IBM Rational System Architect VBA 拡張性ガイド リリース 11.3.1

本書をご使用になる前に、「付録」の『特記事項』(20-1 ページ) を参照してください。

本書は、IBM® Rational® System Architect® バージョン 11.3.1、および新しい版で明記されていない限り、以降のすべてのリリースおよびモディフィケーションに適用されます。

目次

1

はじめに

はじめに
この章では、Rational System Architect および Visual Basic for Applications (VBA) での作業に関する基本事項について説明します。マクロ、VBA エディター、およびオブジェクト・ブラウザーの使用について説明します。

Rational System Architect の自動化

Rational System Architect は VBA に対応しているため、ユーザーは Rational System Architect 環境をプログラムで制御でき、OLE オートメーションを使用して他のアプリケーションを制御できます。

Microsoft VBA とその開発環境は、Rational System Architect にインストールされています。プログラミング環境、デバッグ環境、言語、およびヘルプ・システムは、Microsoft Office 製品などの他の VBA 対応アプリケーションと同じです。

オートメーションを使用して、Rational System Architect と他のアプリケーションを統合するには、以下の 2 つの方法があります。Rational System Architect を使用して、以下を実行します。

- Rational System Architect をオートメーション・コントローラーとして使用し、Rational System Architect スクリプト内から OLE オートメーション・オブジェクトを呼び出します。
- Rational System Architect をオートメーション・サーバーとして使用し、別の OLE 準拠のアプリケーション内から OLE オートメーション・オブジェクトを呼び出します。

Rational System Architect のマクロを使用して、以下を行うことができます。

- Rational System Architect の機能の拡張 – ダイアグラムの表示およびルール・チェックを自動化します。
- 他のアプリケーションに含まれている情報を使用して、ダイアグラム、シンボル、および定義を作成します。

Rational System Architect で発生したイベントを取り込み、ファイルまたはデータベースに保管します。

VBA を使用した Rational System Architect のプログラミング

Rational System Architect では、初期設定ファイル SA2001.INI (Windows フォルダーまたは WinNt フォルダーにあります) に VBA プロジェクト環境の現在の状況が保管されます。**Macros** セクションには、以下のような内容が含まれています。

```
[Macros]
PermanentAutoLoad1=SAAuto.mac
TemporaryAutoLoad1= c:\program files\ibm \system
  architect suite\system
  architect\sample.mac|vxRead|vxShared|vxTransacted|b
  ProjectActive
RunTemporaryAutoMacros=T
OpenReadOnly=T
```

上記の例では、Rational System Architect の開始時に永久的な SAAUTO.MAC プロジェクト・ファイルがロードされることが示されています (非表示かつ読み取り専用)。一時 SAMPLE.MAC プロジェクト・ファイルは現在アクティブで、読み取り専用モードでロードされています。いずれかの一時プロジェクト・ファイルに自動実行モジュールが含まれている場合、マクロは自動的に実行されます。デフォルトでは、新規プロジェクトは読み取り専用で開かれます。

マクロの実行

「**マクロ**」ダイアログから直接マクロを実行する手順は、次のとおりです。

1. **ツール**メニューから**マクロ**サブメニューを選択します。
2. **マクロ実行...**オプションを選択します。

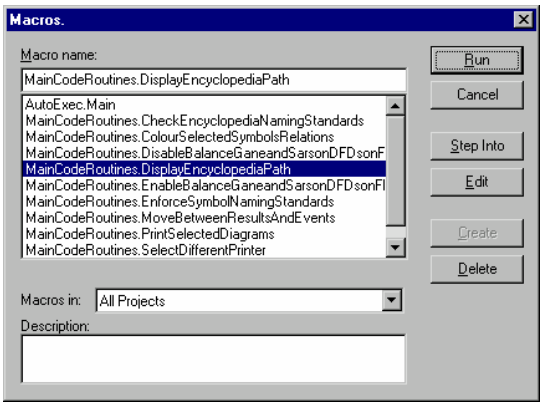


図 2-1.「マクロ」ダイアログ

マクロは、ツールバーから実行することもできます。

ツールバーにマクロを追加するには、以下のようにします。

- メニュー・バーまたは任意のツールバーを右クリックして、「**カスタマイズ...**」オプションを選択します。
- **コマンド**タブを選択し、**マクロ**オプションを選択します。
- 使用可能なマクロをスクロールし、必要なマクロをツールバーに**ドラッグ・アンド・ドロップ**します。

マクロ・プロジェクト

他の一時プロジェクト・ファイルをサンプル・プロジェクトに追加できます。

マクロ・プロジェクトダイアログにアクセスする手順は、次のとおりです。

1. **ツール**メニューを開きます。
2. **マクロ**サブメニューを選択します。
3. **マクロ・プロジェクト...**オプションを選択します。

新しいプロジェクト・ファイルを使用可能なプロジェクトのリストに追加したり、既存のプロジェクトを除去したりできます。

自動マクロを有効にするチェック・ボックスを選択解除すると、自動実行マクロが無効になります。

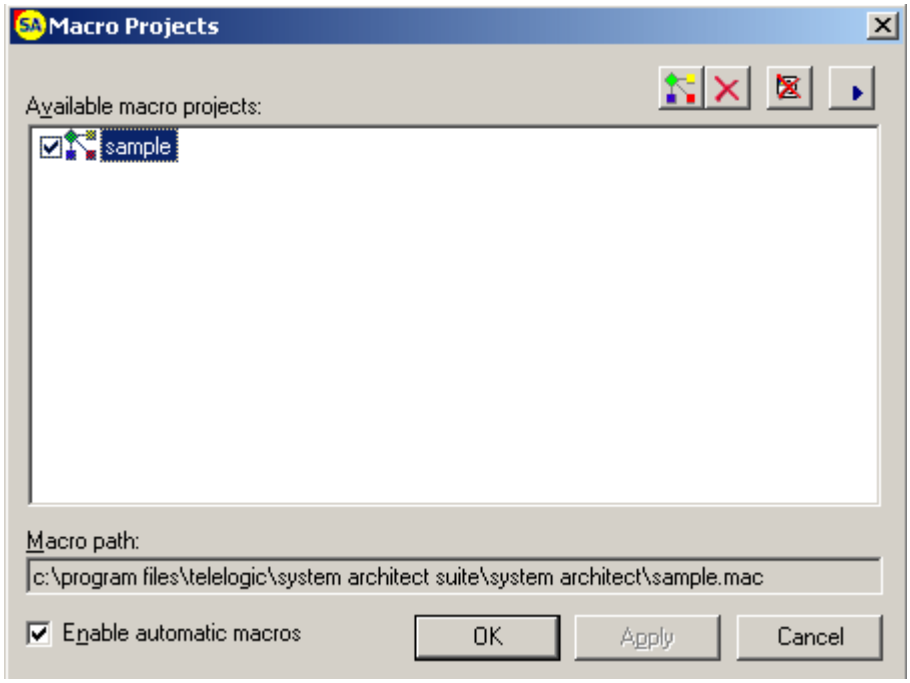


図 2-2. 「マクロ・プロジェクト」ダイアログ

新規プロジェクトの追加

追加コマンド・ボタンをクリックすると、**マクロ・プロジェクトを開く**ダイアログが表示されます。

Rational System Architect プロジェクト・ファイルの拡張子は、.MACです。

既存のプロジェクトを選択して開くことも、リストにはないファイル名を入力して新しいプロジェクトを作成することもできます。

プロジェクトを変更する場合は、プロジェクト・ファイルを開く前に「**読み取り専用で開く (Open as read-only)**」チェック・ボックスが選択解除されていることを確認してください。

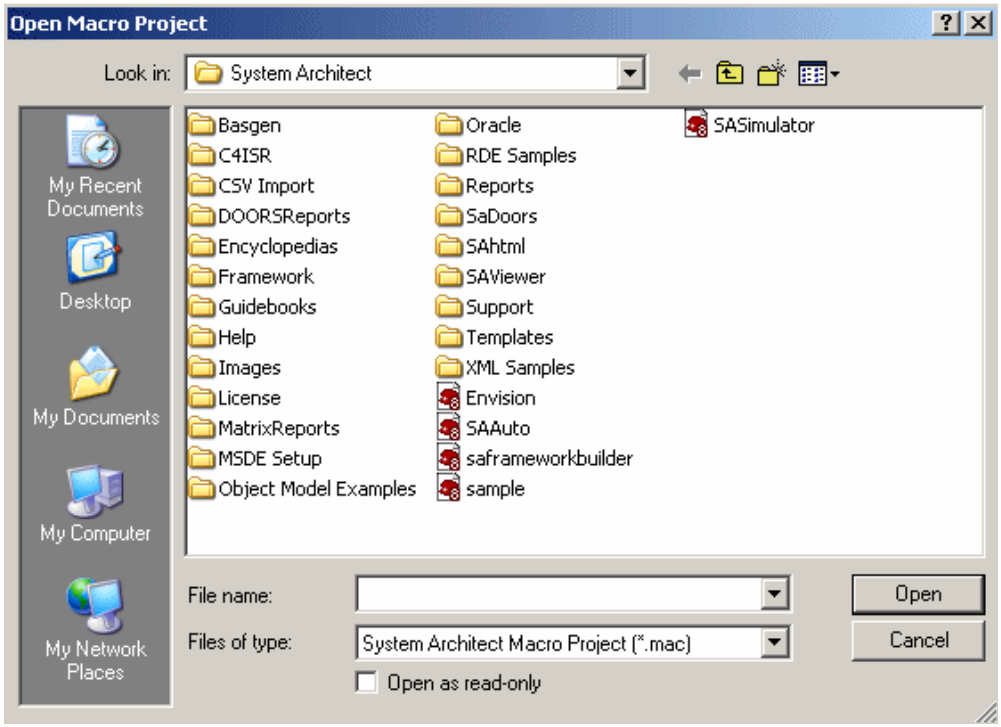


図 2-3 「マクロ・プロジェクトを開く」ダイアログ

VBA エディターへのアクセス

マクロを作成したり、既存のマクロを変更したりするには、VBA エディターを開く必要があります。

VBA エディターを開く方法はいくつかあります。

- ツールメニューから**マクロ**サブメニューを選択し、「VBA エディター」オプションを選択します。
- Alt+F11 を押します。
- マクロダイアログから、**編集**コマンド・ボタンをクリックします。
- マクロ・**プロジェクト**ダイアログから、**適用**コマンド・ボタン (アクティブな場合) をクリックし、**マクロ実行...**コマンド・ボタンをクリックします。これで、**マクロ**ダイアログが開き、**編集**コマンド・ボタンが表示されます。

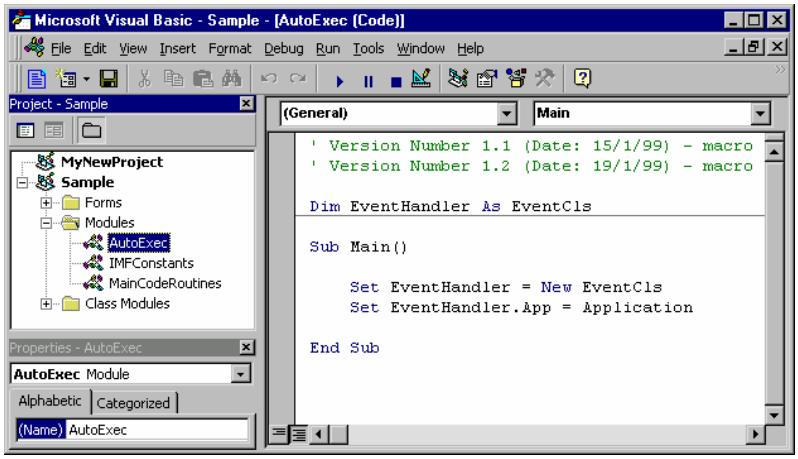


図 2-4 VBA エディター

プロジェクト・エクスプローラー

VBA エディターは、複数のウィンドウで構成されています。通常、VBA エディターに初めてアクセスすると、「プロジェクト」ウィンドウが表示可能になります。

プロジェクトウィンドウを表示するには、次の手順を実行します。

1. **表示**メニューから**プロジェクト・エクスプローラー (Project Explorer)**オプションを選択します。
2. サンプル・プロジェクトには、3 つの**フォルダー・グループ** (**フォーム**、**モジュール**、および**クラス・ モジュール**が含まれています。
3. 各フォルダーを開いてその内容を表示するには、プラス記号をクリックするか、フォルダーまたはフォルダー名をダブルクリックします。

各フォルダーの内容はアルファベット順に表示されますが、フォルダーをオフに切り替えてすべての項目のアルファベット順のリストを表示できます。

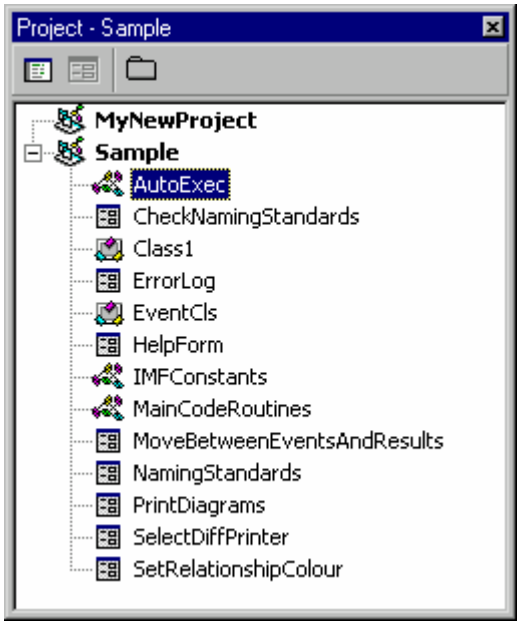


図 2-5 「プロジェクト」ウィンドウ

「プロパティー」ウィンドウ

いずれかの**モジュール**をクリックすると、そのモジュールの名前の「**プロパティー**」ウィンドウが表示されます。この名前は、「**プロパティー**」ウィンドウで変更できます。

モジュール名をダブルクリックすると、そのモジュールのコードが表示されます。

フォーム名をダブルクリックすると、**フォーム・オブジェクト**が表示されます。また、「プロパティー」ウィンドウにフォームのプロパティーがアルファベット順にリスト表示されます。

フォーム名を右クリックすると、**フォームのコード**を**表示**するオプションが表示されます。

また、「**プロパティー**」ウィンドウでは、カテゴリーごとに表示することもできます。

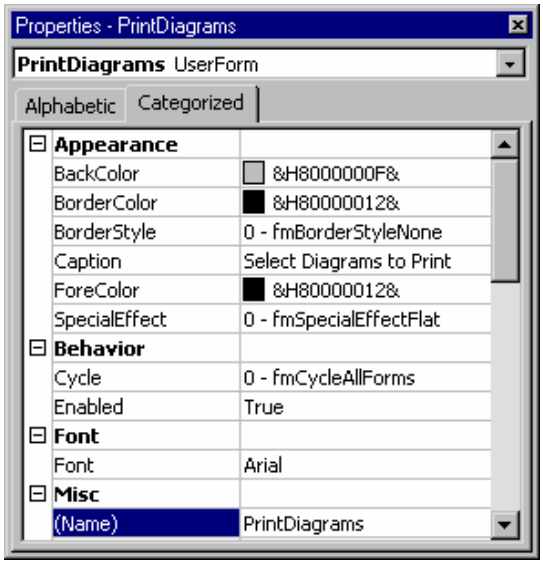


図 2-6 「プロパティー」ウィンドウ

モジュールおよびフォームの挿入

プロジェクトに**モジュール**および**フォーム**を追加するには、**挿入**メニューを使用します。

選択した**モジュール**または**フォーム**を除去し、**エクスポート**する (オプション) には、「**ファイル**」メニューの「**除去**」オプションを使用します。

プロジェクトウィンドウを右クリックして、**挿入**オプションおよび**除去**オプションにアクセスすることもできます。

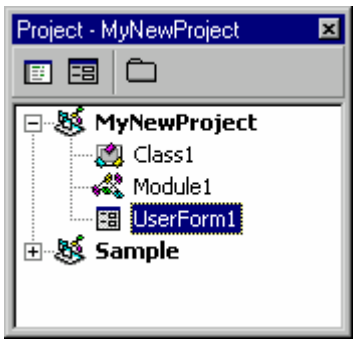


図 2-7 「プロジェクト」ウィンドウ

オブジェクト・ブラウザー

オブジェクト・ブラウザーは VBA エディターの非常に便利な機能で、使用可能なオブジェクト・ライブラリー、タイプ・ライブラリー、およびダイナミック・リンク・ライブラリーを問い合わせることができます。

ライブラリー・ファイルの参照

オブジェクト・ブラウザーで追加のライブラリーを使用可能にするには、次の手順を実行します。

- **ツール**メニューから**参照...**オプションを選択します。

選択した参照のリストには、SA2001 ライブラリーが既に含まれていることが予想されます。

「使用可能な参照 (Available References)」のリストをスクロールダウンして、他の参照を選択できます。

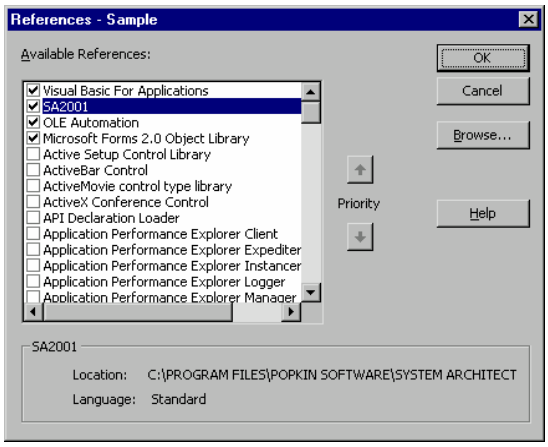


図 2-8 参照の選択

オブジェクト・ブラウザーを表示するには、F2 ファンクション・キーを押すか、次の手順を実行します。

- **表示**メニューから**オブジェクト・ブラウザー (Object Browser)**オプションを選択します。

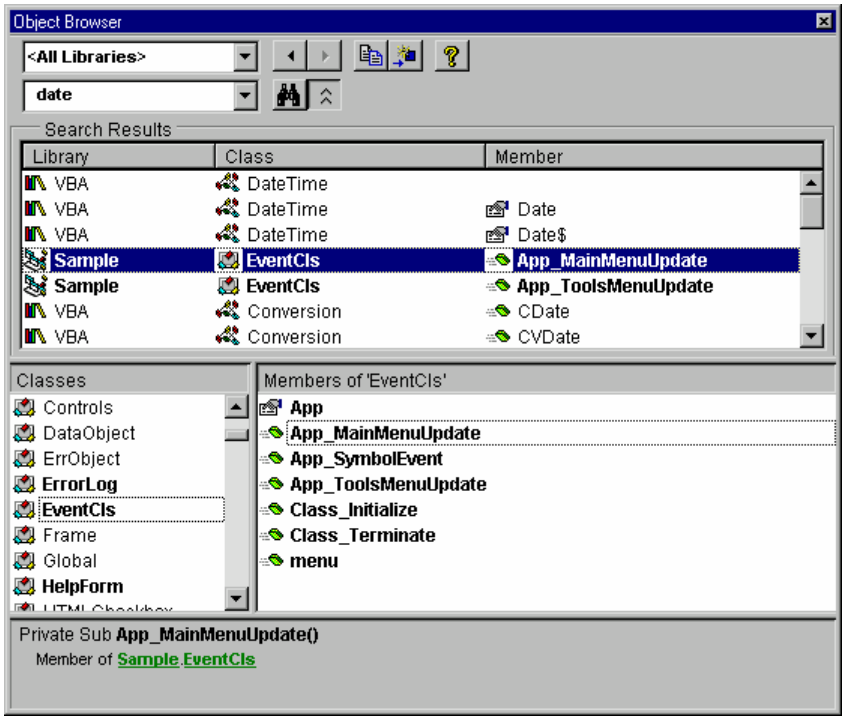


図 2-9 オブジェクト・ブラウザー

オブジェクト・ブラウザーは、特定のライブラリーでフィルタリングできます。また、前述のとおり、クラス名またはメンバー名に特定のテキストが含まれている項目の検索に使用できます。

