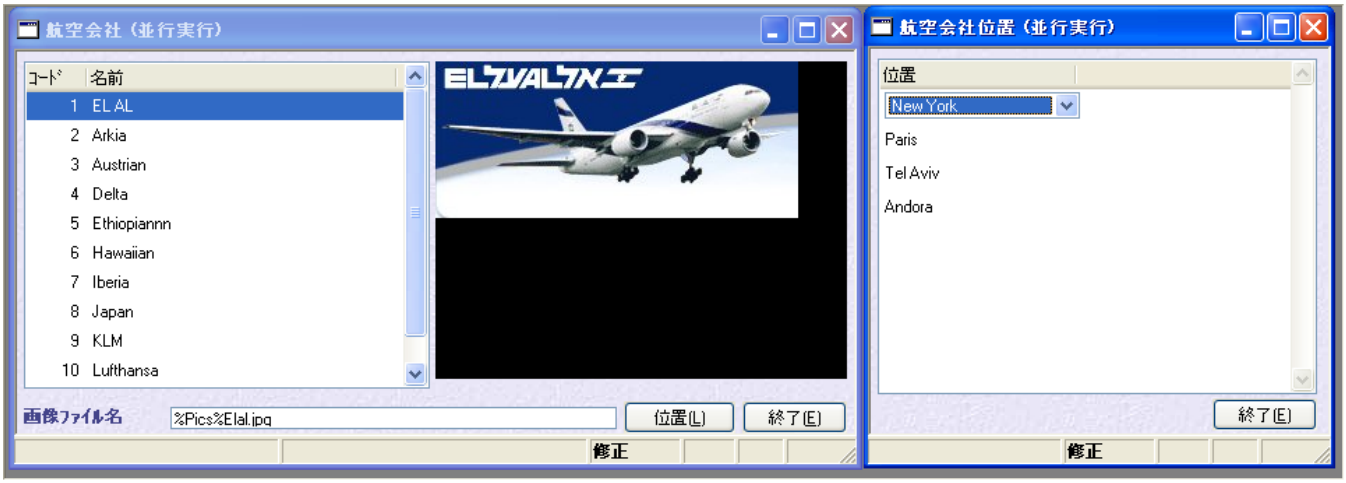
ここでは、サンプルの **航空会社(並行実行)** プログラム (プログラム 18 番)と、**航空会社位置(並行実行)** (プログラム 17 番)との間の、イベントのやりとりを例にとって説明します。

1. このプログラムは、実行モードで **並行実行** メニュー から **航空会社(並行実行)** を選択して起動します。
   航空会社の一覧と、カーソルのある航空会社のロゴマークが表示されます。





2. **位置(L)** ボタンを押します。現在カーソルのある航空会社が運行する都市の一覧が右側の **航空会社位置(並行実行)** ウィンドウに表示されます。左側のウィンドウの航空会社名を移動すると、それに応じて都市の一覧が更新されることを確認してください。



この二つのプログラムは、大略、次のような構成で実現されています。

1. それぞれのプログラムは、並行実行しています。(**並行実行** パラメータにチェックが入っています)。このため、二つのコンテキストが並行して動作していることににになります。
2. V10 では、**CtxSetName** 関数を使って、コンテキストに名前を付けることができます。今の場合、右側の **航空会社位置(並行実行)** のプログラムが動作しているコンテキストに、AirlineLocations という名前が付けられています。
3. また、V10 では、**イベント実行** コマンドで、イベントを発行する先のコンテキスト名を指定することができます。今の例では、**航空会社(並行実行)** プログラム上でカーソルが動いたとき、**コンテキストのリフレッシュ** という名前のユーザ定義イベントを、AirlineLocations という名前のコンテキストに向けて発行します。
4. **航空会社位置(並行実行)** プログラムでは、**コンテキストのリフレッシュ** イベントに対するイベントハンドラを定義しておいて、データの再表示をさせるようにします。
5. V10 では、イベントにパラメータを持たせることができるようになりました。データ再表示に際しては、現在選択されている都市名を知ることが必要ですが、これは **コンテキストのリフレッシュ** イベントのパラメータとして与えるようにします。

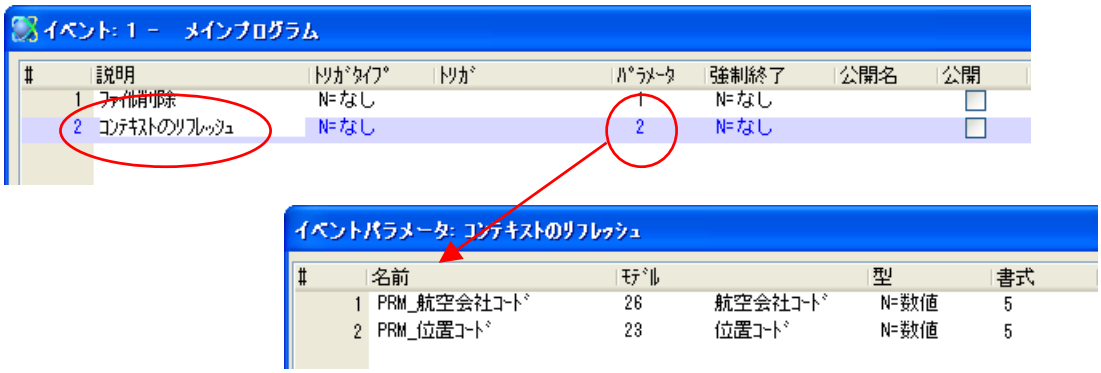
以下に、それぞれについて詳しく説明していきます。

14.5.2. コンテキストのリフレッシュ イベントの作成

最初に、ユーザ定義イベントとして **コンテキストのリフレッシュ** イベントを定義します。このイベントはプログラム 17 番と 18 番とで参照されるので、グローバルイベントとして、メインプログラムで定義されている必要があります。
メインプログラムを開いて、**タスク環境(K)** メニューから **ユーザイベント(U)** (あるいは Ctrl+U) を選択してください。ここに **コンテキストのリフレッシュ** というユーザ定義イベントが以下のように定義されています。

説明: コンテキストのリフレッシュ
トリガタイプ: N=なし
パラメータ: 2 (ズームしてパラメータテーブルを表示)
強制終了: N=なし

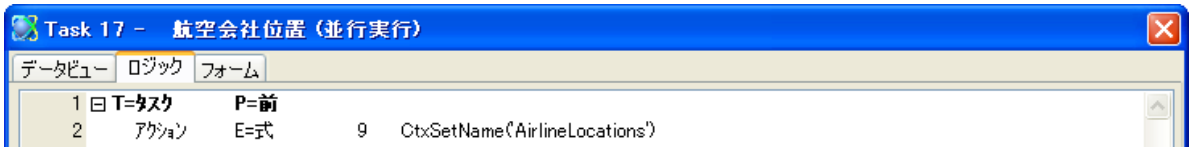
このイベントには、パラメータが二つ定義されています。**パラメータ** 欄からズームすると、パラメータテーブルが表示されます。



14.5.3. コンテキスト名の設定

右側の **航空会社位置(並行実行)** のプログラム(プログラム 17 番)が動作しているコンテキストには、AirlineLocations という名前を付けます。

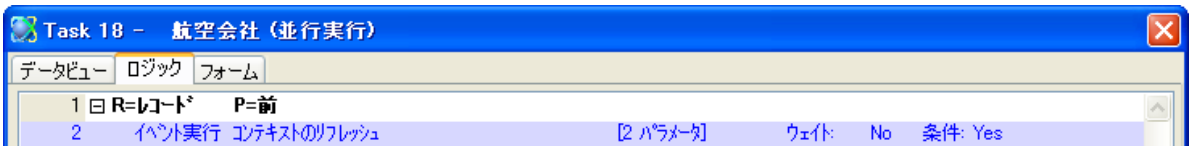
このために、プログラムのタスク前処理で `CtxSetName('AirlineLocation')` を実行します。

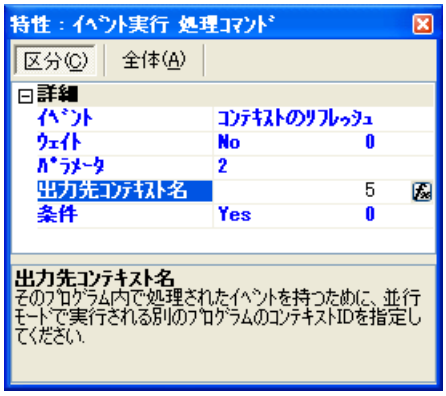


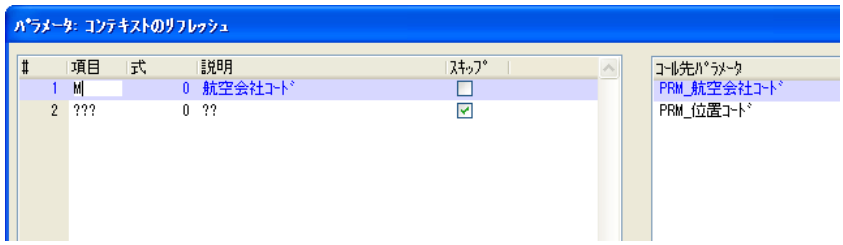
14.5.4. コンテキストのリフレッシュ イベントの実行

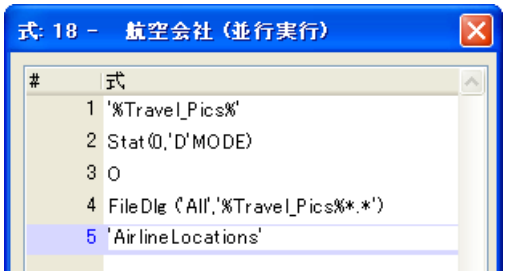
航空会社(並行実行) プログラム(プログラム 18 番)上でカーソルが動いたとき、ユーザ定義イベント **コンテキストのリフレッシュ**を、AirlineLocations という名前のコンテキストに向けて発行します。

- プログラムのレコード前処理で、**イベント実行** コマンドを実行します。。
- **イベント実行** コマンドの特性シート上で、**出力先コンテキスト名** を式で`'AirlineLocation'`と設定します。
- パラメータは、**パラメータ** 欄からズームして設定します。パラメータとしては、**PRM_航空会社コード** と **PRM_位置コード** の二つがありますが、今は **PRM_航空会社コード** の方だけを指定します。







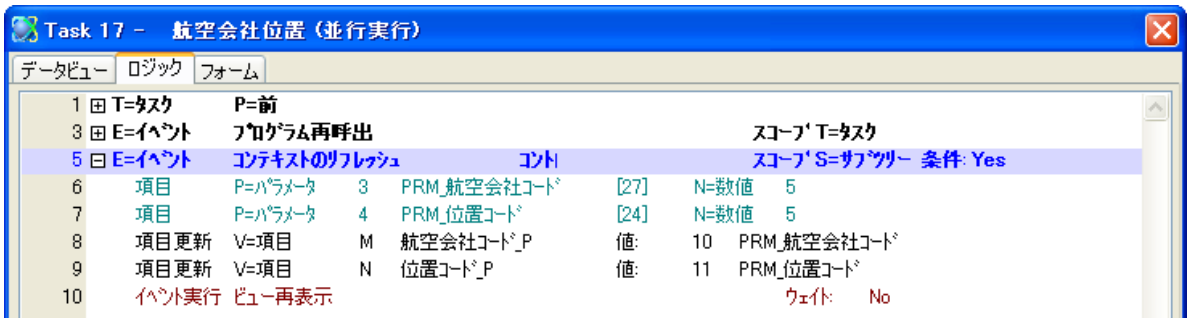


14.5.5. コンテキストのリフレッシュ イベントの処理

このイベントは、**航空会社位置(並行実行)** プログラム(プログラム 17番)でキャッチして処理する必要がありますので、イベントハンドラを定義します。

ここでは、次のような処理を行います。

1. **項目 P=パラメータ** 行により、イベントのパラメータ **PRM_航空会社コード** および **PRM_位置コード** を受け取ります。
2. イベントのパラメータの値を、タスクのパラメータ **航空会社コード_P** および **位置コード_P** に代入します。これらのパラメータは、このタスクのビューの範囲指定に用いられています。
3. **ビュー再表示** イベントを実行し、新しい範囲指定に従って、レコードの再読込を行います。



14.6. コンテキスト ID

14.6.1. コンテキスト ID の生成、設定と取得

コンテキスト ID は、コンテキスト作成時に Magic 実行エンジンが自動的に作成します。システム生成のコンテキスト ID は、十数桁の数値を文字列としたものです。
コンテキスト ID は、**CtxSetName** 関数によりプログラムから明示的に設定することもできます。(14.5.3 コンテキスト名の設定 を参照)。
また、コンテキスト ID は、**CtxGetName** 関数により取得することができます。

14.6.2. メインコンテキスト

アプリケーションをフォアグラウンドの実行エンジンで実行する場合、最低一つのコンテキストが存在します。アプリケーションを開いた直後に実行されるメインプログラム、およびそこから呼び出されるプログラム(**並行実行** がセットされていないもの)は、すべてこのコンテキスト内で実行されます。このコンテキストを **メインコンテキスト** と呼びます。
メインコンテキストのコンテキスト ID は、**Main** です。

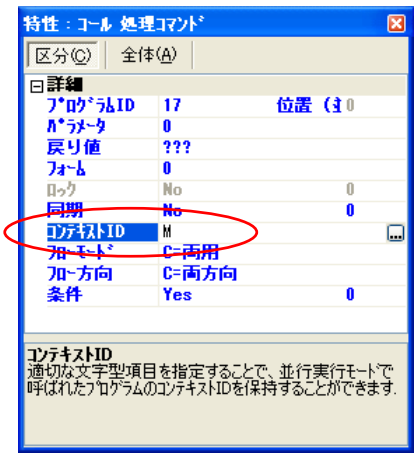
14.6.3. 新規作成したコンテキスト ID の取得

イベント実行 コマンドなどで、特定のコンテキストを指定した操作を行いたい場合、コンテキスト ID を指定しますので、コンテキスト ID を知っておく必要があります。
コンテキスト ID は、比較的単純なアプリケーションでは予め固定的に決めておくこともできます。例えば、前述の **航空会社位置(並行実行)** プログラムの例では、イベント発行先のコンテキスト名は予め AirlineLocations という名前に決められていました。しかし、より高度なプログラムでは、イベント発行先のコンテキスト名を開発時には決められないこともあります。

このような場合には、次に示すように、コンテキストを作成した（すなわち、並行実行がセットされたプログラムを呼び出した）コール コマンドあるいはメニュー項目に、コンテキスト ID を格納する変数を設定しておきます。

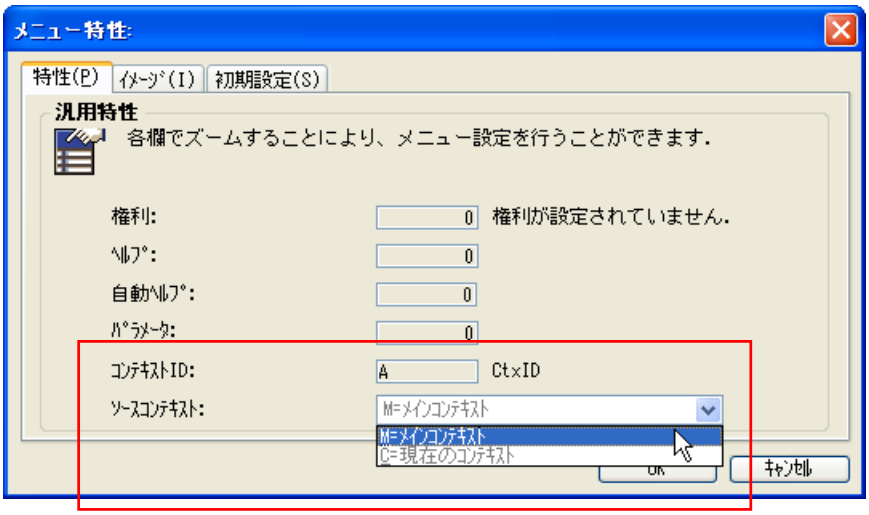
コールコマンドの コンテキスト ID 特性

並行実行 特性がセットされたプログラムを **コール** コマンドで呼び出すと、新しいコンテキストが作成されます。このコンテキスト ID を取得するには、コールコマンドの **コンテキストID** 特性に、文字型あるいは Unicode 型の変数を設定しておきます。こうすると、新規に作成されたコンテキストの ID が、この変数に設定されます。



メニュー項目の コンテキスト ID 特性

並行実行 特性がセットされたプログラムをメニューから呼び出すと、新しいコンテキストが作成されます。このコンテキスト ID を取得するには、メニュー特性の **コンテキストID** 特性に、文字型あるいは Unicode 型の変数を設定しておきます。こうすると、新規に作成されたコンテキストの ID が、この変数に設定されます。



メニューは、アプリケーションにグローバルに有効なものなので、ここにコンテキスト ID として設定する変数項目は、メインプログラムで定義されたグローバル変数でなければなりません。

メニュー特性には、この他に **ソースコンテキスト** という特性があり、次の値を設定することができます:

- M=メインコンテキスト
- C=現在のコンテキスト

これには、メニュー項目から呼び出されるプログラムの **並行実行** フラグがセットされているか否かで、意味が異なります。

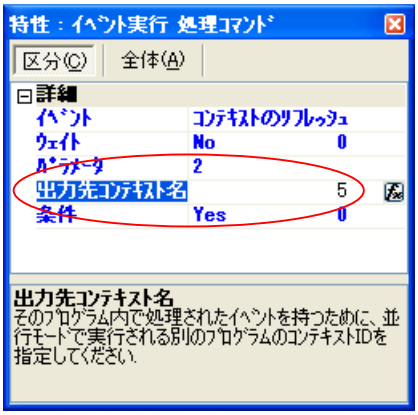
並行実行 がセットされているプログラムが呼び出される場合、新しいコンテキストが作成されますが、このときに、新しく作成されるコンテキストの内容(メインプログラムのグローバル変数や、グローバルパラメータの値)が、どこからコピーされるかが、このパラメータにより指定されます。**M=メインコンテキスト** の場合にはメインコンテキストからコピーされ、**C=現在のコンテキスト** の場合には、現在実行中でフォーカスのあるプログラムのコンテキストからコピーされます。
並行実行 がセットされていないプログラムが呼び出される場合には、新しいコンテキストは生成されません。

14.6.4. コンテキスト ID を指定してのイベントの発行

特定のコンテキストに対してイベントを発行したい場合には、イベント実行コマンド、あるいはイベントタイプのメニュー項目により行います。

イベント実行 コマンドの 出力先コンテキスト名 特性

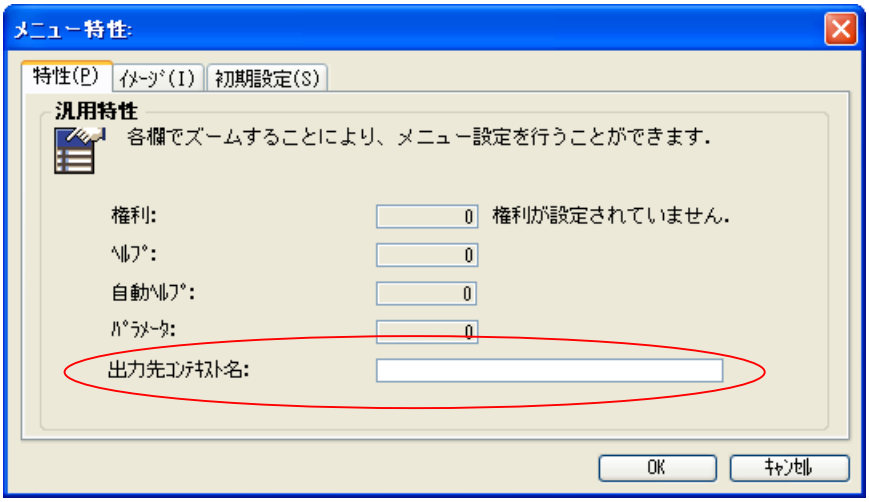
プログラムの中から、特定のコンテキストに対してイベントを発行するには、**イベント実行** コマンドの **出力先コンテキスト名** を指定します。



出力先コンテキスト名は、文字型あるいは Unicode 型の式で指定することができ、実行時に動的に評価されますので、柔軟なコンテキスト指定が可能になります。

イベントメニューの 出力先コンテキスト名 特性

タイプが **イベント** のメニュー項目により、特定のコンテキストに対してイベントを発行するには、メニュー特性の **出力先コンテキスト名** を指定します。



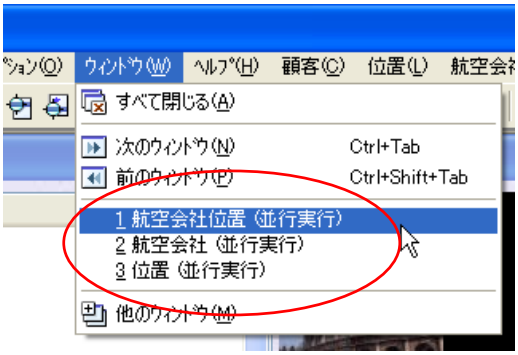
ここに指定できるのは固定された文字列で、実行中に動的に変更することはできません。

14.7. ウィンドウメニュー

14.7.1. メニューウィンドウの表示内容

実行エンジンには、**ウィンドウ** メニューがあります。プログラムが実行されていない状態では、ここには右図のような、ほぼ空のメニュー項目があります。

ここで、**並行実行** メニューから、**位置(並行実行)** および **航空会社(並行実行)** を選択した後、つまり 3 つのプログラムが並行実行している状態では、**ウィンドウ** メニューは以下のように、並行実行しているプログラムのウィンドウタイトルを表示するようになります。



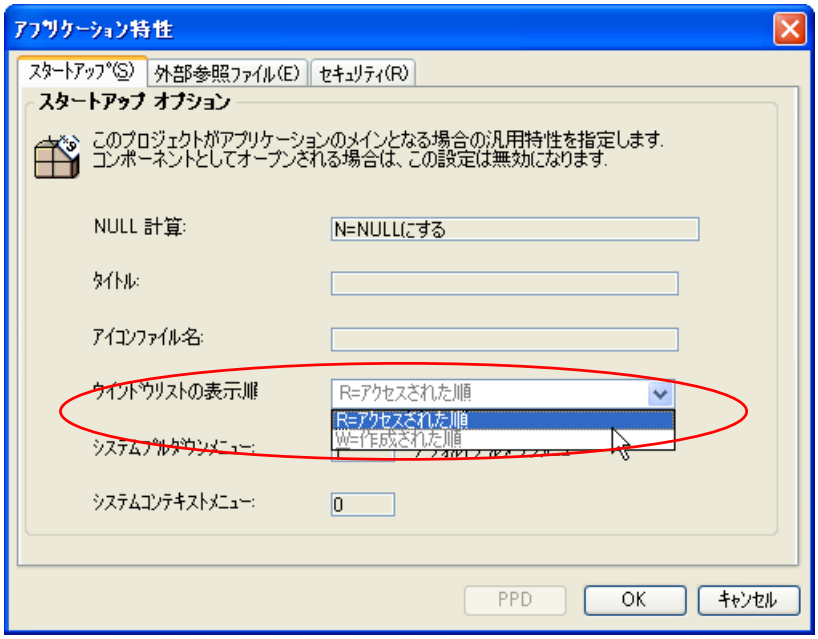
このウィンドウ一覧の中からウィンドウを選択することにより、ウィンドウにフォーカスを当てることができます。例えば、**位置(並行実行)** を選ぶと、**位置(並行実行)** プログラムのウィンドウにフォーカスが移ります。

ウィンドウ名の一覧の他に、**ウィンドウ** メニューには次のようなメニュー項目があります。

メニュー項目	意味
すべて閉じる(A)	現在開いているウィンドウを全て閉じます。プログラムは終了し、コンテキストは削除されます。
次のウィンドウ(N)	現在フォーカスのあるウィンドウから、一覧上で次のウィンドウにフォーカスを移します。
前のウィンドウ(P)	現在フォーカスのあるウィンドウから、一覧上で前のウィンドウにフォーカスを移します。
他のウィンドウ(M)	多くのウィンドウが開いていて、メニュー上に一覧を表示しきれない場合に、別途ウィンドウ一覧を表示する **その他のウィンドウ** ダイアログを表示して、その中から選択できるようにします。

14.7.2. ウィンドウの順序

ウィンドウ一覧に表示されるウィンドウの順序は、アプリケーション特性の **ウィンドウリストの表示順** パラメータにより決定されます。



ここには次のような選択肢があります。

ソートウィンドウ の値	意味
R=アクセスされた順	最後にフォーカスのあったものが先頭に置かれ、その一つ前にフォーカスのあったものがその次、という順に、ダイナミックに順序が変更されます。
W=作成された順	プログラムがメニューあるいはコールコマンドにより呼び出され、コンテキストが作成された順に、一覧に表示されます。

14.7.3. 新しいイベント

並行実行に関連して、次のようなイベントが追加されました。

以下のイベントは、**イベント実行** コマンドか、イベントタイプのメニューにより実行します。

イベント名	
次のウィンドウ	フォーカスが次のウィンドウに移動します。
前のウィンドウ	フォーカスが前のウィンドウに移動します。
他のウィンドウ	その他のウィンドウ ダイアログボックスが開き、ユーザがウィンドウ一覧からフォーカスを当てたいウィンドウを選択することができるようになります。
コンテキストを閉じる(C)	現在実行しているコンテキストに属する全てのプログラムを閉じます。他のコンテキストに属しているプログラムはそのままです。メインコンテキストは閉じることができません。
全て閉じる	全てのコンテキストを終了させます。ただし、メインコンテキストは閉じません。
プロジェクトを閉じる(C)	メインコンテキストを含む全てのコンテキストを終了させ、アプリケーションも終了します。

以下のイベントは、ウィンドウがフォーカスを得たとき、あるいは失ったときに、Magic 実行エンジンが発行するもので、プログラムではこれらのイベントに対するハンドラを定義して、フォーカスの移動時に必要な処理を定義することができます。

イベント名	実行エンジンにより発行されるタイミング
コンテキストがフォーカス取得(G)	現在実行中のプログラムのコンテキストにフォーカスが移った。 パラメータとして、直前までフォーカスのあったコンテキストの ID が渡ります。
コンテキストがフォーカス喪失(L)	現在実行中のプログラムのコンテキストからフォーカスが失われた。 パラメータとして、新たにフォーカスを得たコンテキストの ID が渡ります。

14.7.4. 関数

ウィンドウを指定して、フォーカスを移動させたい場合には、次の関数を使います。

シンタックス: SetWindowFocus (*Window Name*)