「プロジェクト・エクスプローラー (Project Explorer)」ウィンドウでプロジェクトが選択されている場合、オブジェクト・ブラウザーでは、プロジェクトの一部であるすべての項目が強調表示されます。

クラス、定数、列挙型、イベント、グローバル変数、メソッド、モジュール、プロパティー、およびユーザー定義型がすべて表示されます。それぞれ、先頭にアイコンが表示されます。

2

オートメーションと *Rational System Architect*

はじめに

オートメーションとは、別のアプリケーションで作成されたオブジェクトを宣言し、使用するためのアプリケーションの機能です。Visual Basic for Applications (VBA)を使用すると、1 つのプログラムでこのような 1 つ以上のアプリケーションの複数のオブジェクトにアクセスするコードが作成できます。

オートメーションを使用すると、複数の製品の機能に基づいて、完全に統合されたソリューションを構築することが可能になります。Rational System Architect には、VBA が組み込まれているため、他のアプリケーションで その機能を使用できます。この資料の以下のセクションでは、この機能について説明します。

この章では、オートメーションについて説明し、オートメーションによってカスタマイズ可能なソリューションが提供される仕組みを示します。その後で、ソリューションに必要な機能を提供するよう Rational System Architect をセットアップする方法について詳しく説明します。

オートメーション

オートメーションを使用すると、プログラマーは、他のアプリケーションから現行アプリケーションにオブジェクトを取り込んで、統合されたソリューションを提供できます。例えば、Rational System Architect を使用して、何人かのユーザーによって構築された企業データ・モデルを作成した場合に、それとは別のユーザーが特別に設計された Word レポートまたは Excel スプレッドシートでのデータの表示を希望しているとします。オートメーションを使用して、統合された Word または Excel レポートを Rational System Architect 内のカスタム・メニュー項目から直接実行できます。オートメーションを使用すると、ユーザーは統合される特定のタスク用に作成されたツールを使用できます。

また、かつては、オートメーションは OLE オートメーションとも呼ばれ、ActiveX オートメーションと呼ばれることもありました。

オートメーション・コントローラーおよびサーバー

オートメーションは、別のアプリケーションのオブジェクトを制御する方法の 1 つです。技術的には、1 つのアプリケーションでオートメーションを制御する VBA コードを保持し、別のアプリケーションで使用対象のオブジェクトを提供します。このプロセスのドライバーはオートメーション・コントローラーと呼ばれ、プロバイダーはオートメーション・サーバーと呼ばれます。

型ライブラリーの参照

VBA は、任意の VBA プログラム内で使用可能なオブジェクトを確認するためのメカニズムを備えています。VBA は、Rational System Architect でコードが作成されている場合には、そのオートメーション・オブジェクトのライブラリーが使用可能であると推測しますが、他の種類のライブラリーも使用可能であるとは見なしません。ユーザーは、必要なライブラリーへの参照を設定する必要があります。

型**ライブラリー**は、任意のアプリケーションでオートメーションに使用できるすべてのオブジェクトに関する情報が収容されているデータベースです。この情報には、アプリケーションで使用可能なオブジェクト、属性、イベント、およびメソッドに関する詳細が含まれています。型ライブラリーは通常、アプリケーションと同時にインストールされる別個のファイルですが、メイン実行可能ファイルの一部として提供されることもあります。

この型ライブラリー情報を参照するには、ユーザーは、VBA エディターで作業している必要があります。VBA エディターには、関連アプリケーション・メニューからアクセスするか、または **Alt キーを押しながら F11** キーを押してアクセスします。VBA エディターにアクセスしたら、**ツール」>「参照**を選択します。

この例では、VBA および Rational System Architect の標準オブジェクトのみが参照オブジェクトとなります。他のアプリケーションを組み込むには、スクロールダウンして希望するアプリケーション (例えば、Microsoft Excel 9.0 オブジェクト・ライブラリー) を選択しま

す。これにより、現行のアプリケーションに Excel タイプのライブラリーのすべてのコンポーネントが組み込まれます。

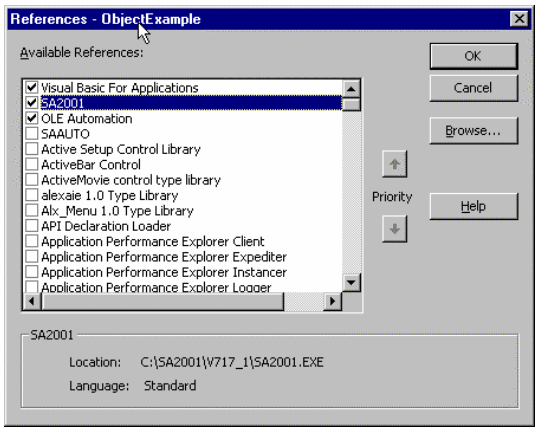


図 2-1.型ライブラリーの選択

オートメーション・オブジェクトの表示

別のアプリケーションに参照を設定すると、現在参照されているすべての型ライブラリーのオブジェクト、属性、およびメソッドのリストを表示できるようになります。VBA エディターには、すべてのプロパティーを表示する独自のオブジェクト・ブラウザーがあります。

VBA オブジェクト・ブラウザーにアクセスするには、F2 を押すか、または VBA エディターで**表示」>「オブジェクト・ブラウザー (Object Browser)**メニュー項目を選択します。

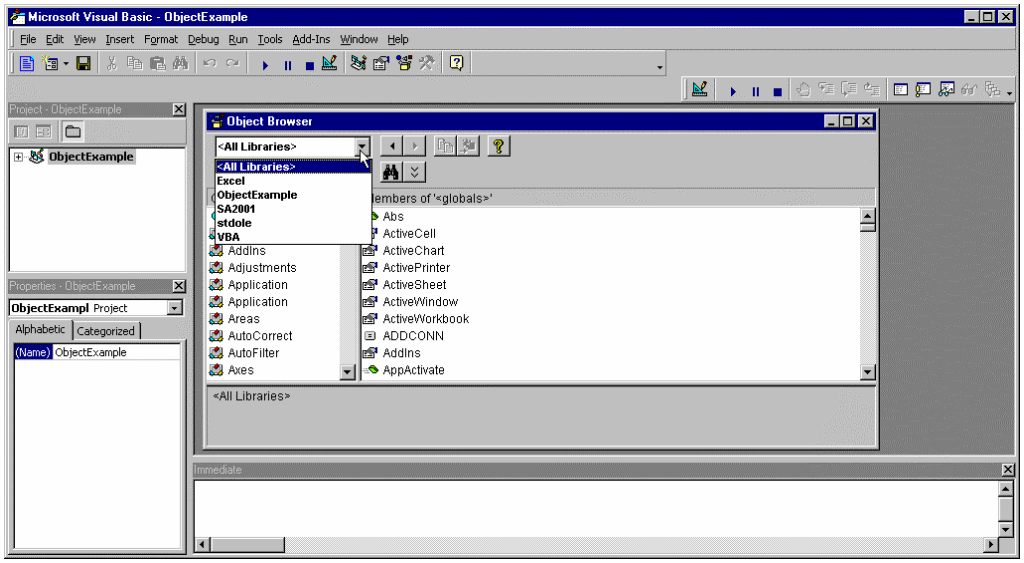


図 2-2. VBA オブジェクト・ブラウザー

オブジェクト・ブラウザーには、型ライブラリー参照で選択されたすべてのライブラリーがリストされます (ただし、個別のオブジェクトのセットを選択することもできます)。以下の例では、表示する型ライブラリーとして SA2001 を選択し、特に、オートメーション・オブジェクトの Definition クラスを選択します。

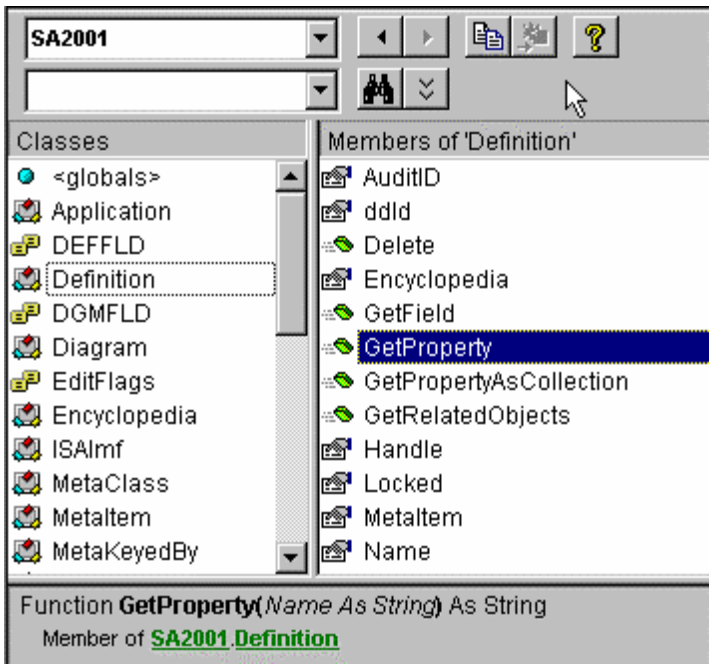


図 2-3. SA2001 型ライブラリーの参照

Definition クラスのメンバーがリストされ、**GetProperty**が選択されています。型および戻りの型を含むパラメーターの詳細が下の表示ウィンドウに表示されます。オートメーション・オブジェクトへの参照を宣言すると、VBA プログラマーが、すべての内部アプリケーション情報を使用して統合ソリューションを構築できるようになります。

アプリケーションのインスタンスの作成

必要なアプリケーションは VBA で参照されますが、アプリケーション・オブジェクトを制御するには、コードを書き込む必要があります。次のセクションでは、アプリケーションの新しいインスタンスをセットアップしてその自動化されたオブジェクトにアクセスする方法、およびオブジェクトを宣言する方法について説明します。

インスタンスとは、必要なアプリケーションのセッションのことです。オートメーション・オブジェクトを使用するには、アプリケーションがメモリーに常駐している必要があります (表示されている必要はありません)。オブジェクトにアクセスするには、VBA で Dim ステートメントおよび Set ステートメントを使用して、あたかも組み込みデータ型に

対して宣言するのと同様に、オートメーション・オブジェクトのインスタンスを宣言します。ただし、オートメーションの場合に唯一違う点として、オブジェクトが使用される時点でそのオブジェクトの新しいインスタンスが Set ステートメントで作成される必要があります。

次のコードでは、変数 *ExObj* を Excel アプリケーションとして宣言しています。宣言の server.class 完全名に注意してください。この名前によって、参照されているアプリケーション・クラスが Excel のクラスであることが確認できます。他のアプリケーションにもアプリケーションと呼ばれるクラスがあることがあります。

```
Dim ExObj as Excel.Application
```

次の行では、Excel アプリケーションの新しいインスタンスを作成し、コード内にインスタンス化が実行されるポイントを設定します。

```
Set ExObj = New Excel.Application
```

これで、変数 *ExObj* は Excel オートメーション・オブジェクトにアクセスできます。例えば、Excel から標準の「**開く**」ダイアログ・ボックスを表示するには、次のコードによってテキスト・ファイル用にフィルタリングされたオートメーション・オブジェクト **GetOpenFilename()** を使用します。

```
fileToOpen = ExObj.GetOpenFilename("Text Files
  (*.txt), *.txt")
```

このように Excel のすべてのオートメーション・オブジェクトが、Rational System Architect のそれと同様に使用可能になります。

アプリケーション・インスタンスの解放

オートメーション・オブジェクトのインスタンスは、セットで作成され、オートメーション関数呼び出しで操作されて、終了します。オートメーション・クラスを終了すると、リソースがメモリーから解放されます。

オートメーション・オブジェクトが不要になった場合は、次のコードを使用してオートメーション・サーバーを実質的に終了させます。

```
Set Exobj = Nothing
```

まとめ

Dim Object = Server.class	サーバーの宣言
Set Object = New Server.class	サーバーのインスタンス化
Automation Code	サーバー・オートメーション・オブジェクトにアクセスする VBA コード
Dim Object = Nothing	メモリーからのオートメーション・サーバーの解放

カスタマイズ可能なソリューション

前のセクションでは、オートメーションの概念を示し、あるアプリケーションのオブジェクトを他のアプリケーションと統合する方法について説明しました。VBA およびオートメーションがサポートされているすべての製品は、カスタマイズ可能なソリューションと見なすことができます。つまり、VBA に対応したものであれば、購入した製品はすべて、必要に応じて変更できます。ただし、ここで言う変更とは、別のアプリケーションに統合することだけを意味するものではありません。アプリケーションのカスタマイズでは、多くの場合、製品が実際に機能する仕組みも変更されます。

メニュー項目の変更、Office 製品との統合、反復タスクのオートメーションなど、カスタマイズに対する考え方は、同じ製品であってもユーザーごとに異なることがありますが、こうした問題にはすべて、VBA およびオートメーションを使用して対処できます。以下の表に、製品のカスタマイズが必要な可能性がある理由を 5 つ示します。これらの問題は、VBA を使用して認識できます。

カテゴリー	意味	Rational System Architect の例
アプリケーション動作の変更	企業のビジネス・ルールおよびプロセスに一致するようアプリケーションの処理方法を変更する。	ダイアグラムでのシンボルの作成時に、企業の標準命名規則をチェックし、規則が守られていない場合には、ユーザーに知らせるためにフラグを立てる。
反復タスクの自動化	一般的な手動タスクのセットと反復して実行される可能性があるアクションを組み合わせる。	一連のユーザー定義ダイアグラムを印刷する。
アプリケーション機能の拡張	購入時にはなかった機能をアプリケーションに追加する。	プロセス・チャートから派生したプロセス・マップ・ダイアグラム、それに割り当てられたロール、およびロールの組織単位を自動作成する。
他のアプリケーションとの統合	別のアプリケーションを制御して、通常は使用できない機能を活用する。	プロセスとエンティティー CRUD を対応させた情報を示す Excel スプレッドシートを作成する。
企業データへのアクセス	リモート・データベースおよび通常はデータベースにアクセスできないアプリケ	通常は直接インポートできないさまざまなソースから情報を自動インポートする。

	ーションとデータを交換する。	

Rational System Architect と VBA を組み合わせると、IntelliSense や Microsoft フォームなどの、Microsoft によって提供されたすべてのプログラミング開発を統合して、カスタマイズ可能なソリューションを作成する、標準の統合開発環境 (IDE) がユーザーに提供されます。

Rational System Architect での自動化されたソリューションの計画

オートメーションには、反復タスクの削減や完全に統合されたアプリケーションの作成など、さまざまな用途があります。次のセクションでは、Rational System Architect によって可能となる変更点について説明します。

動作の制御

Rational System Architect は、操作時に発生する特定のイベントに応答します。このようなイベントは、特定のタスクを自動化するために使用されるコードをトリガーできます。

Rational System Architect でサポートされているイベントには、製品の開始とシャットダウン、エンサイクロペディアの開始と終了、ダイアグラムの開始、保存と終了、監査 ID の変更、およびいくつかのシンボル・イベントなどがあります。シンボル・イベントとしては、ダイアグラムへの配置、命名、(ライン・シンボルとの) 接続と切断、削除があります。

この機能によって、Rational System Architect のリアルタイム操作を VBA を使用して変更できます。

外観の制御

Rational System Architect 10.1 以降では、ユーザーは既存のメニュー構造に対してコマンドを追加または削除して、Rational System Architect のメニューをカスタマイズできます。例えば、ユーザーは、マクロを実行するコマンドを追加できます。Rational System Architect 10.1 より前 (つまり、10.0 以前) では、メニュー項目にコマンドを追加するには、Rational System Architect VBA を使用する必要がありました。例えば、メソドロジーに固有の項目を含むメニューは、コードの実行時に更新することができました。このようなメニュー項目は、選択したメソドロジーに依存する特定のダイアグラムにのみ表示されました。

いずれの場合も、メニュー項目およびメニュー・ポップアップを、これらの項目に割り当てられているメニューおよびビットマップに追加することができます。この手法は、一般には、メニュー項目にマクロを追加したり、マクロを表示するビットマップを提供したりするために使用されます。その後、これをツールバーおよびメニューに配置することもできます。

Rational System Architect 10.0 以前のメニュー項目で使用するために作成されたマクロを Rational System Architect 10.1 用に調整する方法については、IBM サポート・サイトの変換マニュアルを参照してください。

タスクの自動化

リポジトリー内の情報に関してレポートしたり、整合性検査を実行したりするために VBA マクロを作成できます。データ・ディクショナリー内のすべてのオブジェクトおよびプロパティーは、コードで設定された規則に基づいて作成、読み取り、更新、または削除することができます。このため、あるデータ・ディクショナリー・オブジェクトの値を別のデ

ータ・ディクショナリー・オブジェクトの値で更新するなどの、単純な反復的タスクを自動化することができます。

制御の実行

Rational System Architect VBA モデルのイベント・ドリブン・メンバーを使用すると、事前定義された一連の標準をリアルタイムのモデルに適用できます。このような標準には、命名標準や、必須フィールドの記入などがあります。