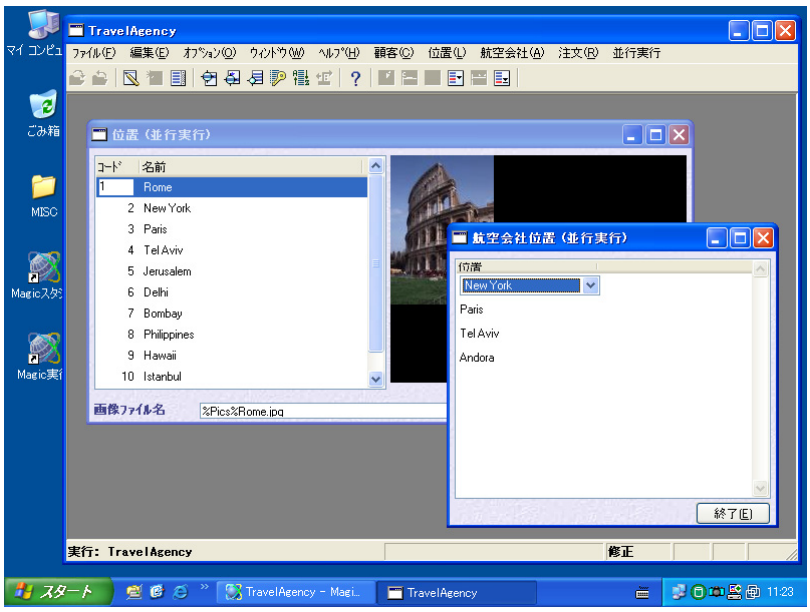
パラメータ: Window Name – フォーカスを設定したいウィンドウのタイトルを表す文字列。

戻り値: 指定したタイトルを持つウィンドウが存在して、フォーカスの移動に成功した場合には True が返り、そうでない場合には False が返ります。

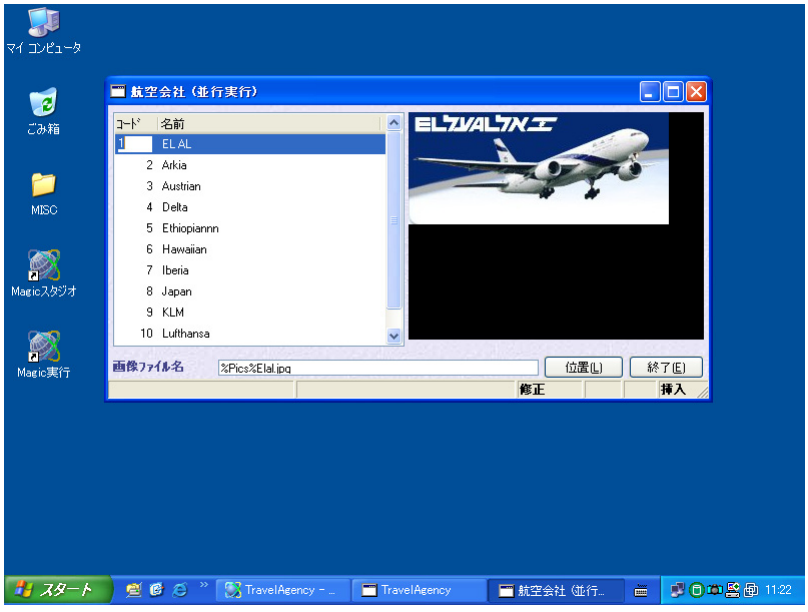
14.8. SDI ウィンドウ

オンラインプログラムの並行実行をより便利に活用するため、MDI（Multiple Document Interface）と SDI（Single Document Interface）というウィンドウ タイプが導入されました。

MDI というのは、Magic 実行エンジンのウィンドウがルートにあり、各タスクのフォームがその子タスクとして表示されるもので、従来の Magic のユーザインターフェースがそれにあたります。



SDI というのは、Magic の実行エンジンのウィンドウとは独立したウィンドウを使って、タスクのフォームが表示されます。



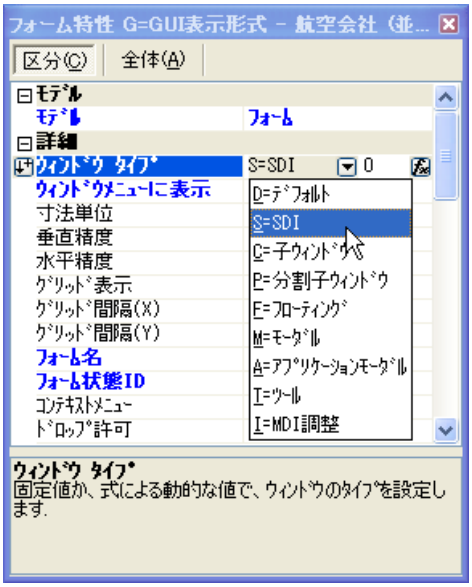
Magic V10 では、従来の MDI のインターフェースに加え、SDI のインターフェースを持ったプログラムも実現できるようになりました。以下に V10 のウィンドウタイプについて説明します。

14.8.1. ウィンドウタイプ特性

Windows 上では、次のようなウィンドウのタイプを利用することができます。

ウィンドウタイプ	意味
子ウィンドウ	二つのウィンドウが親子関係を持っている場合の、子の方のウィンドウ。子ウィンドウは親ウィンドウの内部だけに表示され、その枠の外に表示されることがありません。また、親ウィンドウの外に移動することができません。親ウィンドウの座標が移動すると、子ウィンドウもそれにつれて移動します。 親子関係を持つタスクで、子タスクで詳細行を表示する場合などに良く使われます。
分割ウィンドウ	分割線により、一つのウィンドウを上下または左右に分割して表示します。 親子関係を持つタスクで、上にヘッダ行のデータ、下に明細行データなどを表示したりします。
モーダルウィンドウ	他のウィンドウにフォーカスを移動することができないウィンドウです。 エラーメッセージを表示して注意を喚起したり、ユーザの入力や応答が必須であるような状況で使われます。
フローティングウィンドウ	実行エンジンのウィンドウの枠にとらわれずに移動することができます。
SDI ウィンドウ	フローティングと同様に、実行エンジンのウィンドウの枠にとらわれずに移動することができます。フローティングとは異なり、独自のアイコンやメニューバーを持つこともできます。また、Windows のタスクバーには、Magic の実行エンジンとは別に、自分自身のタスクバー ボタンを持つことができます。
通常	以上のいずれでもない場合、Magic タスクのウィンドウは、Magic 実行エンジンのウィンドウの内部にだけ表示されます。

V9Plus では、これらのウィンドウのタイプを、いくつか個別のパラメータによって設定していましたが、V10 では **ウィンドウ タイプ** という一つのパラメータに整理しました。



#	ウィンドウタイプ値	意味
1.	D=デフォルト	デフォルトの場合、ウィンドウがどのタイプで表示されるかは、呼び出したプログラムのウィンドウのタイプと同じウィンドウタイプで表示されます。例えば、フローティングウィンドウのタスクから呼び出された場合、フローティングウィンドウとなります。
2.	S=SDI	SDI ウィンドウとして、独立したウィンドウの枠をデスクトップ上に持って表示されます。SDI ウィンドウは、並行実行 フラグがセットされているプログラムでのみ有効です。並行実行 フラグがセットされていないプログラムで S=SDI を指定しても、D=デフォルト と同じとして扱われます。
3.	C=子ウィンドウ	子ウィンドウとして、ウィンドウが開きます。
4.	P=分割子ウィンドウ	分割ウィンドウの、子ウィンドウとして開きます。上下分割の場合には下側、左右分割の場合には右側に表示されるのが普通です。
5.	F=フローティング	フローティングウィンドウとして表示されます。
6.	M=モーダル	モーダルウィンドウとして表示されます。同一コンテキスト中の他のウィンドウに移動することはできませんが、別コンテキストのウィンドウに移動することはできます。
7.	A=アプリケーションモーダル	モーダルウィンドウと同様ですが、別のコンテキストのウィンドウに移動することもできなくなります。
8.	T=ツール	フローティングと同様、Magic 実行エンジンのウィンドウにとらわれずに移動することができます。フローティングと異なるのは、タイトルバーが通常よりも低く、文字も小さめの文字で表示されることです。
9.	MDI 調整	Magic 実行エンジン内に、最大化されて表示されます。

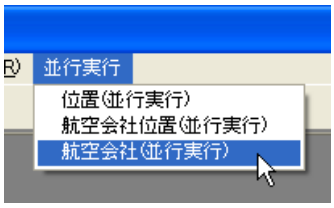
ウィンドウタイプ 特性は、式で指定することもできます。式で指定する場合には、数値型の式で指定し、上表の「#」欄の値を返すようにします。この式はタスクの開始時に評価され、タスクの途中で再評価されることはありません。

- **並行実行** パラメータがセットされているプログラムでは、**D=デフォルト** か **S=SDI** のいずれかを設定しなければなりません。
- ブラウザクライアント タスクでは、**D=デフォルト**と **M=モーダル** だけがサポートされています。

14.8.2. SDI ウィンドウの例

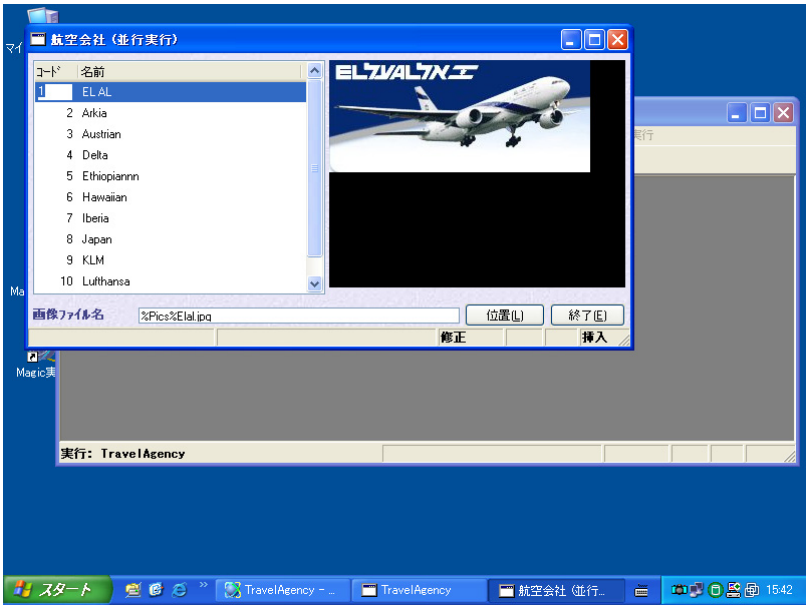
サンプルでは、航空会社(並行実行) プログラム(プログラム 18 番)が、SDI タイプのウィンドウとして定義されています。

1. **デバッグ** メニューから **プロジェクトの実行** (Ctrl+F7 キー) を選んで、実行モードにしてください。
2. **並行実行** メニューから、 **航空会社(並行実行)** を選びぶと、**航空会社(並行実行)** プログラムが開始します。

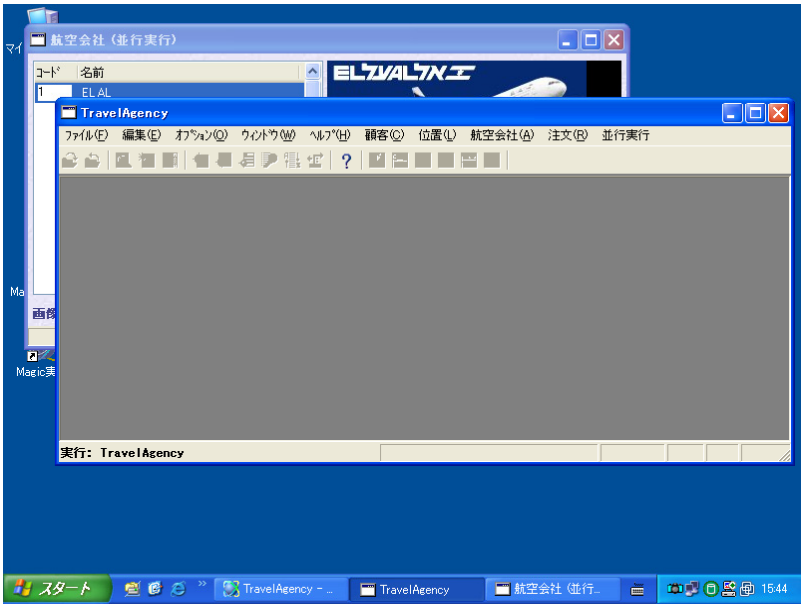


このとき、**航空会社(並行実行)** プログラムのウィンドウについて、次のようになっていることを確認してください。

- **航空会社(並行実行)** プログラムのウィンドウは、Magic 実行エンジンの枠の外に表示されています。

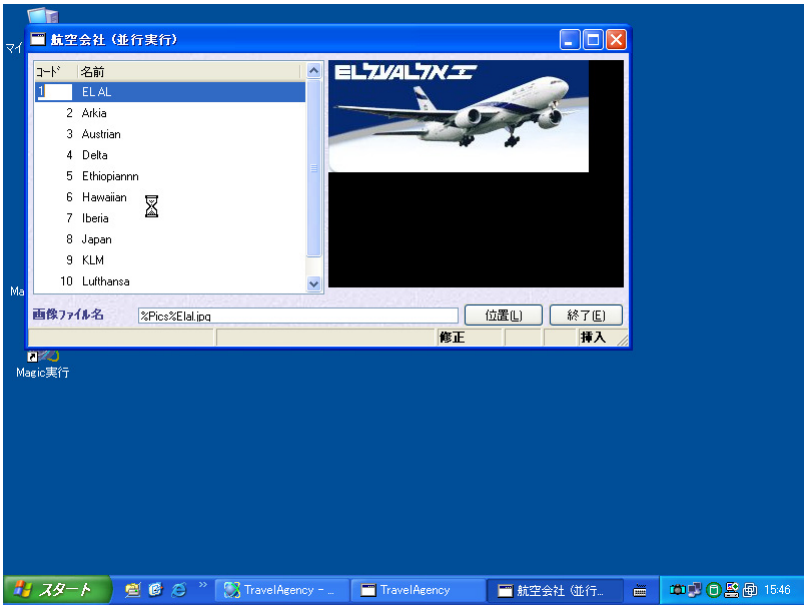


- Magic 実行エンジンのタイトルバーをクリックすると、Magic 実行エンジンのウィンドウが前面に表示されます。

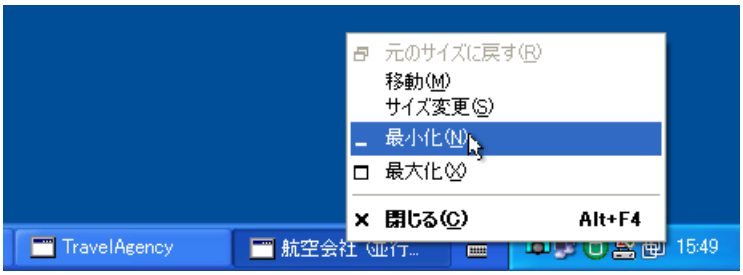


- Magic 実行エンジンの最小化ボタン

  を押すと、Magic 実行エンジンのウィンドウが最小化されますが、航空会社(並行実行）のウィンドウはそのままです。



- Windows のタスクバーには、Magic 実行エンジンのツールバー ボタン(**TravelAgency** という名前のもの)と並んで、**航空会社(並行実行)** というツールバー ボタンがあります。また、それぞれのツールボタンを右クリックすることで、システムメニューを表示させることができ、ウィンドウの移動、サイズ変更などを行うことができます。

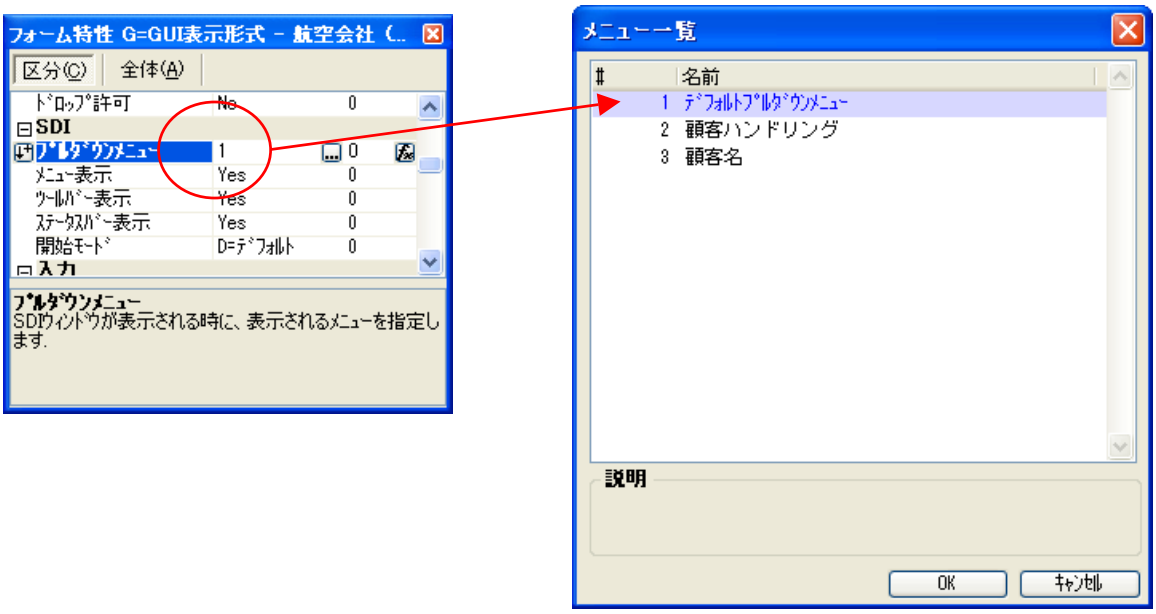


14.8.3. SDI ウィンドウのプルダウンメニュー

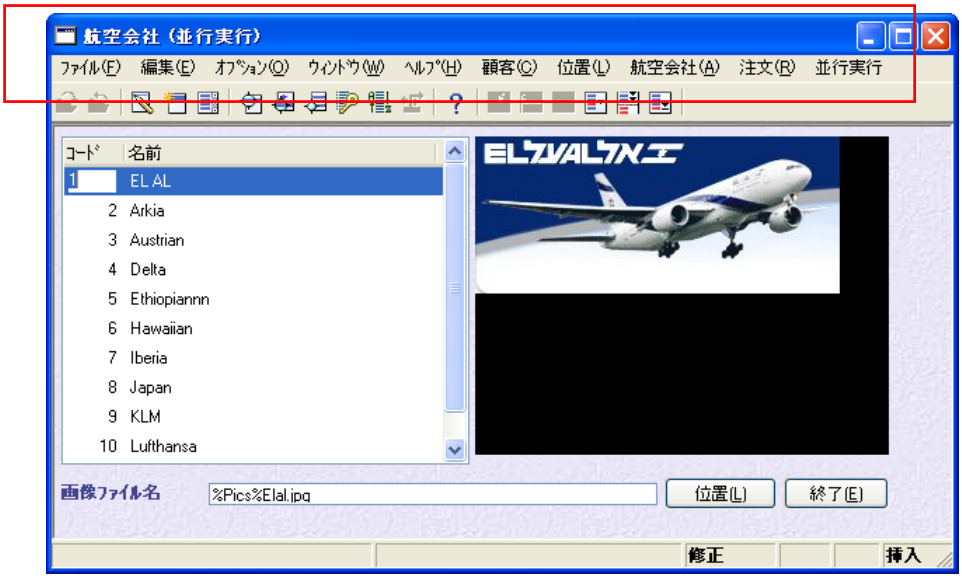
他のタイプのウィンドウとは異なり、SDI ウィンドウには、それ独自のプルダウンメニューを持たせることができます。プルダウンメニューは、フォーム特性の プルダウンメニュー 特性に、メニューリポジトリのエントリ番号を指定して設定できます。

例: 航空会社(並行実行) プログラム（プログラム 18 番)に、プルダウンメニューを付加したい場合には・・・

1. プログラムリポジトリを開き、プログラム 18 番 **航空会社(並行実行)** を開きます。
2. **フォーム** タブを開き、2 番目のフォーム**航空会社(並行実行)** にカーソルをおきます。
3. フォーム特性の **プルダウンメニュー** 特性(SDI カテゴリの中にあります）からズームします。
4. メニュー一覧が表示されるので、1 番 **デフォルトプルダウンメニュー** を選択します。
5. プログラムを閉じます。



このように設定してから、前と同様に、このプログラムを実行させてみてください。ウィンドウにプルダウンメニューが付加されています。



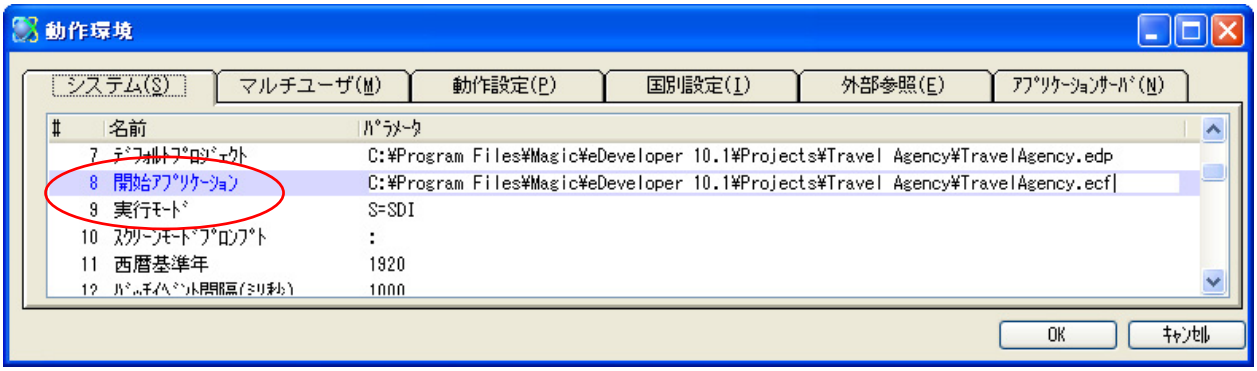
14.8.4. Magic のウィンドウを開かない SDI プログラム

今までは、Magic スタジオから実行モードにしてプログラムを実行していましたが、キャビネットファイルにすることにより、実行エンジンで直接アプリケーションを実行することができます。

このときに、Magic 実行エンジンのウィンドウ枠を表示させずに、SDI プログラムのウィンドウのみを表示させるようなインターフェースにすることもできます。このようにすると、アプリケーションが Magic で作っていることさえユーザからわからないようにできます。

Magic 実行エンジンのウィンドウを開かないアプリケーションを作成するには・・・

1. **動作環境** テーブルを開き、実行したいアプリケーションのキャビネットファイル（ECF ファイル）を **開始アプリケーション** パラメータに設定します。これにより、実行エンジンが起動したとき、自動的にこのアプリケーションが開始されるようになります。
2. **動作環境** テーブルを開き、**システム(S)** タブから **実行モード** パラメータを **S=SDI** に設定します。



実行モード パラメータは、実行エンジンの起動時に、ウィンドウはどのように表示されるかを決定するもので、V9Plus の **アプリケーション起動モード** パラメータに対応するものですが、V9Plus と比べて、V10 では次のように変更されています。

- 開発エンジン(スタジオ)と実行エンジンが分離されたので、**T=開発モード** というオプションがなくなりました。スタジオでプロジェクトを開いたときには、つねに開発モードで開きます。
- V9Plus での **R=実行モード** は、**O=オンライン** と名前が変わりました。これがインストール時のデフォルトの設定です。
- **B=バックグラウンド**は、V9Plus のときと同じです。ユーザインターフェースを使わない、マルチスレッドのアプリケーションサーバとして動作します。
- SDI ウィンドウがサポートされるようになったので、**S=SDI** というオプションが追加されました。

実行モード パラメータを **S=SDI** に設定すると、Magic 実行エンジンは次のように動作します。

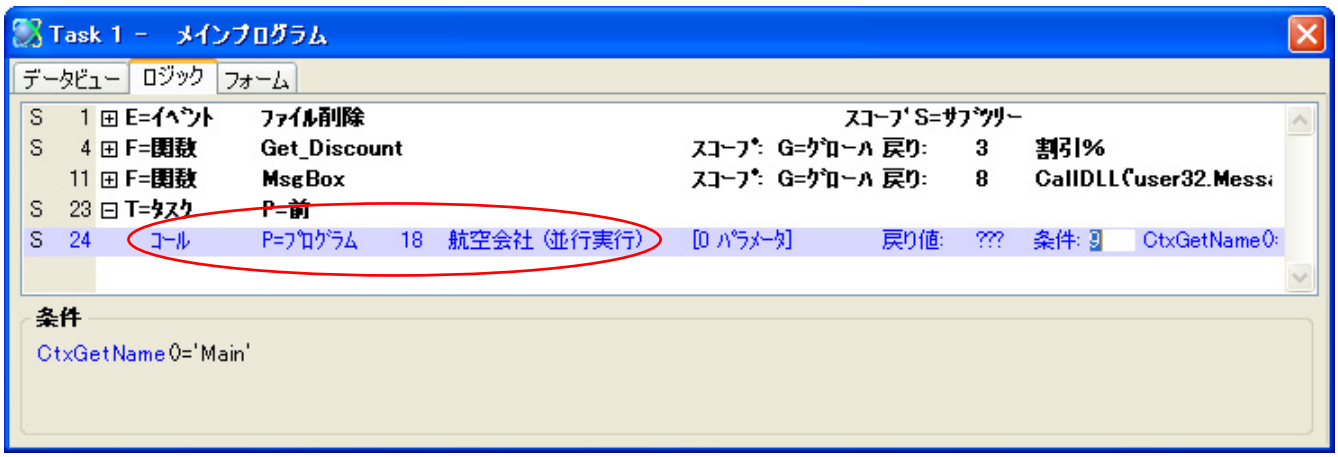
- Magic 実行エンジン自身のウィンドウ枠は表示しません。
- 開始アプリケーション で設定されたアプリケーション ECF ファイルは開かれ、アプリケーションが開始されます。
- メインコンテキストが作成され、メインプログラムが実行されます。
- ユーザからの入力を待ちます。

従って、Magic 実行エンジンのウィンドウを表示させず、アプリケーションのプログラムのみをユーザに表示するようなインターフェースとすることができるようになります。

3. メインプログラムで初期プログラムを呼び出す: **実行モード** パラメータを S=SDI に設定しただけでは、アプリケーションプログラムが開始されません。従って、メインプログラムのタスク前処理で、初期プログラム（SDI ウ