外部アプリケーション間のインターフェース

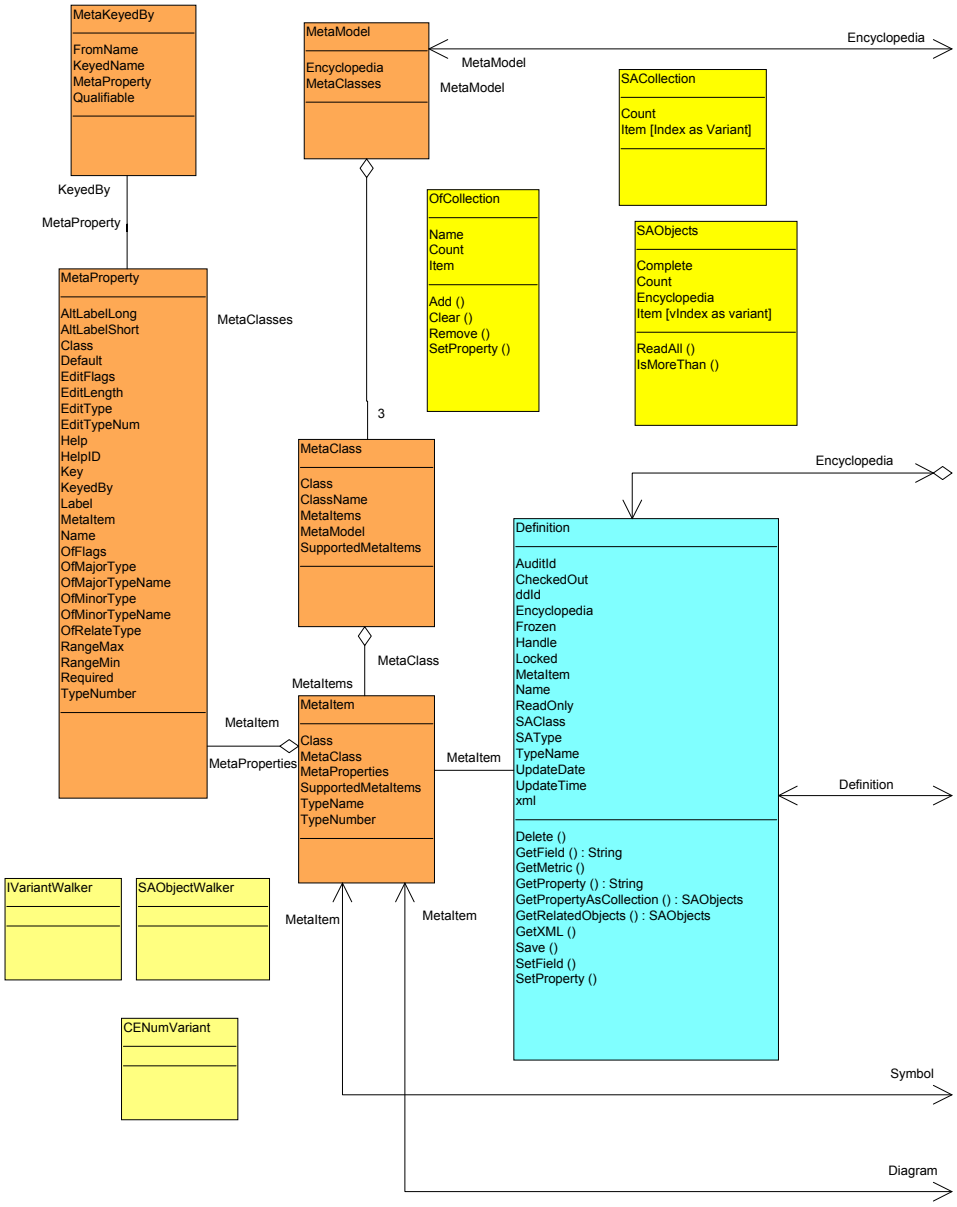
VBA を使用すると、他のアプリケーションの値に基づいてモデル・オブジェクトをインポート、エクスポート、読み取り、作成、変更、更新、および削除することができます。機能の例には、ダイアグラム、シンボル、定義の作成、シンボル間の関係の構築、データ・ディクショナリー・プロパティーの変更などがあります。

3

Rational System Architect のオブジェクト・モデル

VBA で使用できる Rational System Architect のオブジェクトを表示するには、VBA エディターのオブジェクト・ブラウザーを使用します。各オブジェクト・タイプは、サポートされているプロパティーおよびメソッドのリストが含まれているクラスとして定義されています。

モデル全体を表示する場合は、クラス図を使用すると便利です。次の図に、Rational System Architect で UML クラス図表記を使用して描かれた Rational System Architect オブジェクト・モデルを抽出したものを示します。


MetaKeyedBy
FromName
KeyedName
MetaProperty
Qualifiable
KeyedBy
MetaProperty
MetaModel
Encyclopedia
MetaClasses
MetaModel
MetaModel
SACollection
Count
Item [Index as Variant]
OfCollection
Name
Count
Item
Add ()
Clear ()
Remove ()
SetProperty ()
SAObjects
Complete
Count
Encyclopedia
Item [vIndex as variant]
ReadAll ()
IsMoreThan ()
MetaProperty
AltLabelLong
AltLabelShort
Class
Default
EditFlags
EditLength
EditType
EditTypeNum
Help
HelpID
Key
KeyedBy
Label
MetaItem
Name
OfFlags
OfMajorType
OfMajorTypeName
OfMinorType
OfMinorTypeName
OfRelateType
RangeMax
RangeMin
Required
TypeNumber
MetaClasses
3
MetaClass
Class
ClassName
MetaItems
MetaModel
SupportedMetaItems
Encyclopedia
Definition
AuditId
CheckedOut
ddId
Encyclopedia
Frozen
Handle
Locked
MetaItem
Name
ReadOnly
SAClass
SAType
TypeName
UpdateDate
UpdateTime
xml
Delete ()
GetField () : String
GetMetric ()
GetProperty () : String
GetPropertyAsCollection () : SAObjects
GetRelatedObjects () : SAObjects
GetXML ()
Save ()
SetField ()
SetProperty ()
MetaClass
MetaItems
MetaItem
MetaProperties
MetaItem
MetaItem
Class
MetaClass
MetaProperties
SupportedMetaItems
TypeName
TypeNumber
MetaItem
MetaItem
Definition
IVariantWalker
SAObjectWalker
CENumVariant
Symbol
Diagram


Encyclopedia
Application
ConnectString
FullName
MetaModel
Name
OpenObjectAsReadOnly
Path
xml
xmlex
bOpenLockedReadOnly ()
CloseUnlock ()
CreateDefinition () : Definition
CreateDiagram () : Diagram
GetAllDefinitions () : SAObjects
GetAllDiagrams () : SAObjects
GetCurrentDiagram ()
GetDefinitionById () : Definition
GetDiagramById () : Diagram
GetFilteredDefinitions () : SAObjects
GetFilteredDiagrams () : SAObjects
GetRelationMetric ()
GetXML ()
OpenEncyclopedia ()
OpenLock ()
SetXML ()
SetXMLEx ()
Application
Encyclopedia
Mouse Pointer
Visible
AssignBMPtoMacroItem ()
BrowserRefresh ()
CreatePopUpMenu () : long
InsertMacroItemInMenu ()
InsertMacroItemInMenuEx ()
InsertPopupMenuItemInMenu ()
InsertPopupMenuItemInMenuEx ()
Interface ()
OpenEncyclopedia ()
RemoveItemFromMenu ()
RemovePopUpMenu ()
ResetPanelBackGround ()
SetSeparatorBefore ()
WriteStatusLine ()
WriteStatusLineEx ()
Encyclopedia
MetaModel
Application
Encyclopedia
Encyclopedia
Encyclopedia
Encyclopedia
Symbol
ArrowAtEnd
ArrowAtStart
AuditId
ddId
Definition
Diagram
Encyclopedia
FillColor
FontColor
FromCardinality
Handle
LineStyle
MetaItem
Name
PenColor
PenStyle
SAClass
SAType
Selected
ToCardinality
TunnelAtEnd
TunnelAtStart
TypeName
UpdateDate
UpdateTime
XPos
XSize
YPos
YSize
ConnectFrom ()
ConnectTo ()
Delete ()
GetChildDiagrams ()
GetField () : String
GetMetric ()
GetProperty () : String
GetPropertyAsCollection () : SAObjects
GetRelatedObjects () : SAObjects
Save ()
SetField ()
SetProperty ()
Diagram
AuditId
CheckedOut
ddId
Encyclopedia
Frozen
Handle
Hidden
Lock
MetaItem
Name
Picture
ReadOnly
SAClass
SAType
TypeName
UpdateDate
UpdateTime
xml
CreateSymbol () : Symbol
Delete ()
GetAllSymbols () : SAObjects
GetField () : String
GetFilteredSymbols () : SAObjects
GetMetric ()
GetParentSymbol ()
GetProperty () : String
GetPropertyAsCollection () : SAObjects
GetRelatedObjects () : SAObjects
GetSymbolById () : Symbol
GetXML ()
Hide ()
Save ()
SetField ()
SetProperty ()
Show ()
Definition
Diagram
MetaItem
MetaItem

オブジェクト・モデル・クラス

以下に、Rational System Architect オブジェクト・モデル・クラスと、それぞれの使用例をいくつか示します。

クラス	使用例
Application	Rational System Architect のイベント メニュー操作
Encyclopedia	ダイアグラムおよび定義オブジェクトの作成 ダイアグラムおよび定義オブジェクトの検索
Diagram	シンボル・オブジェクトの作成および検索 ダイアグラム・プロパティーの操作
Symbol	シンボル・プロパティーの操作 シンボル接続情報 子ダイアグラム・オブジェクトの検索
Definition	定義プロパティーの操作 関連した定義の操作
SACollection	Rational System Architect のプロパティーのコレクション
SAObjects	Rational System Architect のオブジェクト (ダイアグラム、シンボル、および定義) のコレクション
OfCollection	OneOf または ListOf ダイアグラムまたは定義のコレクション
MetaModel	Rational System Architect Mのメタ・モデル・オブジェクト (サポートされ

MetaClass MetaItem	ているダイアグラム、シンボル、および定義のタイプ)
MetaKeyedBy	定義によってキー付けされた (定義およびその構造によってキー付けされた)、Rational System Architect のメタ・モデル
MetaProperty	Rational System Architect のメタ・モデル・プロパティー・セット (各定義タイプ内のサポートされるプロパティーおよびその構造)

4

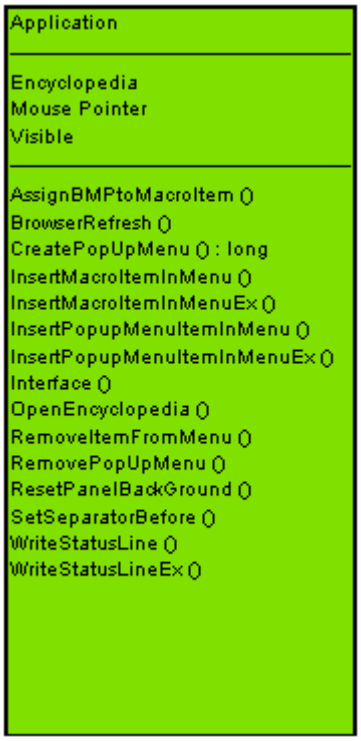
Application クラス

はじめに

Application クラスは、ユーザー・インターフェースを制御する場合に使用できる Rational System Architect アプリケーション・オブジェクトです。また、オブジェクト・モデルの中では最もハイレベルです。

アプリケーション・オブジェクトのインスタンスをインスタンス化するには、次のようにします。

```
Dim oApplication As SA2001.Application
Set oApplication = New Application
```



属性

Encyclopedia

目的
Encyclopedia クラスの属性およびメソッドへのアクセスを可能にする Encyclopedia オブジェクトです。

パラメーター
読取専用

例
```
Dim oEncyclopedia As Encyclopedia
Set oEncyclopedia = oApplication.Encyclopedia
```

MousePointer

目的
アプリケーションを使用する場合にユーザーに表示されるマウス・ポインターのタイプを制御できるようにします。

パラメーター
データ型: 整数

例
マウス・ポインターの現行値を戻すには、次のようにします。

```
Dim MouseValue as Integer
MouseValue  = oApplication.Mousepointer
```

マウス・ポインターを「砂時計」タイプに設定するには、設定値に 11 を使用します。

```
oApplication.Mousepointer  = 11
```

暗黙的値の詳細については、VB ヘルプ・ファイルを参照してください。

Visible

目的
アプリケーションが実行中かどうかを判別します。「False」に設定すると、アプリケーションが終了します。

パラメーター
データ型: ブール値

例
```
oApplication.Visible = False
```

メソッド

AssignBMPtoMacroItem

目的
メニュー・アイコンなどのビットマップを VBA マクロと関連付けます。

構文
```
Application Object.AssignBMPtoMacroItem MacroName,
  BMPFileName
```

Application Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

MacroName

使用: 必須

データ型: 文字列

任意の有効なマクロ・プロジェクト

構文: 「Project Name, Module Name, Subroutine Name()」

BMPFileName

使用: 必須

データ型: 文字列

BMP のファイル・パスおよびファイル名 (「C:\Windows\world.bmp」など)

BrowserRefresh

目的
Rational System Architect ブラウザーをリフレッシュします。最後のリフレッシュ以降にエンサイクロペディアに追加された項目をすべて表示させます。

構文
```
Application Object.BrowserRefresh
```

Application Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

CreatePopUpMenu

目的

ユーザー・インターフェースに挿入するためのポップアップ・メニューを作成します。ポップアップ・メニューにビットマップ・アイコンを関連付けることができます。

構文

Application Object.**CreatePopUpMenu**PopUpName[, BMPFileName]

Application Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

PopUpName

使用: 必須

データ型: 文字列

作成されるポップアップ・メニューの名前

BMPFileName

使用: オプション

データ型: 文字列

BMP のファイル・パスおよびファイル名 (「C:\Windows\world.bmp」など)

InsertMacroItemInMenu

目的

Rational System Architect メニューに、既存のマクロ・サブルーチンを参照するメニュー項目を作成します。

構文

```
Application Object.InsertMacroItemInMenu MacroName,
  MacroItemCaption, InMenuTitleCaption[,
  BeforeMenuItemCaption]
```

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`MacroName`

使用: 必須

データ型: 文字列

任意の有効なマクロ・プロジェクト

構文: 「Project Name, Module Name, Subroutine Name()」

`MacroItemCaption`

使用: 必須

データ型: 文字列

SA メニューに挿入されるマクロ項目の名前

`InMenuTitleCaption`

使用: 必須

データ型: 文字列

マクロ項目が配置される既存の SA ポップアップ・メニューの名前

`BeforeMenuItemCaption`

使用: オプション

データ型: 文字列

マクロ項目が直前に配置される、既存の SA メニュー項目の名前

注: 指定しない場合、マクロはポップアップ・メニューの最下部に配置されます。

InsertMacroItemInMenuEx

目的

Rational System Architect メニューに、既存のマクロ・サブルーチンを参照するメニュー項目を作成します。このメソッドは、InsertMacroItemInMenu メソッドを拡張したものです。

構文

```
Application Object.InsertMacroItemInMenuEx MacroName,
  MacroItemCaption, InMenuTitleCaption[,
  BeforeMenuItemCaption[, Tag[, bAfterSepartor]]]
```

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`MacroName`

使用: 必須

データ型: 文字列

任意の有効なマクロ・プロジェクト

構文: 「Project Name, Module Name, Subroutine Name()」

`MacroItemCaption`

使用: 必須

データ型: 文字列

SA メニューに挿入されるマクロ項目の名前

`InMenuTitleCaption`

使用: 必須

データ型: 文字列

マクロ項目が配置される既存の SA ポップアップ・メニューの名前

`BeforeMenuItemCaption`

使用: オプション

データ型: 文字列

マクロ項目が直前に配置される、既存の SA メニュー項目の名前

注: 指定しない場合、マクロはポップアップ・メニューの最下部に配置されます。

`Tag`

使用: オプション

データ型: 文字列

メニュー項目ごとに一意のタグを設定して、複数のメニュー項目に同じ名前を割り当てたり、それらのメニュー項目で同じサブルーチンを参照したりできるようにします。いずれかのメニュー項目を呼び出すと、タグによって、実行されるコード部分がサブルーチンに伝えられます。例えば、ユーザーはさまざまな Word レポートを作成するマクロを記述できます。Word レポートのタイプごとに異なるサブルーチンを記述しなくても、1 つのサブルーチンにすべてのコードを記述し、それぞれのメニュー項目内で、コード内の異なる関数をポイントする異なるタグを指定することができます。

`bAfterSepartor`

使用: オプション

データ型: ブール値

マクロ項目が直前に配置される、既存のメニュー項目が、その直前の項目と分離線で区切られている場合にのみ使用されます。True を入力した場合、マクロ項目は分離線の後に配置されます。False を入力するか、またはブランクのまま残した場合、マクロ項目は分離線の前に自動的に配置されます。

InsertPopUpMenuItemInMenu

目的

ポップアップ・メニュー項目を作成して、既存の Rational System Architect メニュー項目に挿入します。

構文

```
Application Object.InsertPopupMenuItemInMenu PopUpName,
  InMenuTitleCaption[, BeforeTitleCaption]
```

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`PopUpName`

使用: 必須

データ型: 文字列

作成されるポップアップ・メニューの名前

`InMenuTitleCaption`

使用: 必須

データ型: 文字列

新規ポップアップ・メニューが配置される既存の SA ポップアップ・メニューの名前

`BeforeMenuItemCaption`

使用: オプション

データ型: 文字列

新規ポップアップ・メニューが直前に配置される、既存の SA メニュー項目の名前

注: 指定しない場合、新規ポップアップ・メニューは既存ポップアップ・メニューの最下部に配置されます。

InsertPopUpMenuItemInMenuEx

目的

ポップアップ・メニュー項目を作成して、既存の Rational System Architect メニュー項目に挿入します。このメソッドは、InsertPopUpMenuItemInMenu メソッドを拡張したものです。

構文

```
Application Object.InsertPopupMenuItemInMenuEx PopUpName,
  InMenuTitleCaption[, BeforeTitleCaption[,
  bAfterSeparator]]
```

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`PopUpName`

使用: 必須

データ型: 文字列

作成されるポップアップ・メニューの名前

`InMenuTitleCaption`

使用: 必須

データ型: 文字列

新規ポップアップ・メニューが配置される既存の SA ポップアップ・メニューの名前

`BeforeMenuItemCaption`

使用: オプション

データ型: 文字列

新規ポップアップ・メニューが直前に配置される、既存の SA メニュー項目の名前

注: 指定しない場合、新規ポップアップ・メニューは既存ポップアップ・メニューの最下部に配置されます。

`bAfterSepartor`

使用: オプション

データ型: ブール値

マクロ項目が直前に配置される、既存のメニュー項目が、その直前の項目と分離線で区切られている場合にのみ使用されます。True を入力した場合、マクロ項目は分離線の後に配置されます。False を入力するか、またはブランクのまま残した場合、マクロ項目は分離線の前に自動的に配置されます。

Interface

目的

このメソッドはほとんど使用されませんが、明示的な参照でなく、テキスト・ストリングを使用して、インターフェースを呼び出すことができます。

例

```
Dim sa As Application
Set sa = New Application
Dim ob As Object
Set ob = sa.Interface("ISAIMF")
```

OpenEncyclopedia

目的

既存の Rational System Architect エンサイクロペディアを開きます。

構文

```
Application Object.OpenEncyclopedia(EncyclopediaPath)
```

```
Application Object
```

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

```
EncyclopediaPath
```

使用: 必須

データ型: 文字列

既存のエンサイクロペディアのファイル・パス

(EncyclopediaPath) は UDL ファイルです。UDL ファイルは、パス C:\Document and Settings\<user name>\Local Settings\Application Data\Telelogic\System Architect\Temp UDL ファイル内 に間接的に作成されます。これらの UDL ファイルには SA_563.udl のような名前が付けられます。このパスを表示するには、Rational System Architect を開く必要があります。