ィンドウを持つ）を呼び出す必要があります。

サンプルでは、メインプログラムのタスク前処理で、航空会社(並行実行）プログラム（プログラム 18 番）を、コールコマンドで呼び出しています。

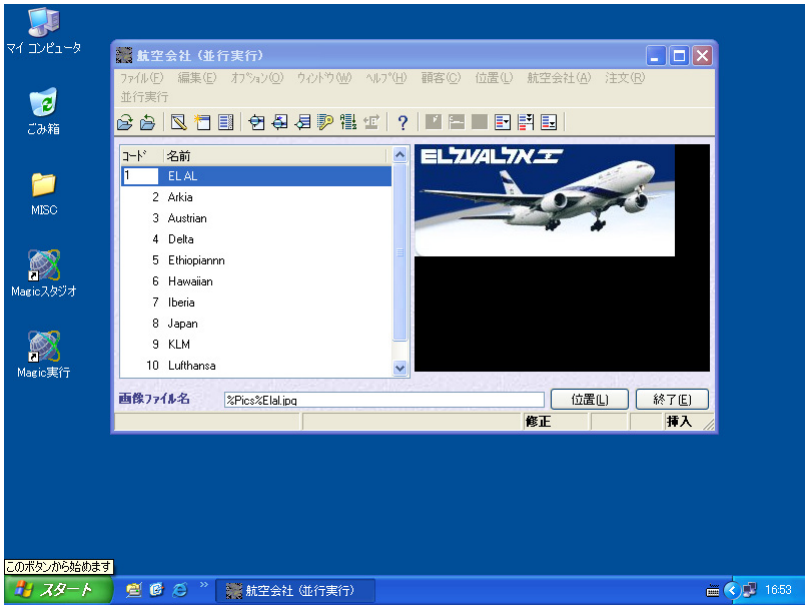


ここで、一つ注意しなければならないことがあります。このままでは無限ループに陥ってしまうからです。この理由と回避方法を以下に説明します：

実行モード パラメータが **S=SDI** に設定されていたら、実行エンジンは最初にメインコンテキストを作成します。このメインコンテキストの中でメインプログラムが実行され、そのタスク前処理で、SDI のプログラム **航空会社(並行実行)** が呼び出されます。
ところが、**航空会社(並行実行)** プログラムは、**並行実行** パラメータがセットされているので、新たにコンテキストを作成します。このコンテキストの中で、やはりメインプログラムが実行されます。すると、このメインプログラムのタスク前処理でも、**航空会社(並行実行)** プログラムが呼び出されます。従って、ここで新たなコンテキストが作成され、またまたメインプログラムが実行される、ということになります。
このように、コンテキスト作成 → メインプログラム実行 → 航空会社(並行実行) プログラム呼出 → コンテキスト作成 → ・・・ という処理が無限に続いてしまうことになります。最終的にはリソースを食い尽くして、システムはハングアップするか異常終了するようになります。

これを防ぐには、メインプログラムで **航空会社(並行実行)** プログラムを呼び出すコールコマンドは、最初に 1 回だけ行う、というように条件付けしてやる必要があります。条件式としては、「メインコンテキストでだけ True となる」というようにしておいてやればよいので、条件欄に、 `CtxGetName()='Main'` と設定してやります。

以上のようにプログラムを変更してから、キャビネットファイルを作成しなおして、実行エンジンを起動してください。Magic の実行エンジンのウィンドウ枠は表示されず、航空会社(並行実行) のウィンドウのみが表示されます。


マイ コンピュータ
ごみ箱
MISC
Magicスタジオ
Magic実行
航空会社（並行実行）
ファイル(F) 編集(E) オプション(O) ウィンドウ(W) ヘルプ(H) 顧客(C) 位置(L) 航空会社(A) 注文(R)
並行実行
コード 名前
1 EL AL
2 Arkia
3 Austrian
4 Delta
5 Ethiopiannn
6 Hawaiian
7 Iberia
8 Japan
9 KLM
10 Lufthansa
画像ファイル名
%Pics%Elal.jpg
位置(L)
終了(E)
修正
挿入
このボタンから始めます
スタート
航空会社（並行実行）
16:53

14.9. コンテキスト関連のイベント

14.9.1. コンテキストを閉じる

コンテキストを閉じる(C) というイベントを実行すると、現在のコンテキストに含まれるすべてのタスクが終了されます。ただし、メインコンテキストは閉じることはできません。

14.9.2. アプリケーションクローズ

アプリケーション クローズ(C) というイベントを実行すると、メインコンテキストを含むすべてのコンテキストが閉じられ、アプリケーションも閉じられます。

15. デバッガ

デバッガとは、アプリケーションのエラーを修正するために、開発者がアプリケーションを実行しながら、実行エンジンの状態やフローの流れなどを確認したり、手動で強制的に変更したりすることのできる、スタジオの機能です。V10 では、開発用のスタジオと、実行用のエンジンが分離されたため、デバッガが非常に使いやすく、かつ堅牢になっています。
本章では、以下のような V10 でのデバッガの基本機能について解説します。

- ブレークポイントを設定する。
- ブレーク時にタスクの項目の値を確認する。
- ブレークポイントに条件を設定する。
- ワッチリストを作成し、実行時に特定の項目の値を確認する。
- 実行時に項目の値を手動で変更する。
- スタジオ内で、実行フローを確認する。
- コールスタックを見る。
- 複数のコンテキストの実行状況を見る。

本章の内容の詳細については、リファレンスマニュアルの以下の箇所を参考にしてください。

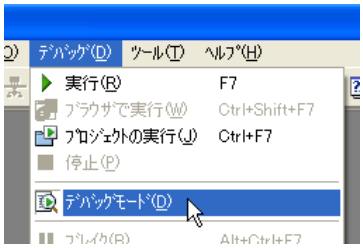
デバッガ一般	**アプリケーションのテスト → デバッガ**

15.1. デバッガを開始する

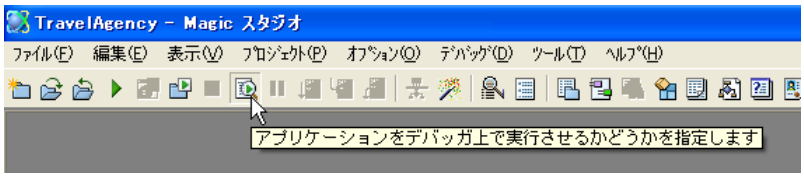
デバッガを利用するには、Magic スタジオが **デバッグモード** で動作している必要があります。

デバッグモードに入るには・・・

デバッグモード は、メニュー **デバッグ(D)** から **デバッグ モード(D)** を選択します。



デバッグモードは、ツールバーで設定することもできます。



デバッグモードを解除するには・・・

デバッグモード を解除するには、同じく **デバッグ(D)** メニューから **デバッグモード(D)** を再度選択します。
このメニューは、選択するたびにデバッグモードがトグルします。

15.2. アクティビティ モニタ

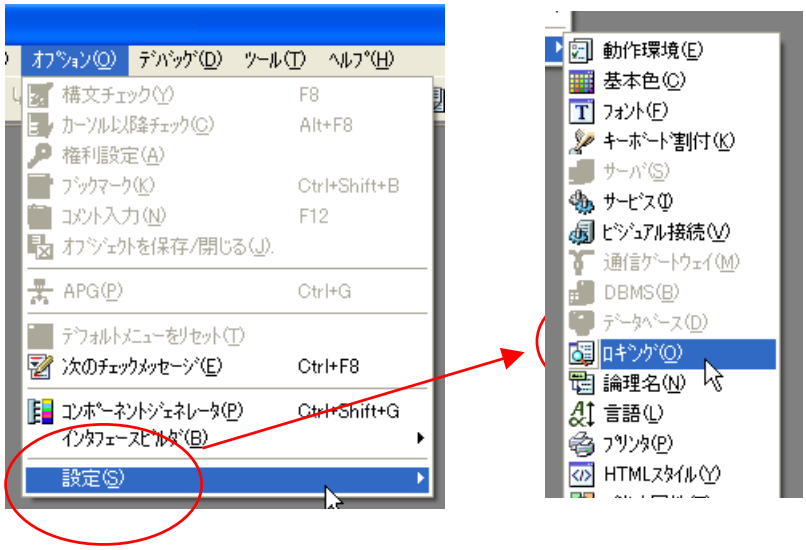
アクティビティモニタは、V9Plus で **フローモニタ** と呼んでいたものと同じもので、Magic プログラムの実行フローのトレースをとります。スタジオの画面上に表示することもできますし、ファイルに記録するように設定することもできます。

15.2.1. ロギング フィルタ

ロギング フィルタは、フローモニタに表示させる内容を制限するもので、**ロギング** ダイアログで設定します。

ロギング ダイアログの表示

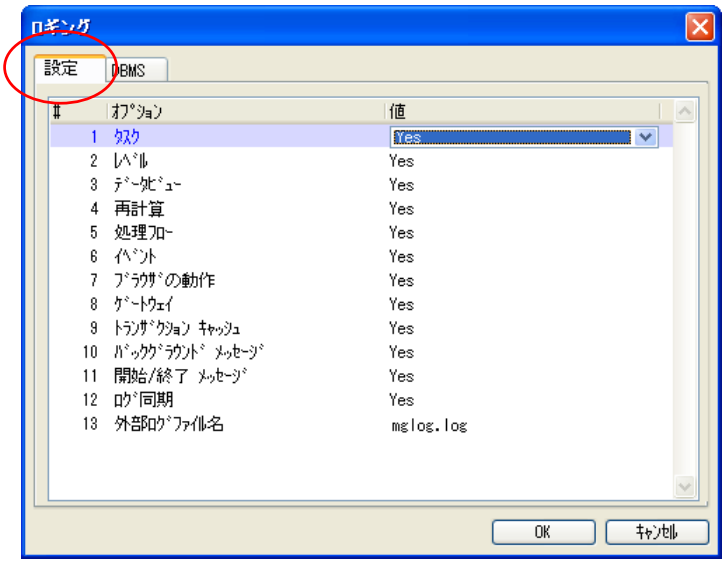
ロギング ダイアログ は、メニュー **オプション(O)** から **設定(S)** → **ロギング(O)** と選択します。



ロギング ダイアログには、**設定** と DBMS という二つのタブがあります。

設定 タブ

設定 タブを開くと、フローモニタに表示するトレースのレベルを設定できます。この設定内容は、V9Plus のフローモニタでの設定とほぼ同様です。

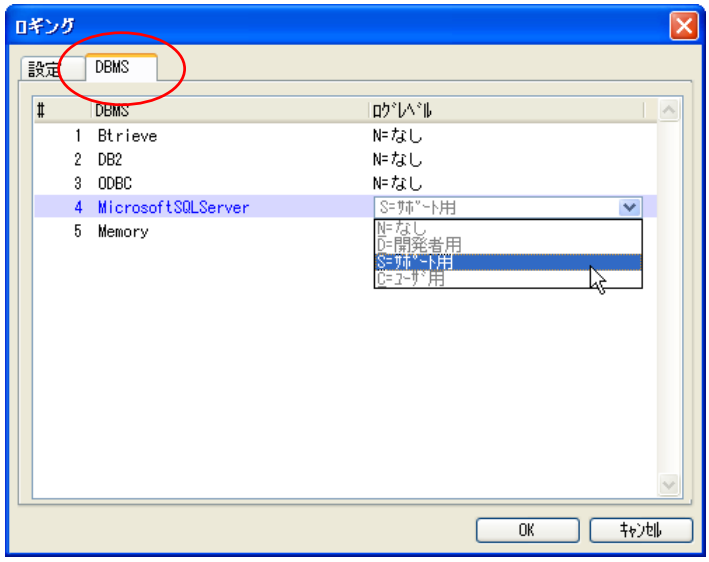


DBMS タブ

DBMS タブには、現在ゲートウェイがロードされている DBMS の一覧が表示され、それぞれにログレベルを設定できるようになっています。

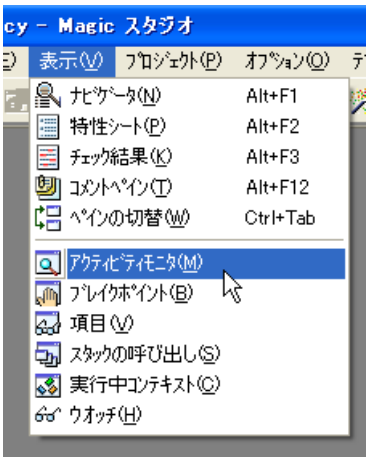
V9Plus では、データベースゲートウェイのログは DBMS テーブルで設定していて、設定を変更した場合、変更を反映させるには Magic を再起動する必要がありました。

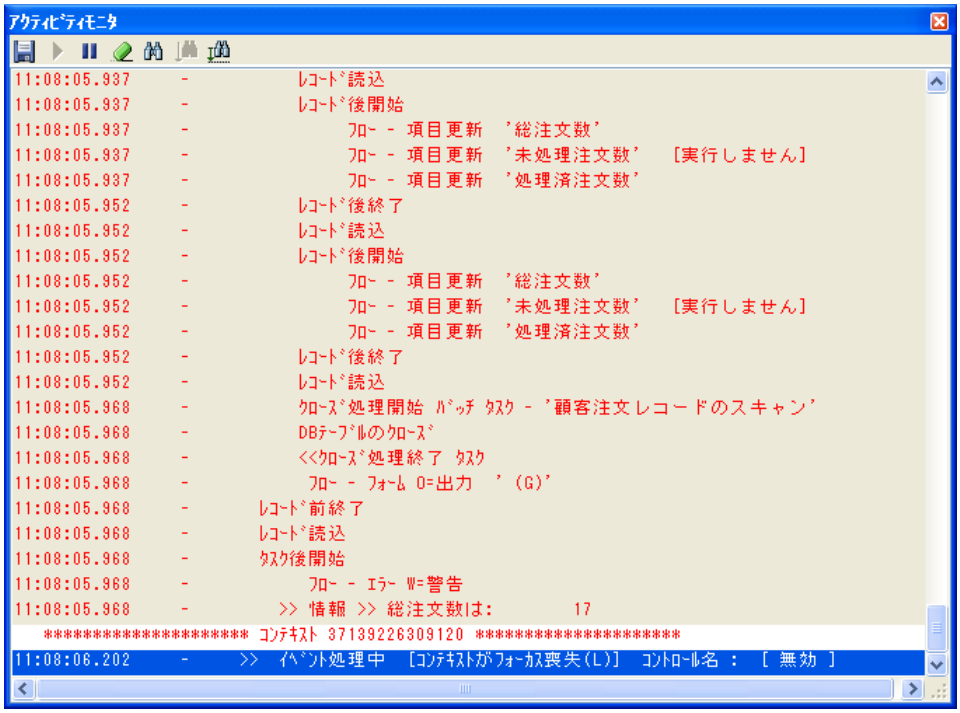
V10 では、すべてのデータベースゲートウェイのログの設定がこの DBMS タブで行われるようになり、また、設定を変更した場合には即座に変更が反映され、Magic の再起動が不要になりました。



15.2.2. アクティビティ モニタ

アクティビティモニタは、**表示(V)** メニューから **アクティビティモニタ(M)** を選択すると表示されます。





アクティビティモニタにはツールバーがあり、次のような機能を持っています。これらの機能は、アクティビティモニタのコンテキストメニューからも実行できます。

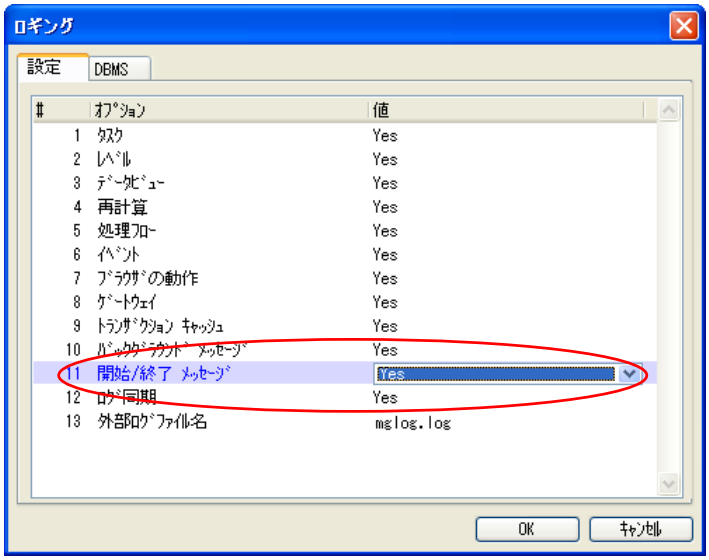
ツールバー

コンテキストメニュー

	コンテキストメニュー	機能
	保存(S)	ログの内容を、テキストファイルに書き出します。
	出力開始(O)	ログ出力を開始します。
	出力停止(U)	ログ出力を中断します。
	クリア(C)	ログ内容を消去します。
	検索(F)	ログの内容から文字列検索を行います。
	再検索(N)	次の文字列検索を続けます。
	開始/終了チェック(M)	対応する **開始** あるいは **終了** に移動します。

最後の **開始/終了チェック(M)** は、ロギング ダイアログで、**開始/終了メッセージ** が Yes になっている場合にだけ有効です。

例えば、アクティビティモニタ上で、**レコード後開始** の行にカーソルがあった場合、**開始/終了チェック(M)** ボタンを押すと、対応する **レコード後終了** の行にカーソルが移動します。



15.2.3. Logging 関数

Magic V10 では、Logging 関数を使って、記録するログの内容を、プログラムから動的に変更することができるようになりました。

この関数を実行すると、即時に変更内容が有効になります。MAGIC.INI やロギング ダイアログでの設定には影響を与えません。

構文: Logging (開始/停止, フィルタ)

パラメータ:

- 開始/停止 … フィルタの開始／停止を指定する論理値
  - True…フィルタを開始します。
  - False … フィルタを停止します。
- フィルタ(文字) … フィルタオプションを表す文字列。以下のオプションが指定できます。

Task	Levels	DataView
Recompute	Flow	Events
LogBrowser	Gateway	TransCache

BackgroundMsg	BeginEndMsg	LogSynch
ALL	RESET	

ExecutionLogFileName=(ファイル名)

Btrieve= (N,D,S,C のどれか)

DB2400= (N,D,S,C のどれか)

Oracle= (N,D,S,C のどれか)

AS400= (N,D,S,C のどれか)

DB2= (N,D,S,C のどれか)

ODBC= (N,D,S,C のどれか)

MicrosoftSQL = (N,D,S,C のどれか)

Memory= (N,D,S,C のどれか)

戻り値: 論理値 … 処理が成功した場合「True」が返ります。

例: Logging ('FALSE'LOG,'Oracle')

注意事項:

- [フィルタ]の指定が正しくない場合、「False」が返ります。
- 「ExecutionLogFileName」で外部ログファイル名が設定され、ファイルが作成できなかった場合、「False」が返ります。メッセージは mgerror.log に書き込まれます。(ログファイル名は、MAGIC.INI ファイルの GeneralErrorLog パラメータで変更できます。)
- 外部ログが設定されず、デバッグモードも設定されていない場合、「False」が返ります。
- [開始/停止]パラメータが「True」で DBMS パラメータが「N」と評価された場合(例:'True','ODBC=N')、ゲートウェイオプションは無視され、「False」が返ります。
- 開発エンジンがデバッグモードでない場合、(指定されていれば)外部ログファイルのみに反映されます。
- この関数は、MAGIC.INI を更新しません。設定された値は、現在のコンテキストでのみ有効です。開発エンジンに戻ると、値は MAGIC.INI に設定されているデフォルト値に戻ります。
- [フィルタ]パラメータの「ALL」が停止(False)とともに使用された場合、全ての値は False に設定されます。
- [フィルタ]パラメータの「ALL」が開始(True)とともに使用された場合、全ての非ゲートウェイ値は True に設定されます。
- 「RESET」フィルタは、INI の値を全てリセットします。この場合、第一パラメータは、True とします。False が設定された場合、無視され「False」が返ります。

15.3. ブレイクポイント

ブレイクポイントは、プログラム中に開発者が設定するもので、プログラムが実行時にその位置に到達したら、エンジンはプログラムの実行を一時的に中断(ブレイク)し、開発者がそのときのプログラムの状態を調査することができるようになります。

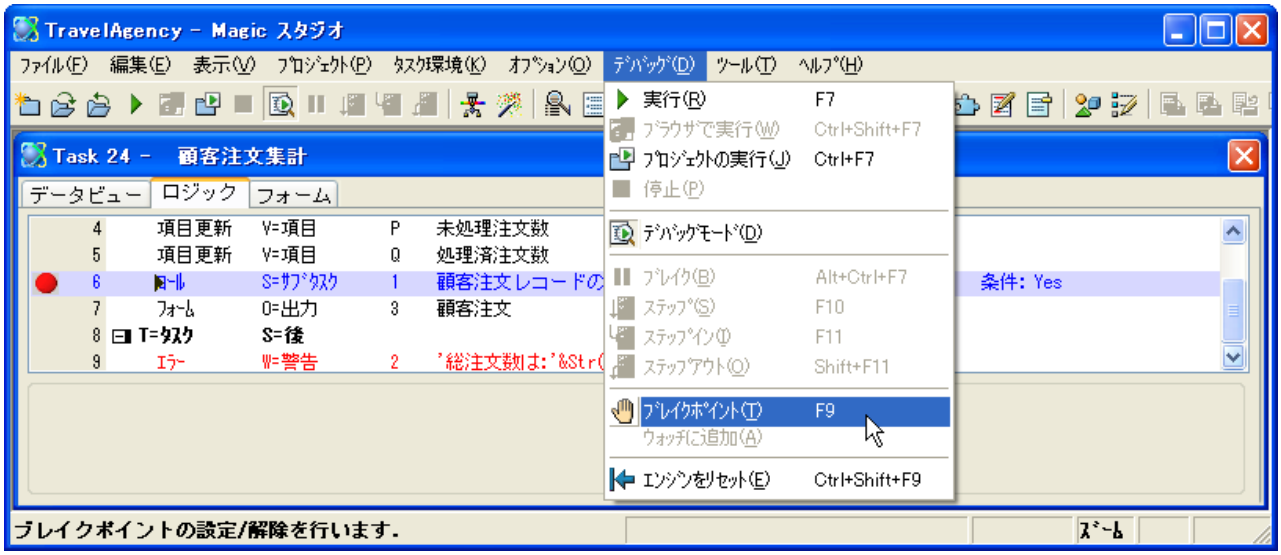
15.3.1. ブレイクポイントを追加する

Magic V10 でのブレイクポイントは、プログラムリポジトリ中の任意のタスクのデータビューエディタ、あるいはロジックエディタ上の行に設定することができます。ただし、コメント行には設定することはできません。

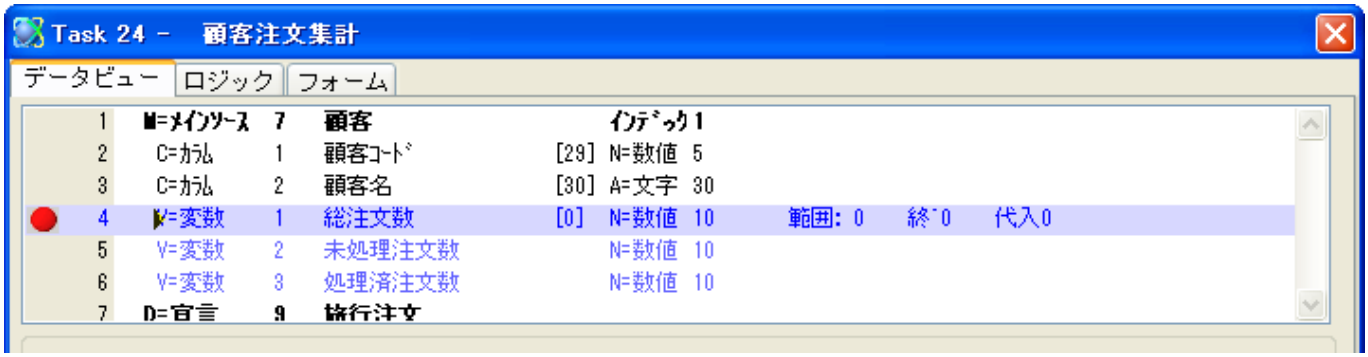
ブレイクポイントを設定するには・・・

1. データビューエディタ、あるいはロジックエディタを開きます。
2. ブレイクポイントを設定したい行にカーソルを置きます。
3. **デバッグ(D)** メニューから、**ブレイクポイント(T)** を選びます。F9 キーで設定することもできます。
   - ブレイクポイントが設定された行には、左側に赤色の ● 印が表示されます。
   - ブレイクポイントの設定は、**デバッグ(D)→ ブレイクポイント(T)** メニュー（あるいは **F9** キー)によりトグルします。

下図は、ロジックエディタにブレイクポイントが設定された例です。



次の図は、データビューエディタにブレイクポイントが設定された例です。



ブレイクポイントの設定は、プロジェクトファイルに記録されます。このため、プロジェクトを閉じた後に再度開いたときにも、前に設定されたブレイクポイントはそのまま有効になっています。

15.3.2. プログラムを実行する

ブレイクポイントの設定されているプログラムを実行すると、次のときにプログラムの実行がブレイク(一時中断)します。

- データビューエディタに設定されているブレイクポイントの場合には、ブレイクポイントの設定されている行で定義されている項目の値に変更があったとき。
- ロジックエディタにブレイクポイントが設定されている場合には、ブレイクポイントの設定されている行まで実行が進んだとき。

ブレイクポイントから実行を再開するには、**デバッグ(D)** メニューから、次のようなオプションを選択できます。

メニュー	キー	意味
継続	F7	実行を継続する。
停止		プログラムを強制的に終了する。
ステップ	F10	現在の行を実行し、次の行で再度ブレイクする。
ステップイン	F11	ステップ と同じだが、現在の行がコールコマンドやイベント実行コマンドの場合、あるいは開発者定義関数を含む場合には、それにより呼ばれるプログラム、タスク、イベントハンドラ、関数などの最初の行まで進んだところで再度ブレイクする。
ステップアウト	Shift+F11	現在実行中のタスク、ハンドラなどを最後まで実行し、呼出元に戻った時点で再度ブレイクする。

15.3.3. 実行中に強制的にブレイクする

プログラムの実行に予想以上に時間がかかる場合とか、期待していた動作でない場合には、強制的にブレイクをかけて、現在のプログラムの状態を調査したい場合があります。

このような場合には、プログラム実行中に、**デバッグ(D)** メニューから、**ブレイク(B)** を選択します。その時点で強制的にブレイクが起こり、スタジオでは、実行中のタスクが自動的に開いて、以下のように、現在実行中の行にカーソルが移動します。