OpenEncyclopediaUsingConnectionString

目的

接続文字列を使用して、既存の Rational System Architect エンサイクロペディアを開きます。

構文

```
Application Object.OpenEncyclopediaUsingConnectionsString
  (strConnection)
```

```
Application Object
```

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

```
strConnection
```

使用: 必須

データ型: 文字列

```
strConnection は UDL ファイルの内容を示す文字列です。
```

以下に例を示します。

```
SA2001.OpenEncyclopediaUsingConnectionString
  ("Provider=SQLOLEDB.1;Integrated
  Security=SSPI;InitialCatalog=DoDAFABM;Data
  Source=SUZANNES\TLOGICSA106")
```

OpenEncyclopediaUsingDisplayName

目的

表示名を使用して、既存の Rational System Architect エンサイクロペディアを開きます。

構文

Application Object.OpenEncyclopediaUsingDisplayName(strDisplayName)

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`strDisplayName`

使用: 必須

データ型: 文字列

strDisplayName は SA キャプション・バーに表示される、connection-name(encyc-name) のような名前です。

例:

Sa2001.OpenEncyclopediaUsingDisplayName “Local Server SUZANNESTLOGICSA 106(Samples)”

RemoveItemFromMenu

目的

指定された Rational System Architect メニュー項目またはポップアップ・メニューからメニュー項目を除去します。

構文

```
Application Object.RemoveItemFromMenu ItemCaption,
  FromMenuTitleCaption
```

Application Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

ItemCaption

使用: 必須

データ型: 文字列

既存のポップアップ・メニューから除去するメニュー項目の名前

FromMenuTitleCaption

使用: 必須

データ型: 文字列

除去対象のメニュー項目が含まれている、既存のポップアップ・メニューの名前

RemovePopUpMenu

目的

Rational System Architect メニュー・システムから、指定されたポップアップ・メニューを除去します。

構文

Application Object.**RemovePopUpMenu**(PopUpName)

Application Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

PopUpName

使用: 必須

データ型: 文字列

除去するポップアップ・メニューの名前

ResetPanelBackGround

目的
ステータス・バー・パネルの背景色をリセットします。

構文
```
Application Object.ResetPanelBackGround(Panel)
```

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`Panel`

使用: 必須

データ型: long

パネルはステータス・バーの「ペイン」 (セクション) であり、左がパネル 1、右がパネル 4 です (パネル 2 および 3 が表示されるのは、シンボルが選択された場合のみです)。

SetSeparatorBefore

目的
指定されたメニュー内のメニュー項目の直前に分離線を配置します。

構文
```
Application Object.SetSeparatorBefore ItemCaption,
  FromMenuTitleCaption, bHasSeparator)
```

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`ItemCaption`

使用: 必須

データ型: 文字列

分離線が直前に配置されるメニュー項目の名前。

`FromMenuTitleCaption`

使用: 必須

データ型: 文字列

分離線が配置される既存の SA ポップアップ・メニューの名前。

`bHasSeparator`

使用: 必須

データ型: ブール値

分離線が配置されるかどうかに応じて、True または False 値を設定します。

WriteStatusLine

目的

短いメッセージをユーザーに伝えて、コードの実行中に、Rational System Architect のステータス・バー (左下隅のバー) に情報を常に表示しておくことができます。

構文

`Application Object.`**`WriteStatusLine`**`(TextToShow)`

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`TextToShow`

使用: 必須

データ型: 文字列

ステータス・バーに表示されるテキスト

WriteStatusLineEx

目的

短いメッセージをユーザーに伝えて、コードの実行中に、Rational System Architect のステータス・バー (左下隅のバー) に情報を常に表示しておくことができます。このメソッドは、WriteStatusLine メソッドを拡張したものです。

構文

```
Application Object.WriteStatusLineEx(Panel, TextToShow,
  BackColor, ForeColor)
```

`Application Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Application クラス

`Panel`

使用: 必須

データ型: long

パネルはステータス・バーの「ペイン」 (セクション) であり、左がパネル 1、右がパネル 4 です (パネル 2 および 3 が表示されるのは、シンボルが選択された場合のみです)。

`TextToShow`

使用: 必須

データ型: 文字列

ステータス・バーに表示されるテキスト

`BackColor`

使用: 必須

データ型: long

ステータス・バーの背景色

`ForeColor`

使用: 必須

データ型: long

ステータス・バーの前景色

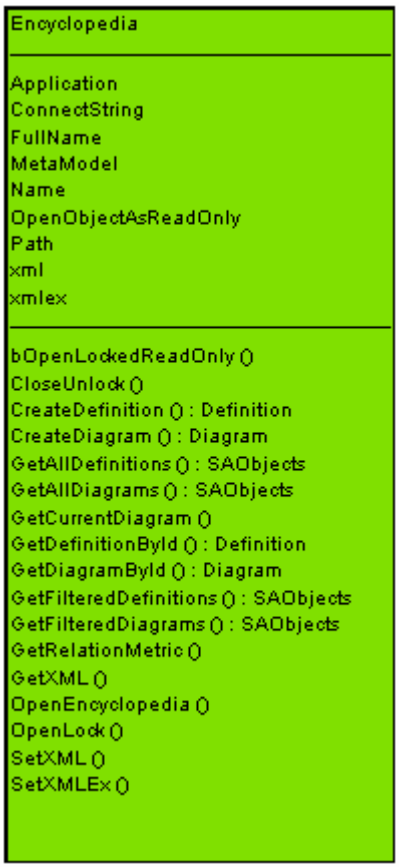
5

Encyclopedia クラス

はじめに

Encyclopedia クラスは、Encyclopedia オブジェクトです。このクラスを次のように使用すると、エンサイクロペディアの属性およびメソッドにアクセスできます。

```
Dim oApplication As SA2001.Application, oEncyclopedia
  As Encyclopedia
Set oApplication = New Application
Set oEncyclopedia = oApplication.Encyclopedia
```



属性

Application

目的
アプリケーション・オブジェクトは、現在の Encyclopedia オブジェクトの親アプリケーション・オブジェクトを戻します。

パラメーター
読取専用

ConnectString

目的
エンサイクロペディアと接続するために必要な情報です。

パラメーター:
データ型: 文字列

読取専用

FullName

目的
絶対パスを含む、現在のエンサイクロペディアの名前です。

パラメーター
読取専用

データ型: 文字列

MetaModel

目的
MetaModel クラスです。すべての MetaModel 属性にアクセスできるようにします。

パラメーター
読取専用

Name

目的
現在のエンサイクロペディアの名前を戻します。

パラメーター
読取専用

データ型: 文字列

OpenObjectsAsReadOnly

目的
SA オブジェクト・モデルのすべてのオブジェクトを読み取り専用として開くかどうかを設定します。

パラメーター
データ型: ブール値

パス

目的
現在のエンサイクロペディアのパスです。

パラメーター
データ型: 文字列

読取専用

Xml

目的

エンサイクロペディアの XML 文字列です。GetXML および SetXML メソッドで操作します。

パラメーター
データ型: 文字列

XmlEx

目的
エンサイクロペディアの XML 文字列です。SetXMLEx メソッドで操作します。

パラメーター
データ型: 文字列

メソッド

bOpenLockedReadOnly

目的
OpenObjectsAsReadOnly 属性が True に設定されているか、またはエンサイクロペディアが読み取り専用で開かれた場合は、True を返します。

CloseUnlock

後述の OpenLock を参照してください。

CreateDefinition

目的
指定された定義名および定義タイプを持つ定義クラスのインスタンスを作成します。

構文
```
Encyclopedia Object.CreateDefinition(Name, SAType)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`Name`

使用: 必須

データ型: 文字列

新規定義の名前

`SAType`

使用: 必須

データ型: Long

作成している Rational System Architect 定義のタイプ (DFXPROCESS や 3 など)

注: すべての SA 定義および内部定数の名前と番号の詳細については、Rational System Architect ディレクトリー内の DEFNS.BAS ファイルを参照してください。

注: SA 定義を正常に作成するには、定義の Save メソッドを呼び出す必要があります。そうしないと、エンサイクロペディアを終了した場合に、新規定義が削除されます。

CreateDiagram

目的

指定されたダイアグラム名およびダイアグラム・タイプを持つ、ダイアグラム・クラスのインスタンスを作成します。

構文

```
Encyclopedia Object.CreateDiagram(Name, SAType)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`Name`

使用: 必須

データ型: 文字列

新規ダイアグラムの名前

`SAType`

使用: 必須

データ型: Long

作成している Rational System Architect ダイアグラムのタイプ (GTCATPROCESSFLOW や 89 など)

注: すべての SA ダイアグラムおよび内部定数の名前と番号の詳細については、Rational System Architect ディレクトリー内の DIAGRAMS.BAS ファイルを参照してください。

GetAllDefinitions

目的
エンサイクロペディア内のすべての定義を定義コレクションとして戻します。

ルール
SAObjects 変数の次元を設定し、定義のコレクションとして設定する必要があります。

例

```
Dim oCollectionofDefinitions As SAObjects

Set oCollectionofDefinitions =
  oEncyclopedia.GetAllDefinitions

Call oCollectionofDefinitions.ReadAll
```

SAObjects コレクションは、コレクションの Complete フラグが True になるまでは、完全に取り込まれません。GetAllDefinitions は、ReadAll または IsMoreThan メソッドと組み合わせて使用する必要があります。

GetAllDiagrams

目的
エンサイクロペディア内のすべてのダイアグラムをダイアグラム・コレクションとして戻します。

ルール
SAObjects 変数の次元を設定し、ダイアグラムのコレクションとして設定する必要があります。

例

```
Dim oCollectionofDiagrams As SAObjects

Set oCollectionofDiagrams =
  oEncyclopedia.GetAllDiagrams

Call oCollectionofDiagrams.ReadAll
```

SAObjects コレクションは、コレクションの Complete フラグが True になるまでは、完全に取り込まれません。GetAllDiagrams は、ReadAll または IsMoreThan メソッドと組み合わせて使用する必要があります。

GetCurrentDiagram

目的
現在開かれているダイアグラムをダイアグラム・オブジェクトとして戻します。

ルール
ダイアグラム・オブジェクトの次元を設定し、現在開かれているダイアグラムとして設定する必要があります。次の例を参照してください。

例

```
Dim OCurrentDiagram As Diagram
Set OCurrentDiagram = oEncyclopedia.GetCurrentDiagram
```

GetDefinitionById

目的
指定された ID から定義オブジェクトとして定義を戻します。

構文

```
Encyclopedia Object.GetDefinitionById(Id)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`Id`

使用: 必須

データ型: Long

Rational System Architect に格納されたすべての定義は、データ・ディクショナリー ID を使用して、内部で一意に識別されます。

例

```
Dim oDefinition As Definition
Set oDefinition = oEncyclopedia.GetDefinitionById(12)
```

GetDiagramById

目的
Rational System Architect に格納されたすべてのダイアグラムは、データ・ディクショナリー ID を使用して、内部で一意に識別されます。このメソッドは、指定された ID からダイアグラム・オブジェクトとしてダイアグラムを戻します。

構文
```
Encyclopedia Object.GetDiagramById(Id)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`Id`

使用: 必須

データ型: Long

Rational System Architect に格納されたすべてのダイアグラムは、データ・ディクショナリー ID を使用して、内部で一意に識別されます。

例
```
Dim oDiagram  As Diagram
Set oDiagram = oEncyclopedia.GetDiagramById(2)
```

GetFilteredDefinitions

目的
エンサイクロペディアの定義コレクションをフィルタリングして戻します。

パラメーター
データ型: SAObjects

構文
```
Encyclopedia Object.GetFilteredDefinitions(WildCardName,
  SAType)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`WildCardName`

使用: 必須

データ型: 文字列

フィルター基準 (「C」で開始するすべての定義を表す「C*」など)

注: ワイルドカード検索では大/小文字の区別が行われます。

`SAType`

使用: 必須

データ型: Long

検索している Rational System Architect 定義のタイプ (DFXPROCESS や 3 など)

注: すべての SA 定義および内部定数の名前と番号の詳細については、Rational System Architect ディレクトリー内の DEFNS.BAS ファイルを参照してください。

例

次の例では、「C」で開始するプロセス定義をすべて戻します。

```
Dim oCollectionofDefinitions As SAObjects
Set oCollectionofDefinitions =
  oEncyclopedia.GetFilteredDefinitions("C*",
  DFXPROCESS)
Call oCollectionofDefinitions.ReadAll
```

SAObjects コレクションは、コレクションの Complete フラグが True になるまでは、完全に取り込まれません。GetFilteredDefinitions は、ReadAll または IsMoreThan メソッドと組み合わせて使用する必要があります。

GetFilteredDiagrams

目的

エンサイクロペディアのダイアグラム・コレクションをフィルタリングして戻します。

パラメーター

データ型: SAObjects

構文

```
Encyclopedia Object.GetFilteredDiagrams(WildCardName,
  SAType)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`WildCardName`

使用: 必須

データ型: 文字列

フィルター基準 (「C」で開始するすべてのダイアグラムを表す「C*」など)

注: ワイルドカード検索では大/小文字の区別が行われます。

`SAType`

使用: 必須

データ型: Long

検索している Rational System Architect ダイアグラムのタイプ (GTCATPROCESSFLOW や 89 など)

注: すべての SA ダイアグラムおよび内部定数の名前と番号の詳細については、Rational System Architect ディレクトリー内の DIAGRAMS.BAS ファイルを参照してください。

例

次の例では、「Pr」で開始する Gane & Sarson ダイアグラムをすべて戻します。

```
Dim oCollectionofDiagrams As SAObjects
Set oCollectionofDiagrams =
  oEncyclopedia.GetFilteredDiagrams("Pr*", GTDFDGS)
Call oCollectionofDiagrams.ReadAll
```

SAObjects コレクションは、コレクションの Complete フラグが True になるまでは、完全に取り込まれません。GetFilteredDiagrams は、ReadAll または IsMoreThan メソッドと組み合わせて使用する必要があります。

GetRelationMetric

目的

エンサイクロペディア内の 2 つの Rational System Architect オブジェクト間の関係情報または振る舞いを取得します。

構文

```
Encyclopedia Object.GetRelationMetric SAObject1,
  SAObject2, Relation, Depth, Metric, FieldType[,
  NbrChars[, NbrDec]]
```

Encyclopedia Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

SAObject1

使用: 必須

データ型: オブジェクト

関係メトリックを実行するために必要な 2 つの必須 Rational System Architect オブジェクトのうちの 1 つ

SAObject2

使用: 必須

データ型: オブジェクト

関係メトリックを実行するために必要な 2 つの必須 Rational System Architect オブジェクトのうちの 1 つ

関係

使用: 必須

データ型: RELATETYPE

パラメーター化された上記の 2 つの SA オブジェクト間の関係のタイプです。すべての Rational System Architect 関係タイプのリストおよびそれらの説明については、第 16 章を参照してください。

深さ

使用: 必須

データ型: Long

パラメーター化された上記の 2 つの SA オブジェクト間の関係を示す数です。例えば、Object 1 というデータ構造の中に、データ要素の Object 2 が含まれている場合、2 つのオブジェクト間の「深さ」は 1 です。

`Metric`

使用: 必須

データ型: RELATIONMETRIC

関係メトリック。すべての関係メトリックの詳細については、以下を参照してください。

`FieldType`

使用: 必須

データ型: FLDTYPE

フィールド・タイプ。Rational System Architect のフィールド・タイプの詳細については、第 17 章を参照してください。

`NbrChars`

使用: オプション

データ型: Long

SA によって戻される、小数点の前に配置される文字数

`NbrDec`

使用: オプション

データ型: Long

SA によって戻される、小数点の後に配置される文字数

GetXML

目的

エンサイクロペディアの XML 文字列を有効な .xml ファイルにエクスポートします。

構文

```
Encyclopedia Object.GetXML strXML, bToFile
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`StrXML`

使用: 必須

データ型: 文字列

`bToFile` が True に設定されている場合、これは、SA がエンサイクロペディアの XML をエクスポートするエクスポート先を示す、有効な XML ファイル名です。`bToFile` が False に設定されている場合、strXML は XML 文字列として機能します。

`bToFile`

使用: 必須

データ型: ブール値

True の場合は、strXML パラメーターで指定されたファイルが作成されます。False の場合は、strXML にエンサイクロペディア XML 文字列が読み込まれます。

OpenLock…CloseUnlock ステートメント

目的

OpenLock および CloseUnlock メソッドは、現在の Rational System Architect エンサイクロペディアのロック状況を制御します。これにより、VBA 処理の実行中に、読み取り専用アクセス、読み取り/書き込みアクセス、または更新アクセスのためにエンサイクロペディアがロックされているかどうかを判別します。

OpenLock が特定のモードで実行される場合は、コードの後半に CloseUnlock を同じモードで実行する必要があります。

エンサイクロペディアでさまざまなレベルのロックが必要な場合は、コード内で OpenLock および CloseUnlock メソッドを複数回実行できます。

VBA コードで OpenLock および CloseUnlock メソッドが実行されない場合、Rational System Architect は Object Model メソッドが発行されるたびに、必要に応じて独自にロックを実行します。これにより、マクロのパフォーマンスが影響を受けることがあります。

2 つのメソッドは、呼び出しが正常に行われたかどうかを示すブール値を戻します。

構文

```
Encyclopedia Object.OpenLock(LockMode)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`LockMode`

使用: 必須

データ型: EncyLockMode

現在の Rational System Architect エンサイクロペディアのロック状況

EncyLockMode	意味
NETOPENREAD	読取専用
NETOPENREADWRITE	読み取り/書き込み
NETOPENUPDATE	VBA アプリケーション実行中の更新アクセス

例

```
Dim sa As Application
Set sa = New Application
sa.Encyclopedia.OpenLock NETOPENREAD
    ' execute SA Code here
sa.Encyclopedia.CloseUnLock NETOPENREAD
Set sa = Nothing
```

SetXML

目的

エンサイクロペディアに .xml ファイルをインポートします。

構文

```
Encyclopedia Object.SetXML(strXML, bFromFile, bValidate)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`StrXML`

使用: 必須

データ型: 文字列

bFromFile が True に設定されている場合、これは、SA が XML コードをインポートするインポート元を示す、有効な .xml ファイル名です。bFromFile が False に設定されている場合、これはエンサイクロペディアの XML 文字列です。

`bFromFile`

使用: 必須

データ型: ブール値

True の場合は、strXML パラメーターで指定されたファイルから XML がインポートされます。False の場合は、strXML にエンサイクロペディア XML 文字列が読み込まれます。

`bValidate`

使用: 必須

データ型: ブール値

True の場合は、パーサーで XML 文字列が検証されます。

SetXMLEx

目的

構文

```
Encyclopedia Object.SetXMLEx(strXML, ICollision,
  bFromFile, bValidate)
```

`Encyclopedia Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Encyclopedia クラス

`StrXML`

使用: 必須

データ型: 文字列

bFromFile が True に設定されている場合、これは、SA が XML コードをインポートするインポート元を示す、有効な .xml ファイル名です。bFromFile が False に設定されている場合、これはエンサイクロペディアの XML 文字列です。

`ICollision`

使用: 必須

データ型: Long

競合オプション	説明
0	既存の定義またはダイアグラムは上書きされません。
1	定義が存在する場合は、定義のプロパティーをすべて取得してから、定義を削除して再作成し、取得したプロパティーを再び取り込みます。
2	データ供給時にフィールドを更新
3	フィールドを更新 - データがなければクリア
256	常に既存の図を置き換える

`bFromFile`

使用: 必須

データ型: ブール値

True の場合は、strXML パラメーターで指定されたファイルから XML がインポートされます。False の場合は、strXML にエンサイクロペディア XML 文字列が読み込まれます。

`bValidate`

使用: 必須

データ型: ブール値

True の場合は、パーサーで XML 文字列が検証されます。

ワークスペース・メソッド

Rational System Architect V 11.3 にワークスペースが追加されましたが、既存マクロに影響はありません。これは、ワークスペース・オブジェクトがオブジェクト・モデルに導入されていないためです。Rational System Architect では一度に単一のワークスペースにしかアクセスが許可されないので、その結果ワークスペース・オブジェクトはほぼ完全に Encyclopedia オブジェクトにマッピングされます。ユーザーがワークスペースを操作するために、オブジェクト・モデルに対して以下の機能拡張が行われました。

IsEncyWorkspaceEnabled

目的
現在のエンサイクロペディアがワークスペース対応である場合、True を返します。

GetWorkspaceID

目的
現在のワークスペース ID を戻します。

SetWorkspaceID

目的
現在のワークスペースを変更します。

GetWorkspaceTree

目的
ワークスペースの名前、ID、およびベースライン・ステータスを含む XML ツリーを戻します。

GetWorkspaceName

目的
現在のワークスペース名を戻します。

IsWorkspaceReadOnly

目的
現在のワークスペースが読み取り専用である場合、True を返します。

ワークスペース・アプリケーション・クラスの呼び出し

ワークスペース用のアプリケーション・クラスの呼び出しを以下に示します。

OpenEncyclopediaUsingConnectionStringAndWorkspace

目的
ワークスペース対応バージョンの OpenEncyclopediaUsingConnectionString

OpenEncyclopediaUsingDisplayNameAndWorkspace

目的
ワークスペース対応バージョンの OpenEncyclopediaUsingDisplayName

ワークスペース・アプリケーション・クラスのイベント

ワークスペース用のアプリケーション・クラスのイベントを以下に示します。

WorkspaceOpen

目的
ワークスペース変更時に発生したイベント

WorkspaceBeforeOpen

目的
キャンセル可能なワークスペースの変更前に発生したイベント。エンサイクロペディアの変更時と同様に、定義およびダイアグラムへのオブジェクト・モデルの参照はすべて無効になることに注意してください。

注: これは Rational System Architect V 11.3.0.2 以降でのみ適用されます。

関係メトリック

関係メトリックはダイアグラム、シンボル、および定義メトリックと異なり、エンサイクロペディア内の 2 つの Rational System Architect オブジェクト間の関係に関する情報を検索する内部機能を構成しています。関係メトリックごとに、どの 2 つのオブジェクトを調査するのか、およびどのような関係がオブジェクト間に存在するのかを宣言する必要があります。利用している関係メトリックに応じて、特定の関係を持つ特定の Rational System Architect オブジェクトのみが有効となります。

これらの関係メトリックにアクセスするには、Encyclopedia クラスで GetRelationMetric メソッドを呼び出す必要があります。SA オブジェクト・ブラウザーの RELATIONMETRIC 列挙型リストには、すべての関係メトリックの一覧があります。次の表に、使用可能なすべての関係メトリック、およびその説明とパラメーターを示します。

関係メトリック	説明	パラメーター
RELMETCRUD	データ・ストアのプロセス名の次に表示される CRUD 文字 (Create、Read、Update、Delete) の組み合わせを文字列として戻します。	SAObjects: データ・ストア、プロセス
RELMETDEPTH	2 つのオブジェクト間の「Uses」関係の個数を数値として戻します。	SAObjects: 「Uses」または「Used By」関係が少なくとも 1 つ 必要です。
RELMETICOMROLE	ICOM 矢印と接続先の機能/アクティビティー・シンボル間の関係ロール (入力、制御、出力、メカニズム、または境界) を文字列として戻します。	SAObjects: ICOM 矢印、機能/アクティビティー RelTypes: RELCONNSTART、RELSTARTAT、RELCONNEND、RELENDAT
RELMETINPUT	フロー・シンボルが別のシンボルまたはダイアグラムに「流入」しているかどうかを調べます。ブール値フィールドを戻します。	SAObjects: フロー・シンボル、ダイアグラムまたはノード・シンボル RelType: RELCONNSTART、RELSTARTAT、RELCONNEND、

関係メトリック	説明	パラメーター
		RELENDAT、 RELDIAGRAMCON、 RELCONDIAGRAM
RELMETOUTPUT	フロー・シンボルが別のシンボルまたはダイアグラムから「流出」しているかどうかを調べます。ブール値フィールドを戻します。	SAObjects: フロー・シンボル、ダイアグラムまたはノード・シンボル RelType: RELCONNSTART、 RELSTARTAT、 RELCONNEND、 RELENDAT、 RELDIAGRAMCON、 RELCONDIAGRAM
RELMETSTATETABLE	状態に接続された、出力遷移ラインに関連付けられているイベントの名前が、状態定義ダイアログに示されているかどうかを調べます。True の場合は、状態名を戻します。	SAObjects: Shlaer 状態、Shlaer 遷移ライン RelType: RELCONNEND、 RELCONNSTART

6

Diagram クラス

はじめに

これは、エンサイクロペディアに含まれるダイアグラムのインスタンスです。

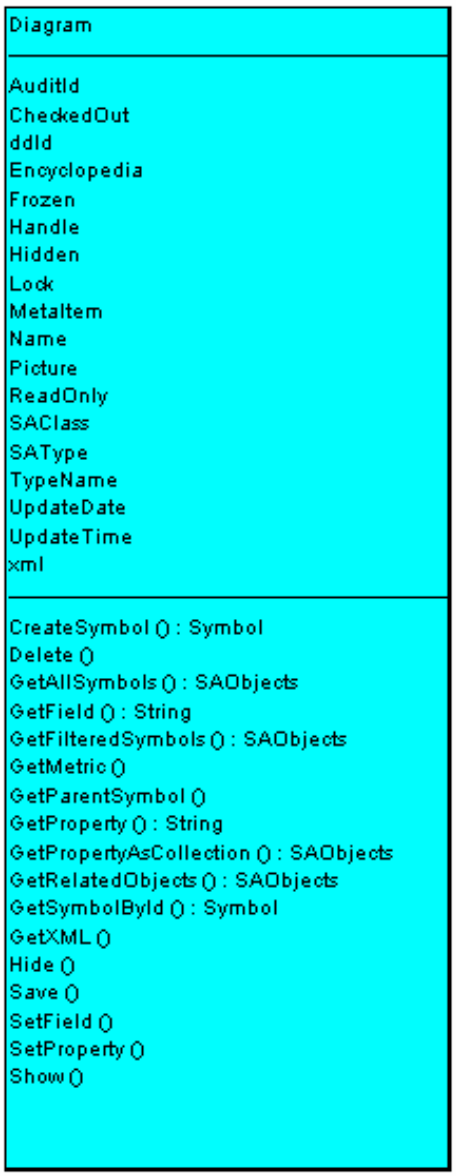
現在アクティブなダイアグラムをオブジェクトとして返すには、以下を使用します。

```
Dim oApplication As
  SA2001.Application

Dim oDiagram As Diagram

Set oApplication = New
  Application

Set oDiagram =
  oApplication.oEncyclopedia.
  GetCurrentDiagram
```



属性

AuditID

目的
Rational System Architect に格納されたすべての項目ダイアグラムには、そのダイアグラムの作成者や最終更新者の ID が AuditId としてタグ付けされています。

パラメーター
データ型: 文字列

読取専用

CheckedOut

目的
True に設定すると、ダイアグラムはチェックアウトを行った AuditID 以外のすべてのユーザーに対して読み取り専用になります。

パラメーター
データ型: ブール値

ddID

目的
Rational System Architect に格納されたすべてのダイアグラムは、データ・ディクショナリー ID を使用して、内部で一意に識別されます。このメソッドは、ダイアグラムの ID を返します。

パラメーター
データ型: Long

Encyclopedia

目的
親エンサイクロペディアの属性およびメソッドを使用したアクセスを容易にします。

パラメーター
読取専用

Frozen

目的
この属性を設定するユーザーには、フリーズ特権が必要です。True に設定すると、ダイアグラムは、ダイアグラムをフリーズした AuditID を含むすべてのユーザーに対して読み取り専用になります。

パラメーター
データ型: ブール値

Handle

目的
実行時にのみ使用可能なダイアグラムのメモリー・ハンドルです。このハンドルは一意ではなく、アクセス時に同じになることはほとんどありません。

パラメーター
データ型: Long

読取専用

例

```
Dim oDiagram as Diagram, Handle As Long
Set oDiagram = oEncyclopedia.GetCurrentDiagram
Handle = oDiagram.Handle
```

Hidden

目的
ダイアグラムが閉じているか、または開いているかを示す「True」または「False」が返されます。

パラメーター
データ型: ブール値

読取専用

Locked

目的
ダイアグラムがロックされているかどうか、つまりユーザーがそのダイアグラムを使用中であるかどうかを示す「True」または「False」の値を返します。

パラメーター
データ型: ブール値

読取専用

MetaItem

目的
MetaItem クラスおよびその属性を使用したアクセスを容易にします。

パラメーター
読取専用

Name

目的

指定されたダイアグラム・オブジェクトの名前。

パラメーター
データ型: 文字列

読取専用

Picture

目的
ダイアグラムが保存された後、Rational System Architect は、ダイアグラムの Windows メタファイル (.wmf) を作成します。このファイルは、エンサイクロペディアのディレクトリーに保存されます。この属性によって、ユーザーは stdPicture 属性およびメソッド、ピクチャーのコンテンツにデータを保持する OLE オートメーション・オブジェクトにアクセスできます。

パラメーター
データ型: stdPicture

読取専用

ReadOnly

目的
ダイアグラムが読み取り専用として開かれた場合に「True」を返します。

パラメーター
データ型: ブール値

読取専用

SAClass

目的
ダイアグラムのクラス・タイプ。メジャー・タイプ番号とも呼ばれます。

パラメーター
データ型: Long

読取専用

注: ダイアグラムに対して常に「1」を返します。

SAType

目的
ダイアグラムの定数 (整数)。Rational System Architect のすべてのダイアグラムには、一意の数値の定数 ID があります。

パラメーター
データ型: Long

読取専用

TypeName

目的
文字列で表されるダイアグラムのタイプ (「エンティティー・リレーション」など)。

パラメーター
データ型: 文字列

読取専用

UpdateDate

目的
ダイアグラムが最後に変更された日付。

パラメーター
データ型: 文字列

読取専用

UpdateTime

目的
ダイアグラムが最後に変更された時間。

パラメーター
データ型: 文字列

読取専用

xml

目的
ダイアグラムの xml 文字列。GetXML メソッドで操作します。

パラメーター
データ型: 文字列

読取専用

メソッド

CreateSymbol

目的
特定の名前およびタイプを指定して、Symbol クラスのインスタンスを作成します。

構文
```
Diagram Object.CreateSymbol(Name, SAType)
```

`Diagram Object`

使用: 必須

データ型: オブジェクト

シンボルの追加先である、インスタンス化された任意の Diagram クラス。

`Name`

使用: 必須

データ型: 文字列

新しいシンボルの名前。

`SAType`

使用: 必須

データ型: Long

作成中の Rational System Architect のシンボルのタイプ (例: ETCATELEMBUSPROC または 445)

注: すべての SA シンボルおよびその内部定数名と番号の詳細については、Rational System Architect ディレクトリーの SYMBOLS.BAS ファイルを参照してください。

注: ダイアグラム上に SA シンボルを正常に作成するには、Diagram クラスの Save メソッドを呼び出す必要があります。そうしないと、エンサイクロペディアを閉じるときに、新しいシンボルがダイアグラムから削除されます。

Delete

目的
ダイアグラム・オブジェクトによって指定されたダイアグラムを削除します。

GetAllSymbols

目的
このメソッドは、指定されたダイアグラム・オブジェクトに含まれるすべてのシンボルを SAObjects コレクションとして返します。SAObjects コレクションは、コレクションの Complete フラグが True になるまでは、完全に取り込まれません。GetAllSymbols は、ReadAll メソッドまたは IsMoreThan メソッドのいずれかとともに使用する必要があります。

ルール
SAObjects 変数は、次元を設定し、シンボルのコレクションとしてする必要があります。次の例を参照してください。

例

```
Dim oDiagram as Diagram, oCollectionofSymbols As
  SAObjects
Set oCollectionofSymbols = oDiagram.GetAllSymbols
Call oCollectionofSymbols.ReadAll
```

GetField

目的
これらは、「シンボルのグリッド・サイズ」、「ラインのグリッド・サイズ」、「レベル番号」など、ダイアグラムの特性です。これらの特性には、「ダイアグラム名」のように設定できるものと「ダイアグラム・タイプ」のように設定できないものがあります。

構文

```
Diagram Object. GetField FieldID
```

```
Diagram Object
```

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

```
FieldID
```

使用: 必須

データ型: DGMFLD

ダイアグラム・フィールド。すべてのダイアグラム・フィールドの詳細については、以下を参照してください。

GetFilteredSymbols

目的

ダイアグラムに含まれるシンボルをフィルタリングするには、最初の引数にフィルター基準をワイルドカードとして指定し、2 番目の引数に対象のシンボルのタイプを指定します。引数の一部は“”とすることもできます。これにより、SAObjects コレクションが返されます。SAObjects コレクションは、コレクションの Complete フラグが True になるまでは、完全に取り込まれません。GetFilteredSymbols は、ReadAll メソッドまたは IsMoreThan メソッドと共に使用する必要があります。

パラメーター

データ型: SAObjects

構文

```
Diagram Object.GetFilteredSymbols(WildCardName, SAType)
```

```
Diagram Object
```

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

```
WildCardName
```

使用: 必須

データ型: 文字列

フィルター基準 (例: “C”=「C」で始まるすべてのシンボル)

注: ワイルドカード検索では大/小文字の区別が行われます。

`SAType`

使用: 必須

データ型: Long

作成中の Rational System Architect のシンボルのタイプ (例: ETCATELEMBUSPROC または 445)

注: すべての SA シンボルおよびその内部定数名と番号の詳細については、Rational System Architect ディレクトリーの SYMBOLS.BAS ファイルを参照してください。

例

この例では、ダイアグラムに含まれる、文字「P」で始まるすべての「エンティティー」シンボルが返されます。

```
Dim oDiagram as Diagram, oCollectionofSymbols As
  SAObjects
Set oCollectionofSymbols =
  oDiagram.GetFilteredSymbols("P", ETECACTIVITY)
Call oCollectionofSymbols.ReadAll
```

Get Metric

目的

ダイアグラムに関連する特定のリスト、計算、および内部機能の一部を呼び出します。

構文

```
Diagram Object.GetMetric Metric[, FieldType[, NbrChars[,
  NbrDec]]]
```

`Diagram Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

`Metric`

使用: 必須

データ型: DIAGRAMMETRIC

ダイアグラムのメトリック。すべてのダイアグラムのメトリックの詳細については、以下を参照してください。

`FieldType`

使用: オプション

データ型: FLDTYPE

フィールド・タイプ。Rational System Architect のフィールド・タイプの詳細については、第 17 章を参照してください。

`NbrChars`

使用: オプション

データ型: Long

フィールド・タイプが入力されている場合は、このパラメーターによって、小数点の前に戻す文字の数が SA に伝えられます。

`NbrDec`

使用: オプション

データ型: Long

フィールド・タイプが入力されている場合は、このパラメーターによって、小数点の後に戻す数字の個数が SA に伝えられます。

GetParentSymbol

目的

ダイアグラムは、データ・フロー・ダイアグラムと同様に、親シンボルの子である場合があります。このメソッドは、指定されたダイアグラム・オブジェクトの親シンボル・オブジェクトを返します。

GetProperty

目的
指定されたダイアグラム・プロパティーの内容を返します。

パラメーター

プロパティーの実名が、定義ダイアログに表示されるものと異なる場合があります。例えば、プロセス・チャート図のエレメンタリー・ビジネス・プロセスでは、「場所」プロパティーは、実際には名前変更されています。本当のプロパティーは「場所タイプ」です。このことは、saprops.cfg でエレメンタリー・ビジネス・プロセスの定義を検索し、プロパティーが実際には「場所タイプ」と呼ばれているにもかかわらず、ラベル名が「場所」になっていることを確認することによって初めてわかります。

```
Property "Location Types" { Edit Listof "Location" Label "Locations"  LENGTH 2000
        HELP "Supporting Location Types (Matrix)"   READONLY }
```

構文
```
Diagram Object.GetProperty Name
```

`Diagram Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

`Name`

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

GetPropertyAsCollection

目的

プロパティーの中には、他のプロパティーとの関係を定義するものがあります。例えば、プロセス・チャートは、「プロセス・スレッド」プロパティーを介してプロセス・スレッドを参照します。このメソッドは、OneOf および ListOf のダイアグラムまたは定義のコレクションを返します。OneOf および ListOf のプロパティー・タイプの詳細については、第 14 章を参照してください。

パラメーター

データ型: OfCollection

プロパティーの実名が、定義ダイアログに表示されるものと異なる場合があります。例えば、プロセス・チャート図のエレメンタリー・ビジネス・プロセスでは、「場所」プロパティーは、実際には名前変更されています。本当のプロパティーは「場所タイプ」です。このことは、saprops.cfg でエレメンタリー・ビジネス・プロセスの定義を検索し、プロパティーが実際には「場所タイプ」と呼ばれているにもかかわらず、ラベル名が「場所」になっていることを確認することによって初めてわかります。

```
Property "Location Types" { Edit Listof "Location" Label "Locations"  LENGTH 2000
        HELP "Supporting Location Types (Matrix)"   READONLY }
```

構文

```
Diagram Object.GetPropertyAsCollection(PropName)
```

`Diagram Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

`PropName`

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

例

```
Dim i As Long, DiagId As Long
i = 0
Do While sa.Encyclopedia.GetFilteredDiagrams("",
  GTCATPROCESSFLOW).IsMoreThan(i)
    i = i + 1
    Dim ThreadColl As OfCollection
    Set SADiag =
  sa.Encyclopedia.GetFilteredDiagrams("",
  GTCATPROCESSFLOW).Item(i)
    Set ThreadColl =
  SADiag.GetPropertyAsCollection("Process Thread")
Loop
```

GetRelatedObjects

目的

このメソッドは、関連オブジェクトの SAObjects コレクションを現在のダイアグラム・オブジェクトに返します。

構文