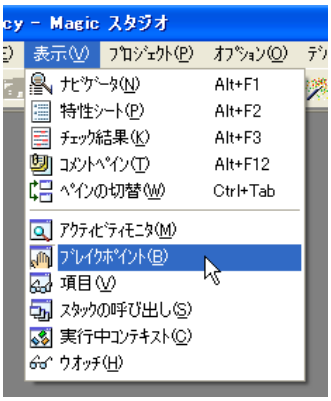
- オンラインタスクでユーザのデータ入力待ちの場合には、データビューエディタが開き、現在フォーカスのある項目に対応する **C=カラム** あるいは **V=変数** などの行のカーソルが移動します。
- その他の場合(バッチタスクの実行中、あるいはオンラインタスクでもタスク前・後処理やレコード前・後処理実行中の場合など)には、ロジックエディタ上で、現在実行中の行にカーソルが移動します。

15.3.4. ブレイクポイントリポジトリ

設定したブレイクポイントは、**ブレイクポイント リポジトリ** で一覧表示できます。

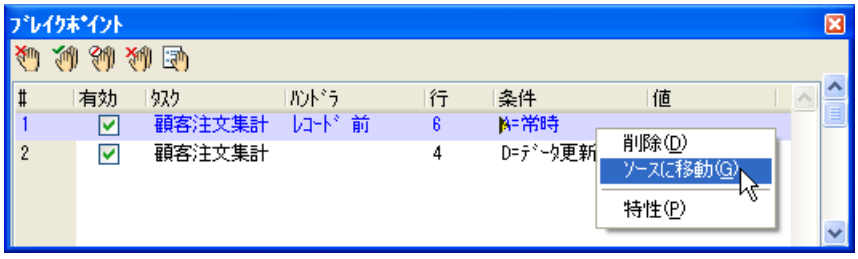
ブレイクポイント リポジトリは、**表示(V)** メニューから、**ブレイクポイント(B)** を選択すると表示されます。

ここには、設定されたブレイクポイントが一覧で表示されます。



有効 欄は、ブレイクポイントの有効性を示すチェックボックスです。開発者はチェックボックスをクリックすることにより、設定をトグルさせることができます。**有効** にチェックが入っていないブレイクポイントは一時的に無効となったブレイクポイントで、設定された行まで実行が進んでもブレイクはかかりません。

ブレイクポイント リポジトリで、右マウスクリックすると、コンテキストメニューが表示されます。



メニュー項目	意味
削除(D)	ブレイクポイントを削除します。
ソースに移動(G)	ブレイクポイントの設定されているタスクが開き、その行にカーソルが移動します。
特性(P)	ブレイクが起こる詳細な条件を設定することができます。詳しくは、後述の「条件付ブレイクポイント」で説明します。

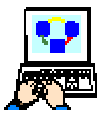
ブレイクポイントオプション

ブレイクポイントリポジトリにはツールバーがあり、次のような操作を行うことができます。

アイコン	対応するメニュー	機能
	削除(D)	現在カーソルのある行のブレイクポイントが削除されます。
		現在設定されているすべてのブレイクポイントが削除されます。
		すべてのブレイクポイントが有効化されます。
		すべてのブレイクポイントが無効化されます。
	特性(P)	ブレイクが起こる詳細な条件を設定することができます。詳しくは、後述の「条件付ブレイクポイント」で説明します。

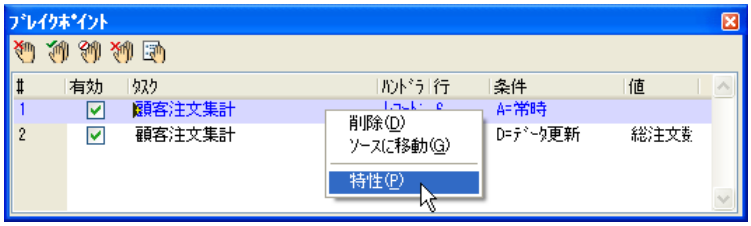
条件付ブレイクポイント

ブレイクポイントリポジトリでは、各ブレイクポイントに対し、ブレイクする条件を設定することができます。

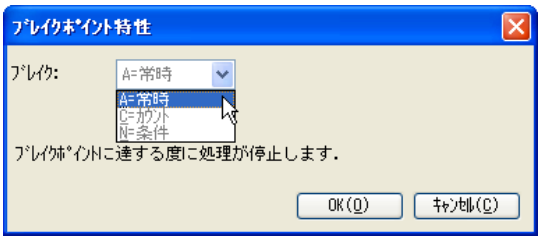
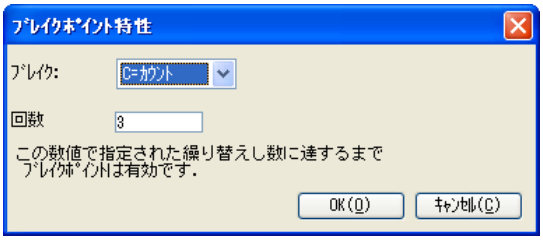
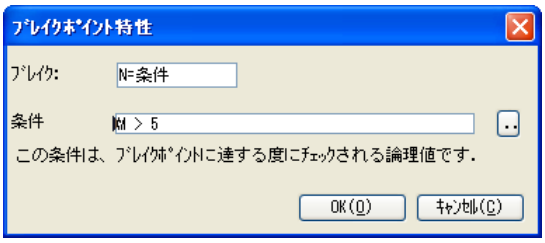


ブレイクポイントの条件を設定するには・・・

1. ブレイクポイントリポジトリで、条件を設定したいブレイクポイントにカーソルを合わせます。
2. 右マウスボタンをクリックし、コンテキストメニューから **特性(P)** を選びます。



3. **ブレイクポイント特性** ダイアログが開きますので、ここでブレイクの条件を、次のオプションから選んで設定します。

ブレイク 値	意味	ダイアログイメージ
A=常時	(デフォルト) ブレイクポイントに達したら常にブレイクします。	
C=カウント	**回数** 欄に設定された回数だけ、このブレークポイントを通過するたびにブレイクします。	
N=条件	**条件** 欄に式を設定し、ブレイクポイントに達したときに、この式を評価した結果、真になった場合にのみブレイクします。 **条件** 欄に設定するには、プログラムリポジトリで、ブレイクポイントの設定されているタスクを開いておく必要があります。その状態で、**条件** 欄からズームして、式テーブルで設定します。	

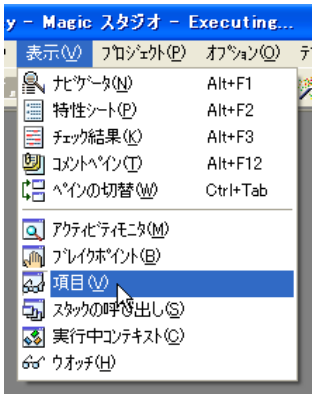
条件は、ロジックビューで定義されたブレイクポイントに対してだけ設定することができます。データビューに設定されたブレイクポイントには、条件を設定することはできません。

15.4. 項目一覧

項目 オプションでは、現在実行中のコンテキスト中の、項目の値を調べ、必要に応じて強制的に変更することができます。

15.4.1. 項目一覧の表示

項目 一覧は、**表示(V)** メニューから **項目(V)** を選択して表示させます。



右図に示すように、現在有効な項目名、型、データソース名、現在の値が表示されます。

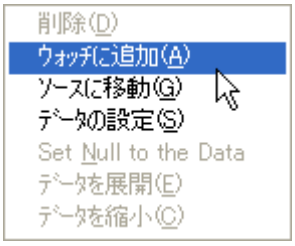
項目 一覧は、プログラムがブレイクしている状態でだけ有効です。

15.4.2. 項目一覧のオプション

項目一覧画面では、次のようなオプションが、右マウスボタンのコンテキストメニュー、あるいはツールバーから操作することができます。

ツールバー

コンテキストメニュー



アイコン	メニュー	意味
	ウォッチに追加	この変数を、ウォッチリスト に追加します。ウォッチリスト については、 **15.5 ウォッチリスト** を参照してください。
	ソースに移動	この変数が定義されている、データビューエディタの行に移動します。
	データの設定	この項目の値を手動で変更します。
	データに Null を設定	この項目の値を、Null に設定します。

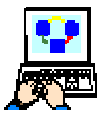
データの設定(S) あるいは **データを Null に設定(N)** によって項目の値が変更された場合には、ビューの再計算は起こりません。

15.5. ウォッチリスト

ウォッチリスト は、項目一覧と同様、項目の現在の値を調査したり設定したりするものですが、項目一覧の場合、現在有効な項目がすべて表示されるのに対し、ウォッチリストは、開発者が指定した項目だけを表示するところが異なります。

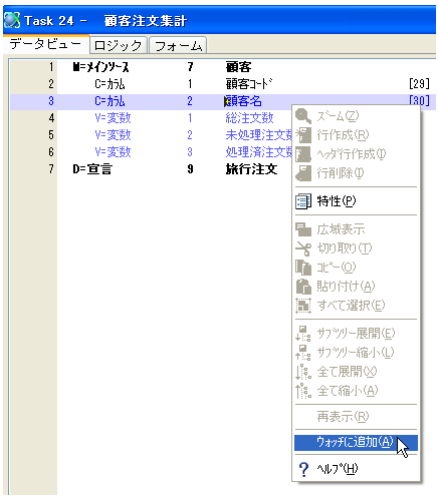
15.5.1. 項目をウォッチに追加する

ウォッチリストは、初期状態では空になっています。



ウォッチリストに項目を追加するには、次のいずれかによります。

- プログラムリポジトリのデータビューエディタで項目定義をしている行にカーソルを置き、右マウスボタンでコンテキストメニューを出して、**ウォッチに追加(A)** を選びます。

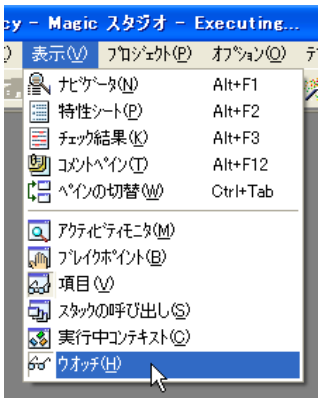


- 項目一覧 から、コンテキストメニューで **ウォッチに追加(A)** を選択します。

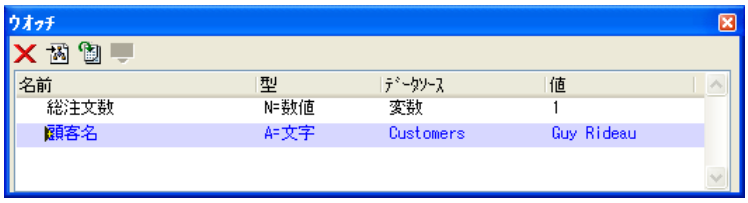
ウォッチリストでは、項目一覧と同じように、**ソースに移動**、**データの設定**、Null **を項目に設定**、などの操作を、ツールバー、あるいはコンテキストメニューから行うことができます。

15.5.2. ウォッチリストの表示

ウォッチリストは、**表示(V)** メニューから **ウォッチ(H)** を選んで表示させます。



ウォッチリストには、項目一覧と同じように、項目名、型、データソース名、および、現在の値が表示されます。



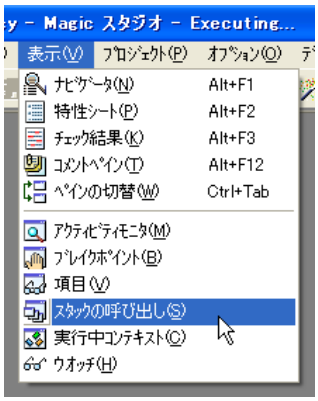
項目一覧の場合と同様、これらの表示は、プログラムがブレイクしている場合にのみ有効です。

15.6. コールスタック

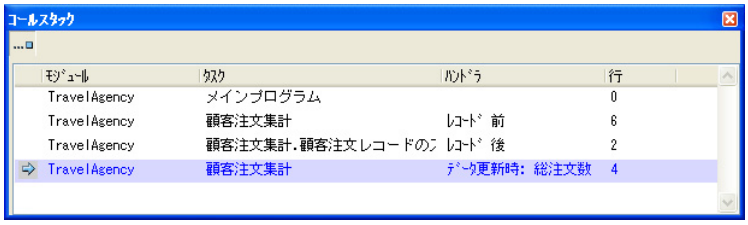
コールスタックは、メインプログラムから現在実行中のプログラムまでの、呼び出しの経路を表示します。

15.6.1. コールスタックの表示

コールスタックは、**表示(V)** メニューから **コールスタック(S)** を選んで表示させます。



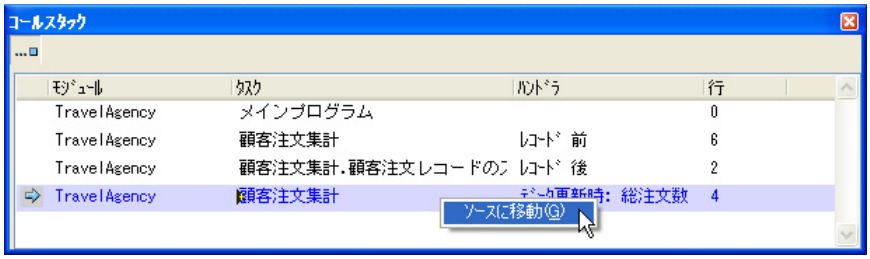
コールスタック ウィンドウには、モジュール名(プロジェクト名)、タスク名、ハンドラ名、およびそのハンドラ中での行数が表示されます。