```
Diagram Object.GetRelatedObjects(RelType)
```

`Diagram Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

`RelType`

使用: 必須

データ型: RELATETYPE

SA 関係。すべての関係の詳細については、第 16 章を参照してください。

GetSymbolById

目的
Rational System Architect に格納されたすべてのダイアグラムは、データ・ディクショナリー ID を使用して、内部で一意に識別されます。このメソッドは、ID からシンボルをオブジェクトとして返します。

構文
```
Diagram Object.GetSymbolById(Id)
```

Diagram Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

Id

使用: 必須

データ型: Long

Rational System Architect に格納されたすべてのシンボルは、データ・ディクショナリーの ID を使用して内部で一意に識別されます。

例
```
Dim oSymbol As Symbol
Set oSymbol = oDiagram.GetSymbolById(12)
```

GetXML

目的
ダイアグラムの XML 文字列を有効な .xml ファイルにエクスポートします。

構文
```
Diagram Object.GetXML(strXMLTextOut)
```

Diagram Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

`StrXMLTextOut`

使用: 必須

データ型: 文字列

SA がダイアグラムの XML 文字列をエクスポートする有効な .xml ファイル。

Hide

目的

現在開いているダイアグラムのインスタンスを閉じるために使用されます。

構文

```
Call oDiagram.Hide
```

Save

目的

ダイアグラムのインスタンスを保存するために使用されます。

構文

```
Call oDiagram.Save
```

SetField

目的

SetField によって、指定した値をダイアグラム・フィールドに設定できます。

構文

```
Diagram Object.SetField FieldID, value
```

Diagram Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

FieldID

使用: 必須

データ型: DGMFLD

ダイアグラム・フィールド。すべてのダイアグラム・フィールドの詳細については、以下を参照してください。

値

使用: 必須

データ型: 文字列

ダイアグラム・フィールドの値。

SetProperty

目的

ダイアグラム・プロパティーの値を設定するには、最初の引数としてプロパティーの名前、2 番目の引数として設定する値を指定する必要があります。プロパティー名は saprops.cfg および usrprops.txt ファイルで調べることができます。

パラメーター

プロパティーの実名が、定義ダイアログに表示されるものと異なる場合があります。例えば、プロセス・チャート図のエレメンタリー・ビジネス・プロセスでは、「場所」プロパティーは、実際には名前変更されています。本当のプロパティーは「場所タイプ」です。このことは、saprops.cfg でエレメンタリー・ビジネス・プロセスの定義を検索し、プロパティーが実際には「場所タイプ」と呼ばれているにもかかわらず、ラベル名が「場所」になっていることを確認することによって初めてわかります。

Property "Location Types" { Edit Listof "Location" Label "Locations" LENGTH 2000
HELP "Supporting Location Types (Matrix)" READONLY }

構文

```
Diagram Object.SetProperty Name, value
```

Diagram Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Diagram クラス

Name

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

Value

使用: 必須

データ型: 文字列

ダイアグラム・プロパティーの値

Show

目的

このメソッドは、Rational System Architect 画面のダイアグラムを開きます。

構文

```
Call oDiagram.Show
```

ダイアグラム・フィールド

ダイアグラム・フィールド・プロパティーには、ダイアグラム関する複数のプロパティーを含めることができます。一般には、ユーザーが直接入力できない、通常使用中に取得される情報が格納されます。オブジェクト・モデル内に、*DGMFLD* と呼ばれる列挙型があります。これは、パラメーターとして命令 **GetField(FieldID as *DGMFLD*)** および **SetField(FieldID as *DGMFLD*, Value as String)** に渡されます。これにより、VBA プログラマーは、低レベルのダイアグラム・フィールドの読み取りと更新の両方を行うことができます。

DGMFLD 定数	説明	データ型
DIAGFLD_BBORDER	「ページ設定」ウィンドウの「レポートの境界線」チェック・ボックスを切り替えます。これにより、ユーザーは、レポートの周囲に境界線を配置できます。	"0" = チェックしない "1" = チェックする
DIAGFLD_BDGMBORDER	「ページ設定」ウィンドウの「ダイアグラムの境界線」ドロップ・リストから、境界線を選択します。	"0" = フォームなし/境界線なし "1" = 単純な境界線
DIAGFLD_BDGMPDEFAULT	「ページ設定」をデフォルト設定として設定します。	"0" = オフ "1" = オン
DIAGFLD_BORDEROFFSET	「ページ設定」ウィンドウのオフセット値を設定します。このフィールドは、境界線をレポート・テキストから指定された値だけ離します。	String 100 分の 1 インチ単位の数値。
DIAGFLD_BPRESENTATIONMENU	描画ツールボックスのプレゼンテーション・メニューを自動的に含めるかどうかを切り	"0" = オフ "1" = オン

DGMFLD 定数	説明	データ型
	替えます。プレゼンテーション・メニューに含まれている追加のシンボルには、コンピューター、電話、人、ディスク、およびプリンターの基本的な形が含まれています。シンボルは、その他のブロック・シンボルと同じメソッドによってダイアグラムに描画され、適宜、名前が付けることができます。	
DIAGFLD_BREADONLY	ダイアグラム・オブジェクトを読み取り専用にします。	"0" = False "1" = True
DIAGFLD_BSHOWGRID	背面のグリッドの自動表示を切り替えます。	"0" = オフ "1" = オン
DIAGFLD_BSHOWLINESHADOW	すべてのライン・シンボルの周りのシャドーを自動的に含めるかどうかを切り替えます。	"0" = オフ "1" = オン
DIAGFLD_BSHOWNODESHADOW	すべてのノード・シンボルの周りのシャドーを自動的に含めるかどうかを切り替えます。	"0" = オフ "1" = オン
DIAGFLD_BSHOWPAGES	「ダイアグラム表示オプション (Diagram Display Options)」ウィンドウの「ページ」チェック・ボックスを切り替えます。チェック・マークが付いていると、ページの印刷域が点線で表示されるため、ダイアグラムが「実サ	"0" = チェックしない "1" = チェックする

DGMFLD 定数	説明	データ型
	イズ」モードで印刷される場合、ページ境界をプレビューできます。	
DIAGFLD_BSHOWRULER	PC のローカル設定に合わせてセンチメートル単位またはインチ単位で、x 軸と y 軸の罫線マーカーを表示するかどうかを切り替えます。オンに切り替えると、x と y の罫線が描画とともに保存されます。	"0" = オフ "1" = オン
DIAGFLD_BSHOWSCROLL	スクロール・バーの自動表示を切り替えます。現行の表示モードでダイアグラムが画面より大きい場合に、スクロール・バーによって、ダイアグラムの上下左右を表示できます。このオプションのデフォルトは、「チェックする」です。	"0" = チェックしない "1" = チェックする
DIAGFLD_BSHOWTEXTSHADOW	すべてのテキスト・シンボルの周りのシャドーを自動的に含めるかどうかを切り替えます。	"0" = オフ "1" = オン
DIAGFLD_BSNAPGRIDENT	グリッドの設定をより粗いグリッドまたはより細かいグリッドの設定に変更した後で、ダイアグラムのすべてのノード・シンボルを最も近いグリッド線 (通常は非表示) に合うように配置します。	"0" = オフ "1" = オン

DGMFLD 定数	説明	データ型
DIAGFLD_BSNAPGRIDLIN	グリッドの設定をより粗いグリッドまたはより細かいグリッドの設定に変更した後で、ダイアグラムのすべてのライン・シンボルを最も近いグリッド線 (通常は非表示) に合うように配置します。	"0" = オフ "1" = オン
DIAGFLD_CGRAPHNAME	ダイアグラムの名前。	String
DIAGFLD_CLEVELNUMBER	ダイアグラムのレベル番号	読取専用 String
DIAGFLD_DDDIAGRAM_DDIDENTITY	ダイアグラムのデータ・ディクショナリーの ID 番号	読取専用 Numeric
DIAGFLD_IDGMFORM	「ページ設定」ウィンドウの「ダイアグラムの境界線」ドロップダウン・リストから、境界線のフォームを選択します。	"0" = フォームなし/境界線なし "1" = IDEF0 作業中 "2" = IDEF0 発行 "3" = IDEF3 "4" = IDEF3 リリース済み "5" = SSADM フォーム
DIAGFLD_IGRAPHTYPE	ダイアグラムの SA タイプ	String ダイアグラムの内部定数番号。詳細については、SA ディレクトリーの diagrm.bas ファイルを参照してください。
DIAGFLD_PGRIDNUMENT	ノード・シンボルのグリッド	“[Vertical] [Horizontal]”

DGMFLD 定数	説明	データ型
	/inch。	**注:** DIAGFLD_PGRIDSIZEENT とともに使用する必要があります
DIAGFLD_PGRIDNUMLIN	ライン・シンボルのグリッド /inch。	“[Vertical] [Horizontal]” **注:** DIAGFLD_PGRIDSIZELIN とともに使用する必要があります
DIAGFLD_PGRIDSIZEENT	ノード・シンボルの inch/グリッド。	“[Vertical] [Horizontal]”100 分の 1 インチ単位。 **注:** DIAGFLD_PGRIDNUMENT とともに使用する必要があります
DIAGFLD_PGRIDSIZELIN	ライン・シンボルの inch/グリッド。	“[Vertical] [Horizontal]”100 分の 1 インチ単位。 **注:** DIAGFLD_PGRIDNUMLIN とともに使用する必要があります
DIAGFLD_PGRIDUNIT100	オブジェクトをグリッド内でドラッグできる距離を設定します。デフォルト値 = “100 100”	“[Vertical] [Horizontal]”100 分の 1 インチ単位。
DIAGFLD_PSHADOWDELTA	シャドーを配置するシンボルからの距離を設定します。デフォルト値 = “20 10”	“[Vertical] [Horizontal]”100 分の 1 インチ単位。
DIAGFLD_RGBSHADOWCOL	シャドーの色を設定します。	“[RGB Color]”

DGMFLD 定数	説明	データ型
OR		
DIAGFLD_RMARGIN	「ページ設定」ウィンドウの「余白」を設定します。	“[Left] [Top] [Right] [Bottom]”100 分の 1 インチ単位。
DIAGFLD_SAAUDITID	ダイアグラムの監査 ID	読取専用 String
DIAGFLD_SAIDENTITY	ダイアグラムのデータ・ディクショナリーの ID 番号。	読取専用 Numeric
DIAGFLD_SALOCK	ダイアグラムをロックします。	"0" = アンロック "1" = ロック
DIAGFLD_SAMAJORTYPE	メジャー・タイプ (「ダイアグラム」など)。	読取専用 String
DIAGFLD_SAMAJORTYPENUMBER	メジャー・タイプ番号 (「1」など)	読取専用 Numeric
DIAGFLD_SANAME	ダイアグラムの名前。	読取専用 String
DIAGFLD_SANUMBER	ダイアグラムのレベル番号 (IDEF0 のみ)	読取専用 Numeric
DIAGFLD_SATYPE	ダイアグラムのタイプ (「プロセス・チャート」、「エンティティー・リレーション」など)。.	読取専用 String
DIAGFLD_SATYPENUMBER	ダイアグラムの内部定数番号。	読取専用

DGMFLD 定数	説明	データ型
		Numeric
DIAGFLD_SAUPDATEDATE	最終更新日。	読取専用 日付フィールド
DIAGFLD_SAUPDATETIME	最終更新時刻。	読取専用 時間フィールド
DIAGFLD_USEDENTCOUNT	ダイアグラムのシンボルの数。	long (16 進数) 読取専用
DIAGFLD_WBORDERPENSTYLE	「ページ設定」ウィンドウの境界線のペン・スタイル。	“[SymPenStyle number]” すべての SA ペン・スタイルの詳細については、第 7 章を参照してください。
DIAGFLD_WORIENTATION	「ページ設定」ウィンドウの「ダイアグラムの印刷方向 (Diagram Printing Orientation)」。	"0" = プリンターのデフォルト "1" = 縦長 "2" = 横長 "3" = 最適化

ダイアグラムのメトリック

従来、メトリックはさまざまな Rational System Architect レポートにおいて、リストの作成、ルール・チェックの実行、および計算を行うために使用されてきました。現在では、Diagram クラスの GetMetric メソッドを呼び出すことによって、ユーザーが個別のメトリックを実行することが可能になりました。SA オブジェクト・ブラウザーの DIAGRAMMETRIC 列挙リストに、すべてのダイアグラムのメトリックの一覧があります。以下の表に、使用可能なすべてのダイアグラムのメトリックと、その説明を示します。

ダイアグラムのメトリック	説明
DIAGMETBALANCE	ダイアグラムの入力行および出力行と、その親プロセスの入力行および出力行とを比較します。一致しない入力行と出力行の名前およびシンボル・タイプを示したリストを作成します。(バランス親のヘルプ・ファイルを参照してください。)
DIAGMETCHARCOUNT	ダイアグラムの説明プロパティーの文字数を返します。
DIAGMETCURRENT	T/F ブール値フィールド。ダイアグラムが現在表示されている場合には、True を返します。
DIAGMETELELMENTLIST	ダイアグラムのすべてのシンボル定義について、最下位レベル要素のリストを作成します。拡張している関係がない場合、要素は最下位です。
DIAGMETINPUTLIST	ダイアグラムのすべてのシンボル定義について、入力として使用される最下位レベル要素のリストを作成します。拡張している関係がない場合、要素は最下位です。
DIAGMETLEVELNUMBER	ダイアグラムの階層の位置を記述する数値を文字列として返します (例: 5.3、5.3.1 など)。
DIAGMETLEVELNUMBERSORT	ダイアグラムの階層の位置を記述する数値を文字列として返します。各数値は、3 桁で構成されます

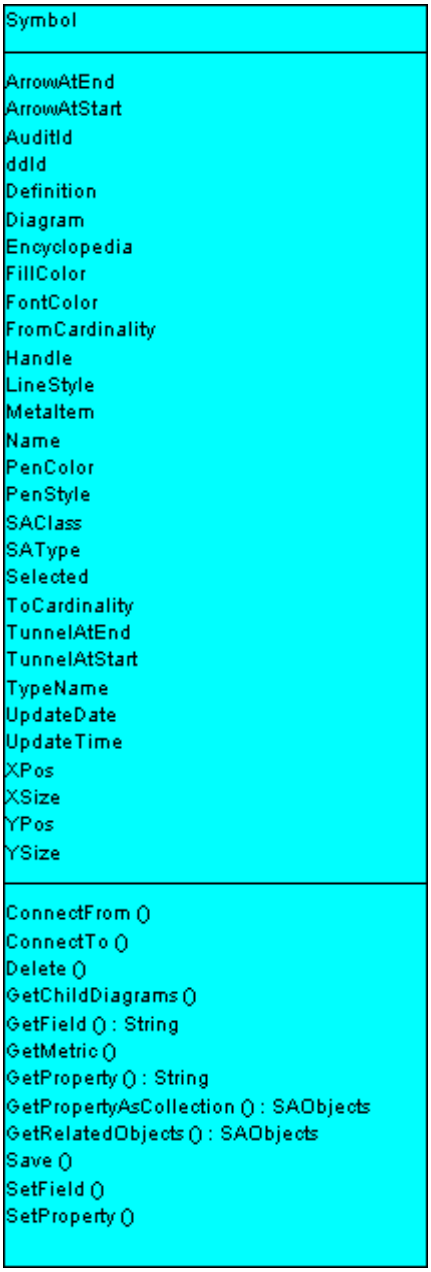
ダイアグラムのメトリック	説明
	(例: 003.005.002)。これにより、ユーザーは結果をソートしやすくなります。
DIAGMETLINECOUNT	ダイアグラムの説明プロパティーの行数を返します。
DIAGMETOUTPUTLIST	ダイアグラムのすべてのシンボル定義について、出力として使用される最下位レベル要素のリストを作成します。拡張している関係がない場合、要素は最下位です。
DIAGMETREFERENCE	ダイアグラムが他のオブジェクトによって参照されている場合に、True を返します。
DIAGMETRULES	ダイアグラムの標準のメソドロジー・ルールの違反を検索するルール・チェックを実行します。
DIAGMETSELECTED	ダイアグラムが現在開いており、選択されているシンボルがある場合に、True を返します。
DIAGMETTOP	ダイアグラムがシンボルの子である場合に、False を返します。ダイアグラムがシンボルから拡張されていない場合に、True を返します。
DIAGMETUNMARKEDLIST	入力としても出力としてもマークされていない (矢印のない)、ダイアグラムのすべてのライン・シンボルの定義によって使用される最下位レベル要素のリストを作成します。拡張している関係がない場合、要素は最下位です。
DIAGMETWORDCOUNT	ダイアグラムの説明プロパティーのワード数を返します。

7

シンボル・クラス

はじめに

Symbol クラスをその属性とメソッドとともに右の図に示します。



属性

ArrowAtEnd

目的
ライン・シンボルの結合プロパティーを設定します。ラインの終了側に矢印を作成します。

パラメーター
データ型: ブール値

例

```
oSymbol.ArrowAtEnd = True
```

ArrowAtStart

目的
ライン・シンボルの結合プロパティーを設定します。ラインの開始側に矢印を作成します。

パラメーター
データ型: ブール値

例

```
oSymbol.ArrowAtStart = True
```

AuditId

目的
All itemsRational System Architect に格納されるすべての項目のシンボルは、シンボルを作成したか、または最後に変更したユーザーの識別を使用してタグ付けされています。この識別は AuditId としてシンボルにタグ付けされています。

パラメーター
データ型: 文字列

読取専用

ddId

目的
Rational System Architect に格納されたすべてのシンボルは、データ・ディクショナリーのIDを使用して内部で一意に識別されます。このメソッドによって、シンボルの識別が返されます。

パラメーター
データ型: long

読取専用

定義

目的
シンボルの定義クラスへのアクセスを容易にします。

パラメーター
読取専用

ダイアグラム

目的
シンボルが作成されたダイアグラム・クラスへのアクセスを容易にします。

パラメーター
読取専用

エンサイクロペディア

目的
シンボルが属する Encyclopedia クラスへのアクセスを容易にします。

パラメーター
読取専用

FillColor

目的
このプロパティーを使用すると、シンボルを塗りつぶす色が返されます。色の値は OLE_COLOR 値です。

パラメーター
OLE_COLOR 値は、BGR (青、緑、赤) 値です。BGR 値を決定するには、以下の数式で青、緑、および赤を指定します (それぞれの値は 0 から 255 までです)。

BGR 値 = (青 * 65536) + (緑 * 256) + 赤

FontColor

目的
このプロパティーを使用すると、シンボルのフォントを塗りつぶす色が返されます。色の値は OLE_COLOR 値です。

パラメーター
OLE_COLOR 値は、BGR (青、緑、赤) 値です。BGR 値を決定するには、以下の数式で青、緑、および赤を指定します (それぞれの値は 0 から 255 までです)。

BGR 値 = (青 * 65536) + (緑 * 256) + 赤

FromCardinality

目的
このプロパティーを使用すると、エンティティー・リレーション・ダイアグラムの関係線、または物理データ・モデルの制約に関して、「初端」の基数を判別および設定できます。

設定される、または返される定数は、以下のとおりです。

定数	番号	意味
CARDINALITYZERO	0	ゼロ
CARDINALITYONLYONE	1	1 つのみ
CARDINALITYZEROONE	2	ゼロまたは 1 つ
CARDINALITYONEMULT	3	1 つまたは複数
CARDINALITYZEROONEMULT	4	ゼロ、1 つ、または複数
CARDINALITYMULT	5	複数
CARDINALITYUNKNOWN	6	マークなし
CARDINALITYNOTUSED	7	基数なし

ハンドル

目的
実行時にのみ使用可能なシンボルのメモリー・ハンドルです。このハンドルは一意ではなく、アクセス時に同じになることはほとんどありません。

パラメーター
データ型: long

読取専用

例