ここでの表示は、プログラムがブレイクされて、実行が中断している状態でのみ有効です。

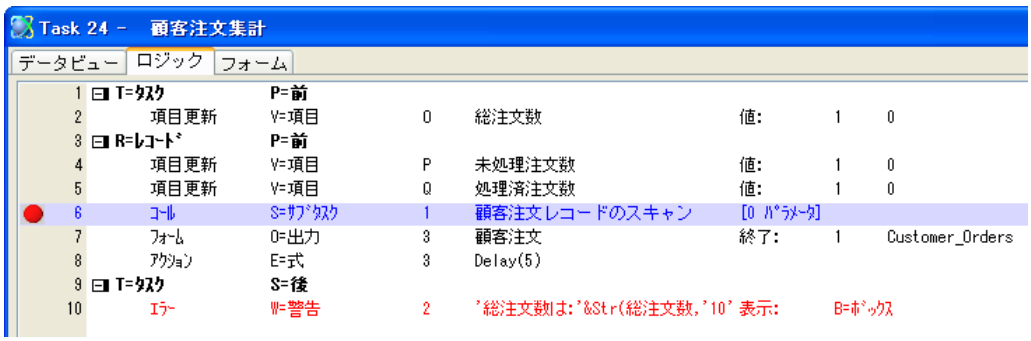
15.6.2. コールスタックのオプション

コールスタックウィンドウでは、コンテキストメニューから **ソースに移動(G)** を選ぶと、そのタスクで現在実行中の行を、プログラムリポジトリ上で表示します。



オンラインプログラムで、ユーザからのデータ入力待ちの状態のとき、あるいは、データビュー上でカラムや変数にブレイクポイントが設定されている場合にブレイクがかかったときには、**ソースに移動(G)** を行うと、データビューエディタが開き、その項目を定義している行が表示されます。

ロジックエディタ設定されているブレイクポイントでブレイクしたときには、**ソースの移動(G)** を行うと、ロジックエディタが開き、ブレイクの起こった行が表示されます。

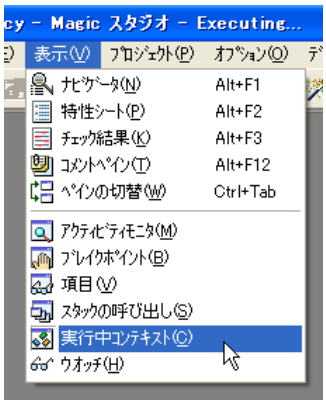


15.7. 実行コンテキスト

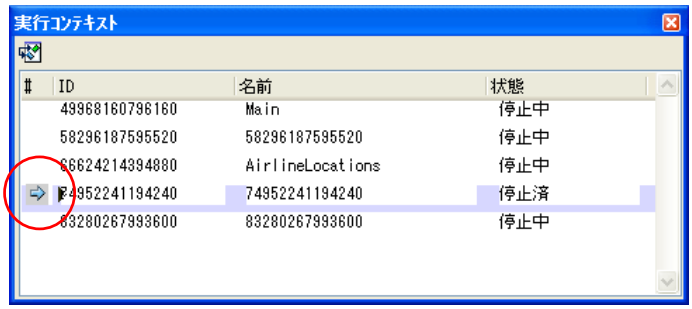
V10 の実行エンジンはマルチスレッドで並行実行をサポートしています。
並行実行を使わないアプリケーションならば、実行コンテキストは一つなので、コンテキストの選択について気を使う必要はありませんが、並行実行を使うアプリケーションでは、複数のコンテキストが同時に実行されていることがあります。
今まで見てきたコールスタック、項目一覧などは、各コンテキスト毎に独立して存在するものなので、正しい値を調査するには、正しいコンテキストを選択することが必要です。

15.7.1. 実行コンテキストの表示

実行コンテキストは、**表示(V)** メニューから、**実行コンテキスト(C)** を選択して表示させます。



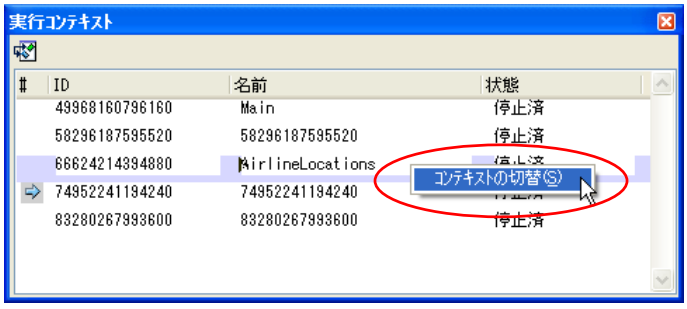
実行コンテキスト ウィンドウには、コンテキスト ID (Magic 実行エンジンが内部で管理している番号)、コンテキスト名 (CtxSetName 関数で設定された名前)、およびコンテキストの状態が表示されます。
右図は、TravelAgency プロジェクトを実行して、いくつかの並行実行プログラムを起動した状態で、ブレイクをかけたときの **実行コンテキスト** ウィンドウの例です。



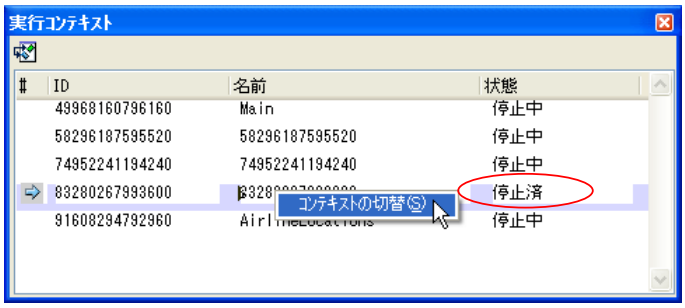
この中で、# の欄に矢印 が表示されている行がありますが、これは現在選択されているコンテキストを表しています。項目一覧、ウォッチリスト、コールスタックなどの表示は、現在選択されているコンテキストのものが表示されます。

15.7.2. コンテキスト切替え

選択されているコンテキストを切り替えるには、実行コンテキスト ウィンドウのコンテキストメニューあるいはツールバーから、**コンテキストの切り替え(S)** を選択します。
コンテキストを切り替えると、項目一覧、ウォッチリスト、コールスタックなどの表示が更新され、新しく選択されたコンテキストについての情報が表示されるようになります。



コンテキストを切り替えるには、状態が **停止済** になっている必要があります。状態が **停止中** になっている場合には、いったん、**停止済** 状態にあるコンテキストに対して、**コンテキストの切り替え(S)** をおこなってください。停止可能な状態であれば、**停止中** が **停止済** に変更されます。



付録 A. 利用可能なコードページの一覧

Code Page	Description
377	IBM EBCDIC - U.S./Canada
437	OEM - United States
500	IBM EBCDIC - International
708	Arabic - ASMO 708
709	Arabic - ASMO 449+, BCON V4
710	Arabic - Transparent Arabic
720	Arabic - Transparent ASMO
737	OEM - Greek (formerly 437G)
775	OEM - Baltic
850	OEM - Multilingual Latin I
852	OEM - Latin II
855	OEM - Cyrillic (primarily Russian)
857	OEM - Turkish
858	OEM - Multlingual Latin I + Euro symbol
860	OEM - Portuguese
861	OEM - Icelandic
862	OEM - Hebrew
863	OEM - Canadian-French
864	OEM - Arabic
865	OEM - Nordic
866	OEM - Russian
869	OEM - Modern Greek
870	IBM EBCDIC - Multilingual/ROECE (Latin-2)
874	ANSI/OEM - Thai (same as 28605, ISO 8859-15)
875	IBM EBCDIC - Modern Greek
932	ANSI/OEM - Japanese, Shift-JIS
936	ANSI/OEM - Simplified Chinese (PRC, Singapore)
949	ANSI/OEM - Korean (Unified Hangeul Code)
950	ANSI/OEM - Traditional Chinese (Taiwan; Hong Kong SAR, PRC)
1026	IBM EBCDIC - Turkish (Latin-5)
1047	IBM EBCDIC - Latin 1/Open System
1140	IBM EBCDIC - U.S./Canada (037 + Euro symbol)

1141	IBM EBCDIC - Germany (20273 + Euro symbol)
1142	IBM EBCDIC - Denmark/Norway (20277 + Euro symbol)
1143	IBM EBCDIC - Finland/Sweden (20278 + Euro symbol)
1144	IBM EBCDIC - Italy (20280 + Euro symbol)
1145	IBM EBCDIC - Latin America/Spain (20284 + Euro symbol)
1146	IBM EBCDIC - United Kingdom (20285 + Euro symbol)
1147	IBM EBCDIC - France (20297 + Euro symbol)
1148	IBM EBCDIC - International (500 + Euro symbol)
1149	IBM EBCDIC - Icelandic (20871 + Euro symbol)
1200	Unicode UCS-2 Little-Endian (BMP of ISO 10646)
1201	Unicode UCS-2 Big-Endian
1250	ANSI - Central European
1251	ANSI - Cyrillic
1252	ANSI - Latin I
1253	ANSI - Greek
1254	ANSI - Turkish
1255	ANSI - Hebrew
1256	ANSI - Arabic
1257	ANSI - Baltic
1258	ANSI/OEM - Vietnamese
1361	Korean (Johab)
10000	MAC - Roman
10001	MAC - Japanese
10002	MAC - Traditional Chinese (Big5)
10003	MAC - Korean
10004	MAC - Arabic
10005	MAC - Hebrew
10006	MAC - Greek I
10007	MAC - Cyrillic
10008	MAC - Simplified Chinese (GB 2312)
10010	MAC - Romania
10017	MAC - Ukraine
10021	MAC - Thai
10029	MAC - Latin II
10079	MAC - Icelandic
10081	MAC - Turkish
10082	MAC - Croatia

12000	Unicode UCS-4 Little-Endian
12001	Unicode UCS-4 Big-Endian
20000	CNS - Taiwan
20001	TCA - Taiwan
20002	Eten - Taiwan
20003	IBM5550 - Taiwan
20004	TeleText - Taiwan
20005	Wang - Taiwan
20105	IA5 IRV International Alphabet No. 5 (7-bit)
20106	IA5 German (7-bit)
20107	IA5 Swedish (7-bit)
20108	IA5 Norwegian (7-bit)
20127	US-ASCII (7-bit)
20261	T.61
20269	ISO 6937 Non-Spacing Accent
20273	IBM EBCDIC - Germany
20277	IBM EBCDIC - Denmark/Norway
20278	IBM EBCDIC - Finland/Sweden
20280	IBM EBCDIC - Italy
20284	IBM EBCDIC - Latin America/Spain
20285	IBM EBCDIC - United Kingdom
20290	IBM EBCDIC - Japanese Katakana Extended
20297	IBM EBCDIC - France
20420	IBM EBCDIC - Arabic
20423	IBM EBCDIC - Greek
20424	IBM EBCDIC - Hebrew
20833	IBM EBCDIC - Korean Extended
20838	IBM EBCDIC - Thai
20866	Russian - KOI8-R
20871	IBM EBCDIC - Icelandic
20880	IBM EBCDIC - Cyrillic (Russian)
20905	IBM EBCDIC - Turkish
20924	IBM EBCDIC - Latin-1/Open System (1047 + Euro symbol)
20932	JIS X 0208-1990 & 0121-1990
20936	Simplified Chinese (GB2312)
21025	IBM EBCDIC - Cyrillic (Serbian, Bulgarian)
21027	Extended Alpha Lowercase

21866	Ukrainian (KOI8-U)
28591	ISO 8859-1 Latin I
28592	ISO 8859-2 Central Europe
28593	ISO 8859-3 Latin 3
28594	ISO 8859-4 Baltic
28595	ISO 8859-5 Cyrillic
28596	ISO 8859-6 Arabic
28597	ISO 8859-7 Greek
28598	ISO 8859-8 Hebrew
28599	ISO 8859-9 Latin 5
28605	ISO 8859-15 Latin 9
29001	Europa 3
38598	ISO 8859-8 Hebrew
50220	ISO 2022 Japanese with no halfwidth Katakana
50221	ISO 2022 Japanese with halfwidth Katakana
50222	ISO 2022 Japanese JIS X 0201-1989
50225	ISO 2022 Korean
50227	ISO 2022 Simplified Chinese
50229	ISO 2022 Traditional Chinese
50930	Japanese (Katakana) Extended
50931	US/Canada and Japanese
50933	Korean Extended and Korean
50935	Simplified Chinese Extended and Simplified Chinese
50936	Simplified Chinese
50937	US/Canada and Traditional Chinese
50939	Japanese (Latin) Extended and Japanese
51932	EUC - Japanese
51936	EUC - Simplified Chinese
51949	EUC - Korean
51950	EUC - Traditional Chinese
52936	HZ-GB2312 Simplified Chinese
54936	Windows XP: GB18030 Simplified Chinese (4 Byte)
57002	ISCII Devanagari
57003	ISCII Bengali
57004	ISCII Tamil
57005	ISCII Telugu
57006	ISCII Assamese

57007	ISCII Oriya
57008	ISCII Kannada
57009	ISCII Malayalam
57010	ISCII Gujarati
57011	ISCII Punjabi
65000	Unicode UTF-7
65001	Unicode UTF-8

Magic eDeveloper V10

Magic eDeveloper V10
新機能チュートリアル
Copyright © 2006, Magic Software Japan K.K.,
All rights reserved.

第 1 版　　2006 年 12 月 20 日
発行　　〒151-0053　東京都渋谷区代々木三丁目二十五番地三号
あいおい損保新宿ビル 14 階
マジック ソフトウェア・ジャパン（株）