```
Dim Handle As Long
Handle = oSymbol.Handle
```

LineStyle

目的
ダイアグラムに描画される線は、複数のスタイルから 1 つ選択し、決定して設定できます。

設定される、または返される一般的な定数のいくつかを以下に示します。

定数	番号	意味
LSARC	4	楕円弧
LSAUTOSTROR	19	自動、直線、直角
LSTRAA	1	直線、角度自由
LSTROR	3	直線、直角 (手動)

MetaItem

目的
MetaItem の属性に容易にアクセスできるようにします。

パラメーター
読取専用

Name

目的
シンボル名

パラメーター
データ型: 文字列

読取専用

PenColor

目的
このプロパティーは、シンボルのペンの色を返すか設定します。色の値は OLE_COLOR 値です。

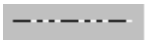
パラメーター
OLE_COLOR 値は、BGR (青、緑、赤) 値です。BGR 値を決定するには、以下の数式で青、緑、および赤を指定します (それぞれの値は 0 から 255 までです)。

BGR 値 = (青 * 65536) + (緑 * 256) + 赤

PenStyle

目的
このプロパティーを使用すると、シンボルのペンのスタイルを返すか設定できます。

PEN という接頭部を使用した一定範囲の定数が存在します。これらは、Rational System Architect の「書式」>「シンボル・スタイル」>「線と色」のオプションに対応します。

定数	番号	意味
PENDASH	1	
PENDASH2DOT	4	

PENDASHDOT	3	
PENDOT	2	
PENNULL	5	ペンのスタイルなし
PENSOLID1	16	
PENSOLID2	48	
PENSOLID3	64	
PENSOLID4	128	
PENSOLID4A	384	

SAClass

目的
シンボルのクラス・タイプ。メジャー・タイプ番号とも呼ばれます。

パラメーター
データ型: long

読取専用

注: シンボルの場合、常に「2」になります。

SAType

目的
シンボルの定数 (数値)。

パラメーター
データ型: long

読取専用

選択済み

目的
シンボルがダイアグラムで強調表示されるかどうかを示します。

パラメーター
データ型: ブール値

ToCardinality

目的
このプロパティーを使用すると、エンティティー・リレーション・ダイアグラムの関係線、または物理データ・モデルの制約に関して、終端の基数を判別または設定できます。

設定される、または返される定数は、以下のとおりです。

定数	番号	意味
CARDINALITYMULT	0	複数
CARDINALITYNOTUSED	1	基数なし
CARDINALITYONEMULT	2	1 つまたは複数
CARDINALITYONLYONE	3	1 つのみ
CARDINALITYUNKNOWN	4	マークなし
CARDINALITYZERO	5	ゼロ
CARDINALITYZEROONE	6	ゼロまたは 1 つ
CARDINALITYZEROONEMULT	7	ゼロ、1 つ、または複数

TunnelAtEnd

目的
このプロパティーを使用すると、IDEF0 機能ダイアグラムのライン・シンボルに関して、矢印の終わりの「トンネル」を判別および設定できます。

パラメーター
データ型: ブール値

TunnelAtStart

目的
このプロパティーを使用すると、IDEF0 機能ダイアグラムのライン・シンボルに対して、矢印の始めの「トンネル」を判別および設定できます。

パラメーター
データ型: ブール値

TypeName

目的
文字列としてのシンボルのタイプ名。例: 「Entity」

パラメーター
データ型: 文字列

読取専用

更新日付

目的
シンボルが最後に変更された日付。

パラメーター
データ型: 文字列

読取専用

UpdateTime

目的
シンボルが最後に変更された時刻。

パラメーター
データ型: 文字列

読取専用

Xpos

目的
シンボルの水平位置 (X 座標)。100 分の 1 インチ単位。

シンボルの右下隅の位置を示し、ダイアグラムの左を基点として測定されます。

パラメーター
データ型: long

Xsize

目的
シンボルの幅 (X 軸)。測定単位は 100 分の 1 インチです。

パラメーター
データ型: long

Ypos

目的
シンボルの垂直位置 (Y 座標)。100 分の 1 インチ単位。

シンボルの右下隅の位置を示し、ダイアグラムの上部を基点として測定されます。

パラメーター
データ型: long

Ysize

目的
シンボルの高さ (Y 軸)。測定単位は 100 分の 1 インチです。

パラメーター
データ型: long

メソッド

ConnectFrom

目的
線の「初」端でシンボルに接続するために使用されます。

構文
```
Symbol Object.ConnectFrom Line
```

Symbol Object

使用: 必須

データ型: オブジェクト

ライン・シンボルの接続元のインスタンス化された Symbol クラス。

ライン

使用: 必須

データ型: シンボル

インスタンス化されたライン・シンボル

例
```
Dim oFirstsymbol As Symbol, oSecondsymbol As Symbol,
  oLine As Symbol

Set oFirstsymbol = oDiagram.CreateSymbol("Customer",
  ETPROCESS)

Set oSecondsymbol = oDiagram.CreateSymbol("Order",
  ETPROCESS)

Set oLine = oDiagram.CreateSymbol("places",
  ETDATAFLOW)

Call oFirstsymbol.ConnectTo oLine

Call oSecondsymbol.ConnectFrom oLine
```

ConnectTo

目的
ラインの「終」端でシンボルに接続するために使用されます。

構文
```
Symbol Object.ConnectTo Line
```

Symbol Object

使用: 必須

データ型: オブジェクト

ライン・シンボルの接続先のインスタンス化された Symbol クラス。

ライン

使用: 必須

データ型: シンボル

インスタンス化されたライン・シンボル

例
```
Dim oFirstsymbol As Symbol, oSecondsymbol As Symbol,
  oLine As Symbol
Set oFirstsymbol = oDiagram.CreateSymbol("Customer",
  ETPROCESS)
Set oSecondsymbol = oDiagram.CreateSymbol("Order",
  ETPROCESS)
Set oLine = oDiagram.CreateSymbol("places",
  ETDATAFLOW)
Call oFirstsymbol.ConnectTo oLine
Call oSecondsymbol.ConnectFrom oLine
```

削除

目的
シンボルをダイアグラムから削除します。ダイアグラムが閉じられるときに、ダイアグラム上のすべてのシンボルは、ダイアグラム・オブジェクトが保存される場合を除き、自動的に削除されます。

GetChildDiagrams

目的
このメソッドは、シンボルに子が接続されている場合、そのシンボルについて、ダイアグラムの SAObjects コレクションを取得します。

SAObjects コレクションは、コレクションの Complete フラグが True になるまでは、完全に取り込まれません。GetAllSymbols は、ReadAll メソッドまたは IsMoreThan メソッドのいずれかとともに使用する必要があります。

例
```
Dim oSymbol as Symbol, oCollectionofSymbols As
  SAObjects
Set oCollectionofSymbols = oSymbol.GetChildDiagrams
Call oCollectionofSymbols.ReadAll
```

GetField

目的
これらは、「フォント・タイプ」、「フォントの高さ」など、シンボルの特性です。これらの中には、「フォント・タイプ」のように設定できるものと、「AuditId」のように設定できないものとがあります。

構文
```
Symbol Object. GetField FieldID
```

```
Symbol Object
```

使用: 必須

データ型: オブジェクト

インスタンス化された Symbol クラス

`FieldID`

使用: 必須

データ型: SYMBOLFIELDS

シンボル・フィールド。すべてのシンボル・フィールドの詳細については、以下を参照してください。

Get Metric

目的

シンボルに関する特定のリスト、計算、および内部機能の部分を呼び出します。

構文

```
Symbol Object.GetMetric Metric[, FieldType[, NbrChars[,
  NbrDec]]]
```

`Symbol Object`

使用: 必須

データ型: オブジェクト

インスタンス化された Symbol クラス

`計量`

使用: 必須

データ型: SYMBOLMETRIC

シンボルのメトリック。すべてのシンボルのメトリックの詳細については、以下を参照してください。

`FieldType`

使用: オプション

データ型: FLDTYPE

フィールド・タイプ。Rational System Architect のフィールド・タイプの詳細については、第 17 章を参照してください。

`NbrChars`

使用: オプション

データ型: long

フィールド・タイプが入力されている場合は、このパラメーターによって、小数点の前に戻す文字の数が SA に伝えられます。

`NbrDec`

使用: オプション

データ型: long

フィールド・タイプが入力されている場合は、このパラメーターによって、小数点の後に戻す数字の個数が SA に伝えられます。

GetProperty

目的

指定のシンボル・プロパティーに関してシンボル・プロパティーのコンテンツを返します。すべてのプロパティー名の詳細については、usrprops.txt および saprops.cfg を参照してください。

パラメーター

プロパティーの実名が、定義ダイアログに表示されるものと異なる場合があります。例えば、IDEF3 プロセス・フロー/ OV-6a ダイアグラムのジャンクションで、「Logic」プロパティーの名前が変更されていて、実際のプロパティーが「Junction Logic」になっている場合などがあります。このことは、saprops.cfg でジャンクションの定義を検索し、プロパティーが実際には「Junction Logic」と呼ばれているが、「Logic」とラベル付けされていることを確認した場合にのみわかります。

PROPERTY "Junction Logic" { EDIT Text ListOnly LIST "Junction Logic" LENGTH 3 DEFAULT "And" LABEL "Logic" }

構文

```
Symbol Object.GetProperty Name

Symbol Object
```

使用: 必須

データ型: オブジェクト

インスタンス化された Symbol クラス

`Name`

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

GetPropertyAsCollection

目的

プロパティーの中には、他のプロパティーとの関係を定義するものがあります。例えば、エンティティーは「モデル」プロパティーを経由して、モデルを参照する場合があります。このメソッドは、OneOf および ListOf のダイアグラムまたは定義のコレクションを返します。OneOf および ListOf のプロパティー・タイプの詳細については、第 14 章を参照してください。

パラメーター

データ型: OfCollection

プロパティーの実名が、定義ダイアログに表示されるものと異なる場合があります。例えば、IDEF3 プロセス・フロー/ OV-6a ダイアグラムのジャンクションで、「Logic」プロパティーの名前が変更されていて、実際のプロパティーが「Junction Logic」になっている場合などがあります。このことは、saprops.cfg でジャンクションの定義を検索し、プロパティーが実際には「Junction Logic」と呼ばれているが、「Logic」とラベル付けされていることを確認した場合にのみわかります。

PROPERTY "Junction Logic" { EDIT Text ListOnly LIST "Junction Logic" LENGTH 3 DEFAULT "And" LABEL "Logic" }

構文

```
Symbol Object.GetPropertyAsCollection(PropName)
```

`Symbol Object`

使用: 必須

データ型: オブジェクト

インスタンス化された Symbol クラス

`PropName`

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

例

```
Dim Models As OfCollection
    Set Models =
  SASym.GetPropertyAsCollection("Model")
```

GetRelatedObjects

目的

このメソッドは、関連オブジェクトの SAObjects コレクションを現在のシンボル・オブジェクトに返します。

構文

```
Symbol Object.GetRelatedObjects(RelType)
```

`Symbol Object`

使用: 必須

データ型: オブジェクト

インスタンス化された Symbol クラス

`RelType`

使用: 必須

データ型: RELATETYPE

SA 関係。すべての関係の詳細については、第 16 章を参照してください。

例

```
Dim oCollectionOfRelatedItems As SAObjects
Set oCollectionOfRelatedItems =
  oSymbol.GetRelatedObjects(RELCONNEND)
oCollectionOfRelatedItems.ReadAll
```

保存

目的

ダイアグラムのシンボルを作成後に保存するには、**save** メソッドを呼び出します。

例

```
oSymbol.Save
```

SetField

目的

シンボルにフィールド値を設定します。フィールドおよびその値の 2 つの引数を必要とします。

構文

```
Symbol Object. SetField FieldID, value
```

`Symbol Object`

使用: 必須

データ型: オブジェクト

インスタンス化された Symbol クラス

`FieldID`

使用: 必須

データ型: SYMBOLFIELD

シンボル・フィールド。すべてのシンボル・フィールドの詳細については、以下を参照してください。

value

使用: 必須

データ型: 文字列

シンボル・フィールドの値

SetProperty

目的

プロパティー名とその値がわかると、シンボルのプロパティーを設定できます。すべてのプロパティー名の詳細については、usrprops.txt および saprops.cfg を参照してください。

パラメーター

プロパティーの実名が、定義ダイアログに表示されるものと異なる場合があります。例えば、IDEF3 プロセス・フロー/ OV-6a ダイアグラムのジャンクションで、「Logic」プロパティーの名前が変更されていて、実際のプロパティーが「Junction Logic」になっている場合などがあります。このことは、saprops.cfg でジャンクションの定義を検索し、プロパティーが実際には「Junction Logic」と呼ばれているが、「Logic」とラベル付けされていることを確認した場合にのみわかります。

PROPERTY "Junction Logic" { EDIT Text ListOnly LIST "Junction Logic" LENGTH 3 DEFAULT "And" LABEL "Logic" }

構文

```
Symbol Object.SetProperty Name, value
```

Symbol Object

使用: 必須

データ型: オブジェクト

インスタンス化された Symbol クラス

Name

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

値

使用: 必須

データ型: 文字列

シンボル・プロパティーの値

シンボル・フィールド

シンボル・フィールド・プロパティーには、シンボルについて保持されるいくつかのプロパティーを含めることができます。一般には、ユーザーが直接入力できない、通常使用中に取得される情報が格納されます。例えば、「更新時間」、「監査」、「位置」、「サイズ」、「タイプ名」が含まれます。

オブジェクト・モデル内に、*SYMBOLFIELD* と呼ばれる列挙型があります。これは、パラメーターとして **GetField(FieldID as *SYMBOLFIELD*)** および **SetField(FieldID as *SYMBOLFIELD*, Value as String)** 操作に渡されます。これにより、VBA プログラマーは、低レベルのシンボル・フィールドの読み取りと更新の両方を行うことができます。

SYMBOLFIELD の定数	説明	データ型	
SYMFLD_AUDITID	シンボルの監査 ID	文字列 読取専用	
SYMFLD_BARRANGMENT	シンボル・ツリーの配置	“0” = 子を水平に配置 “1” = 子を垂直に配置 “2” = 子をブロック型に配置	
SYMFLD_BOTHERSYMBOLOGY	シンボルの代替フォームを表示 (例: シンボルのステレオタイプ)	ブール	
SYMFLD_CBGCOLOR	文字画面ダイアグラムのシンボルの背景色。	“0” = 黒 “1” = 青 “2” = 緑 “3” = シアン	“4” = 赤 “5” = マゼンタ “6” = 茶/黄色 “7” = 白
SYMFLD_CFGCOLOR	文字画面ダイアグラムのシン	“0” = 黒	“4” = 赤

SYMBOLFIELD の定数	説明	データ型	
	ボルの前景色。	“1” = 青 “2” = 緑 “3” = シアン	“5” = マゼンタ “6” = 茶/黄色 “7” = 白
SYMFLD_COCCOFFSET	文字画面ダイアグラム上での、一部の入力フィールドの値の各反復の間のスペーシングを決定します。	数値	
SYMFLD_COCCURS	文字画面ダイアグラム上での、一部の入力フィールドの値の複数の反復を許可します。	数値	
SYMFLD_COMMENT	シンボルのグラフィック・コメントのプロパティーを設定します。	文字列	
SYMFLD_CPROMPT	文字画面ダイアグラム上のシンボルのコバルト色のプロンプト文字	1 ビットの文字列	
SYMFLD_CUNCLECOUNT	フロー線によって親シンボルに直接接続しているシンボルの数を返します。	16 進数 読取専用	
SYMFLD_DDCOMMENT	シンボルのコメント・シンボルのデータ・ディクショナリーの ID 番号。	数値 読取専用	
SYMFLD_DDIDENTITY	シンボルのデータ・ディクショナリーの ID 番号。	数値 読取専用	

SYMBOLFIELD の定数	説明	データ型
SYMFLD_DESCLOC	シンボルのグラフィック・コメントの場所。	“XPos YPos” 数値
SYMFLD_DESCSIZE	シンボルのグラフィック・コメントのサイズ。	“XSize YSize” 数値
SYMFLD_DWSTYLE	グラフィック画面シンボルに関してユーザーがオプションとして選択した内容を、グラフィック画面ダイアグラムに反映します。	16 進数
SYMFLD_ENDLOC	ライン・シンボルが終了する場所。	“XPos YPos” 数値
SYMFLD_ERROR1	Rational System Architect によって検出された最初のエラー。	エラー番号
SYMFLD_ERROR2	Rational System Architect によって検出された 2 番目のエラー。	エラー番号
SYMFLD_FONTFLAGS	シンボルのフォントの太字、イタリック、下線、または取り消し線を切り替えます。	16 進数 “0x0002” = 太字 “0x0005” = イタリック “0x000B” = 下線 “0x0010” = 取り消し線
SYMFLD_FONTHEIGHT	シンボルのフォント・サイズ	16 進数
SYMFLD_FONTNAME	シンボルのフォント名 (例: Arial、Times New Roman な	文字列

SYMBOLFIELD の定数	説明	データ型
	ど)	
SYMFLD_FREXARCCHAR	ライン・シンボルの初端に排他的な弧を挿入します。	ブール
SYMFLD_FROMCARDINALITY	FromCardinality の名前を返します (例: 1 つのみ、1 つまたは複数など)。 FromCardinality 値の詳細については、Symbol クラス属性の FromCardinality 属性を参照してください。	文字列 読取専用
SYMFLD_FROMCARDNUMBER	FromCardinality 定数を返します。FromCardinality 値の詳細については、Symbol クラス属性の FromCardinality 属性を参照してください。	数値 読取専用
SYMFLD_FROMCONNECTCOMPASSPOINT	「初」端に接続している ICOM 矢印の開始点の東西南北を返します。	文字列 読取専用
SYMFLD_HASFROMARROW	線の「初」端に矢印がある場合は True を返します。	ブール 読取専用
SYMFLD_HASFROMTUNNEL	ICOM の矢印が「初」端でトンネルされている場合は True を返します。	ブール 読取専用
SYMFLD_HASTOARROW	線の「終」端に矢印がある場合は True を返します。	ブール 読取専用
ICOM の矢印が「終」端でトンネルされている場合は True を返します。	ICOM の矢印が「終」端でトンネルされている場合は True を返します。	ブール 読取専用

SYMBOLFIELD の定数	説明	データ型
SYMFLD_LINESTYLE	シンボル線種	4 バイトの 16 進数
SYMFLD_LOC	ダイアグラムのシンボルの場所。	“XPos YPos” 数値
SYMFLD_NAME	シンボル名	文字列
SYMFLD_NAMECRLF	復帰および改行があるシンボルの名前。名前フィールドに、ユーザーはテキストを最大 5 行まで入力できます。	文字列
SYMFLD_NAMECRLF1	名前が連続したテキストの文字列として表示される場合 (例: JimJaneTomLouRon)、テキストの何文字目から 2 行目が始まるか (例: 4)。	数値
SYMFLD_NAMECRLF2	名前が連続したテキストの文字列として表示される場合 (例: JimJaneTomLouRon)、テキストの何文字目から 3 行目が始まるか (例: 8)。	数値
SYMFLD_NAMECRLF3	名前が連続したテキストの文字列として表示される場合 (例: JimJaneTomLouRon)、テキストの何文字目から 4 行目が始まるか (例: 11)。	数値
SYMFLD_NAMECRLF4	名前が連続したテキストの文字列として表示される場合 (例: JimJaneTomLouRon)、テキストの何文字目から 5 行目が始まるか (例: 14)。	数値
SYMFLD_NAMELOC	「シンボル名」フィールドの	“XPos YPos”

SYMBOLFIELD の定数	説明	データ型
	場所。	数値
SYMFLD_NAMESIZE	「シンボル名」フィールドのサイズ	“XSize YSize” 数値
SYMFLD_ORDER	関連端の順序付け	“O” = 順序付けなし “1” = 順序付けあり “2” = ソート済み
SYMFLD_PENSTYLE	シンボルのペン・スタイルおよび幅	4 バイトの 16 進数
SYMFLD_ROTATION	構造チャートでフラグ・シンボルを回転します。	0 から 31 までの数値を指定することにより、フラグ・シンボルは右回りに回転します。例えば、以下のとおりです。 “0” = 南 “8” = 西 “16” = 北 “24” = 東
SYMFLD_SAMAJORTYPE	メジャー・タイプ (例: シンボル)	文字列 読取専用
SYMFLD_SAMAJORTYPENUMBER	メジャー・タイプ番号 (「2」など)	数値 読取専用
SYMFLD_SEQNUM	エンティティー・シンボルのエンティティー番号。	数値

SYMBOLFIELD の定数	説明	データ型
SYMFLD_SIZE	シンボル・サイズ	“XSize YSize” 数値
SYMFLD_STARTLOC	ライン・シンボルが開始する場所。	“XPos YPos” 数値
SYMFLD_STYLEFLAGS	シンボルの色を有効にします	16 進数 “0x0001” = ペンの色 “0x0002” = 塗りつぶす色 “0x0004” = フォントの色
SYMFLD_SUPERSUB	シンボルのスーパーとサブの関係値を設定します。	“0” = いずれでもない “1” = スーパー “2” = サブ
SYMFLD_TEXTFLAGS	シンボルの「テキスト」プロパティー	16 進数
SYMFLD_TOCARDINALITY	ToCardinality の名前を返します (例: 1 つのみ、1 つまたは複数など)。ToCardinality 値の詳細については、Symbol クラス属性の ToCardinality 属性を参照してください。	文字列 読取専用
SYMFLD_TOCARDNUMBER	ToCardinality 定数を返します。ToCardinality 値の詳細については、Symbol クラス属性の ToCardinality 属性を参照してください。	数値 読取専用
SYMFLD_TOCONNECTCOM	「終」端に接続している ICOM 矢印の開始点の東西南	文字列

SYMBOLFIELD の定数	説明	データ型
PASSPOINT	北を返します。	読取専用
SYMFLD_TOEXARCCHAR	ライン・シンボルの終端に排他的な弧を挿入します。	ブール
SYMFLD_TYPE	シンボル SA タイプ	SAType の内部定数。
SYMFLD_TYPENAME	シンボル SA タイプ名 (例: エンティティー、ICOM 矢印など)。	文字列 読取専用
SYMFLD_U_S1_WPICTYPE	ピクチャー・タイプ (ダイアグラムに追加されたグラフィック・ファイル)。	文字列 読取専用
SYMFLD_U_S1_ZPPICFILE	ピクチャーの表示に使用されているファイルのパス名。	文字列 読取専用
SYMFLD_UPDATEDATE	最終更新日	日付フィールド 読取専用
SYMFLD_UPDATEDATEINTL	最終更新日 (国際書式)	日付フィールド 読取専用
SYMFLD_UPDATETIMEINTL	最終更新時刻 (国際書式)	時間フィールド 読取専用
SYMFLD_XPENTITY	シンボルに与えられている内部番号。	数値 読取専用
SYMFLD_XPGROUP	シンボルの親に与えられている内部番号。	数値 読取専用
SYMFLD-XPLINK	選択したシンボルがリンクさ	数値

SYMBOLFIELD の定数	説明	データ型
	れているシンボルの内部番号 (例: IDEF3 プロセス・フロー・ダイアグラムの「振る舞いの単位」にリンクされている参照先)。	読取専用
SYMFLD_XPSIBLING	次の連続した兄弟の内部番号。	数値 読取専用
SYMFLD_XPSUBORDINATE	最初の子シンボルに与えられている内部番号	数値 読取専用
SYMFLD_ZPDESC	シンボルのグラフィック・コメントを設定します。	文字列
SYMFLD-ZPSSADMSTR	不明。	

シンボルのメトリック

従来、メトリックはさまざまな Rational System Architect レポートにおいて、リストの作成、ルール・チェックの実行、および計算を行うために使用されてきました。現在では、Symbol クラスの GetMetric メソッドを呼び出すことによって、ユーザーが個別のメトリックを実行することが可能になりました。SA オブジェクト・ブラウザーの SYMBOLMETRIC 列挙リストには、すべてのシンボルのメトリックの一覧があります。以下の表に、使用可能なすべてのシンボルのメトリックと、その説明を示します。

シンボルのメトリック	説明
SYMMETANNOTATION	シンボルが注釈シンボル (文書ブロック、テキスト・ボックス、長方形、ページ・コネクター) である場合に、True を返します。
SYMMETBALANCE	シンボルの入力行および出力行と、その子プロセスの入力行および出力行とを比較します。一致しない入力行と出力行の名前およびシンボル・タイプを示したリストを作成します。(バランス子のヘルプ・ファイルを参照してください。) シンボルがデータ・ストア、AND コネクター、または XOR コネクターである場合に、それらのシンボルの、定義された要素と構造を比較します。シンボル内の定義されていない要素のリストを作成し、着信および発信データ・フローに、定義された要素があるかどうかを示します。(「バランス 水平」のヘルプ・ファイルを参照してください。)
SYMMETBALANCEMSPEC	シンボル定義のミニスペックのバランスを取ります。データ・フロー・ダイアグラムのプロセスおよび構造チャート・ダイアグラムのモジュールに使用されます。詳しくは、Rational System Architect のヘルプ・ファイルを参照し、キーワード「ミニスペック」を使用してください。
SYMMETBOTTOM	派生 T/F ブール値フィールドが最下位レベルになりま

シンボルのメトリック	説明
	す。ダイアグラムまで拡張していない場合、シンボルの値は True です。
SYMMETCHARCOUNT	シンボルの説明プロパティーの文字数を返します。
SYMMETCONNECTOR	シンボルがコネクター・シンボル (AND コネクター、XOR コネクター、または ICOM 矢印ジョイン) の場合に True を返します。
SYMMETCURRENT	T/F ブール値フィールド。シンボルが現在表示されているダイアグラムに存在する場合は、True を返します。
SYMMETELEMENTLIST	シンボル定義について、最下位レベル要素のリストを作成します。拡張している関係がない場合、要素は最下位です。
SYMMETEXPRESSION	シンボル定義の式で使用されている、間違った式構文、または未定義のデータ要素やデータ構造のリストを作成します。
SYMMETICOMDEST	ICOM 矢印の宛先ロール (入力、制御、機構、または境界) を文字列として返します。
SYMMETICOMSOURCE	ICOM 矢印のソース・ロール (呼び出し、出力、または境界) を文字列として返します。
SYMMETINPUTLIST	シンボル定義の入力として使用される、最下位レベル要素のリストを作成します。拡張している関係がない場合、要素は最下位です。
SYMMETISFOREIGNKEY	シンボル定義が外部キーの場合に True を返します。
SYMMETKEYCOMPNBR	シンボルの主キーのコンポーネント番号を返します。コンポーネント番号 (@1、@2 など) は、ブラウザー詳細内のシンボル定義の属性リストを展開して表示することもできます。
SYMMETLEVELNUMBER	シンボルの階層の位置を記述する数値を文字列として

シンボルのメトリック	説明
	返します (例: 5.3、5.3.1 など)。
SYMMETLEVELNUMBERSORT	シンボルの階層の位置を記述する数値を文字列として返します。各数値は、3 桁で構成されます (例: 003.005.002)。これにより、ユーザーは結果をソートしやすくなります。
SYMMETLINECOUNT	シンボル定義の説明プロパティーの行数を返します。
SYMMETNORMALIZE1	シンボル定義が第 1 正規形になっているかどうかのチェックを実行します。繰り返しグループが含まれていないエンティティーは、第 1 正規形です。
SYMMETNORMALIZE23	シンボル定義が第 2 正規形および第 3 正規形になっているかどうかのチェックを実行します。エンティティーが第 1 正規形に従っていて、それぞれの非キー属性の機能が完全に主キーに従属している場合、そのエンティティーは第 2 正規形になっています。エンティティーが第 2 正規形に従っていて、各非キー属性が主キーのみに従属している場合、そのエンティティーは第 3 正規形になっています。
SYMMETOUTPUTLIST	シンボル定義の出力として使用される、最下位レベル要素のリストを作成します。拡張している関係がない場合、要素は最下位です。
SYMMETPARENTSLASHDATA	属性のキー設定元の情報が含まれているシンボル定義の外部キー・スラッシュ・データを返します。外部キー・スラッシュ・データは、ブラウザー詳細内のシンボル定義の属性リストを展開して表示することもできます。スラッシュ・データは、次の例のような形式で表示されます。 Row_Number / FKFROM “Stock_Location.Row_Number(stores)” /
SYMMETREFERENCE	シンボル定義が他のオブジェクトによって参照されている場合に、True を返します。

シンボルのメトリック	説明
SYMMETRULES	シンボルの標準のメソドロジー・ルールの違反を検索するルール・チェックを実行します。
SYMMETSELECTED	現在開いているダイアグラム上のシンボルが強調表示されている場合に True を返します。
SYMMETSEQINPARENTSLIST	親オブジェクトのプロパティー・セットを調べ、子オブジェクトのリストを検索します。そのリストでシンボル定義が表示される回数を返します。
SYMMETTOP	選択されたシンボルのあるダイアグラムが他のシンボルの子である場合に、False を返します。ダイアグラムが他のシンボルから拡張されたものでない場合に、True を返します。
SYMMETUNMARKEDLIST	入力または出力のいずれとしてもマークが付けられていない (矢印のない)、すべてのライン・シンボルの定義によって使用される最下位レベル要素のリストを作成します。拡張している関係がない場合、要素は最下位です。
SYMMETUPDATEUSES	データ・ディクショナリーの関係テーブル (RELATN.DBF) を更新します。戻り値はありません。
SYMMETWORDCOUNT	シンボル定義の説明プロパティーの単語数を返します。

8

Definition クラス

はじめに

Definition クラスの属性およびメソッドを、次に示します。

属性

AuditID

目的
Rational System Architect に格納されたすべての項目定義には、その定義の作成者または最終更新者の ID が AuditId としてタグ付けされています。

パラメーター
データ型: 文字列

読取専用

CheckedOut

目的
True に設定されている場合は、定義をチェックアウトした AuditID を除くすべてのユーザーに対して、定義は読み取り専用となります。

パラメーター
データ型: ブール値

ddID

目的
Rational System Architect に格納されたすべての定義は、データ・ディクショナリー ID を使用して、内部で一意に識別されます。

パラメーター
データ型: long

読取専用

エンサイクロペディア

目的
Encyclopedia クラスの属性およびメソッドに容易にアクセスできるようにします。

パラメーター
読取専用

Frozen

目的
この属性を設定するユーザーには、フリーズ特権が必要です。True に設定されている場合は、定義をフリーズした AuditID を含むすべてのユーザーに対して、定義は読み取り専用となります。

パラメーター
データ型: ブール値

ハンドル

目的
実行時にのみ使用できる、定義のメモリー・ハンドルです。このハンドルは一意ではなく、アクセス時に同じになることはほとんどありません。

パラメーター
データ型: long

読取専用

例

```
Dim Handle As Long
Handle = oDefinition.Handle
```

Locked

目的

定義がロックされているかどうか (ユーザーが使用中かどうか) を示す「True」または「False」値を戻します。有効な定義オブジェクトが必要です。

パラメーター
データ型: ブール値

読取専用

MetaItem

目的
MetaItem の属性に容易にアクセスできるようにします。

パラメーター
読取専用

Name

目的
指定された定義オブジェクトの名前。

パラメーター
データ型: 文字列

読取専用

ReadOnly

目的
定義が読み取り専用かどうか

パラメーター
データ型: ブール値

読取専用

SAClass

目的
定義のクラス・タイプです。メジャー・タイプ番号とも呼ばれます。

パラメーター
データ型: long

読取専用

注: 定義の場合は、常に「3」を戻します。

SAType

目的
定義を表す整数定数です。Rational System Architect のすべての定義には、一意の定数 ID が設定されている必要があります。

パラメーター
データ型: long

読取専用

TypeName

目的
文字列で表される定義のタイプ (Process など)

パラメーター
データ型: 文字列

読取専用

更新日付

目的
定義の最終更新日。

パラメーター
データ型: 文字列

読取専用

UpdateTime

目的
定義の最終更新時刻。

パラメーター
データ型: 文字列

読取専用

xml

目的
定義の XML 文字列。GetXML メソッドで操作します。

パラメーター
データ型: 文字列

読取専用

メソッド

削除

目的
定義オブジェクトで指定された定義を削除します。

例

```
Call oDefinition.Delete
```

GetField

目的
「Type Number」、「Undefined Flag」、「Major Type Name」などの定義の特性です。

構文

```
Definition Object. GetField FieldID
```

`Definition Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Definition クラス

`FieldID`

使用: 必須

データ型: DEFFLD

定義フィールド。定義フィールドの詳細については、以下を参照してください。

Get Metric

目的
定義に関連する特定のリスト、計算、および内部機能を呼び出します。

構文

```
Definition Object.GetMetric Metric[, FieldType[,
  NbrChars[, NbrDec]]]
```

`Definition Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Definition クラス

`計量`

使用: 必須

データ型: DEFINITIONMETRIC

定義メトリック。すべての定義メトリックの詳細については、以下を参照してください。

`FieldType`

使用: オプション

データ型: FLDTYPE

フィールド・タイプ。Rational System Architect のフィールド・タイプの詳細については、第 17 章を参照してください。

`NbrChars`

使用: オプション

データ型: long

フィールド・タイプが入力されている場合は、このパラメーターによって、小数点の前に戻す文字の数が SA に伝えられます。

`NbrDec`

使用: オプション

データ型: long

フィールド・タイプが入力されている場合は、このパラメーターによって、小数点の後に戻す数字の個数が SA に伝えられます。

GetProperty

目的

指定された定義プロパティーの内容を戻します。

パラメーター

プロパティーの実名が定義ダイアログに表示される名前と異なることがあります。例えば、サービス・タイム・プロファイルの「Time Units」プロパティーの名前が変更されていて、実際のプロパティーが「Duration Time Units」になっている場合などがあります。このことは、saprops.cfg でサービス・タイム・プロファイルの定義を調べて、プロパティー名が実際は「Duration Time Units」であるにもかかわらず、ラベル名が「Time Units」になっていることを確認した場合にのみわかります。

```
PROPERTY "Duration Time Units" { EDIT Text LISTONLY List "Time Units" LABEL "Time
        Units" DEFAULT "Hour"  LENGTH 20 READONLY }
```

構文

```
Definition Object.GetProperty Name
```

```
Definition Object
```

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Definition クラス

```
Name
```

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

GetPropertyAsCollection

目的

プロパティーの中には、他のプロパティーとの関係を定義するものがあります。例えば、Entity Definition は「Description」プロパティーを介して属性を参照します。このメソッドは、OneOf および ListOf のダイアグラムまたは定義のコレクションを返します。OneOf および ListOf のプロパティー・タイプの詳細については、第 14 章を参照してください。

パラメーター

データ型: OfCollection

プロパティーの実名が定義ダイアログに表示される名前と異なることがあります。例えば、サービス・タイム・プロファイルの「Time Units」プロパティーの名前が変更されていて、実際のプロパティーが「Duration Time Units」になっている場合

などがあります。このことは、saprops.cfg でサービス・タイム・プロファイルの定義を調べて、プロパティー名が実際は「Duration Time Units」であるにもかかわらず、ラベル名が「Time Units」になっていることを確認した場合にのみわかります。

```
PROPERTY "Duration Time Units" { EDIT Text LISTONLY List "Time Units" LABEL "Time
      Units" DEFAULT "Hour"  LENGTH 20 READONLY }
```

構文

```
Definition Object.GetPropertyAsCollection(PropName)
```

`Definition Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Definition クラス

`PropName`

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

例

```
Dim i As Long, DiagId As Long
i = 0
Do While sa.Encyclopedia.GetFilteredDefinitions("",
  DFXACTIVITY).IsMoreThan(i)
    i = i + 1
    Dim AttribColl As OfCollection
    Set SADef =
  sa.Encyclopedia.GetFilteredDefinitions("",
  DFXACTIVITY).Item(i)
    Set AttribColl =
  SADef.GetPropertyAsCollection("Description")
Loop
```

GetRelatedObjects

目的

関連オブジェクトの SAObjects コレクションを現在の定義オブジェクトに戻します。

構文

```
Definition Object.GetRelatedObjects(RelType)
```

`Definition Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Definition クラス

`RelType`

使用: 必須

データ型: RELATETYPE

SA 関係。すべての関係の詳細については、第 16 章を参照してください。

GetXML

目的

定義の XML 文字列を有効な .xml ファイル名にエクスポートします。

構文

```
Definition Object.GetXML(strXMLTextOut)
```

`Definition Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Definition クラス

`StrXMLTextOut`

使用: 必須

データ型: 文字列

SA が定義の XML 文字列をエクスポートする、有効な .xml ファイル。

保存

目的

定義のインスタンスを保存する場合に使用します。

例

```
Call oDefinition.Save
```

SetField

目的

定義フィールドに、指定された値を設定します。

構文

```
Definition Object.SetField FieldID, value
```

`Definition Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Definition クラス

`FieldID`

使用: 必須

データ型: DGMFLD

定義フィールド。定義フィールドの詳細については、以下を参照してください。

値

使用: 必須

データ型: 文字列

定義フィールドの値

SetProperty

目的

定義プロパティー値を設定するには、プロパティー名を最初の引数に指定して、設定する値を 2 番目の引数に指定する必要があります。プロパティー名は saprops.cfg および usrprops.txt ファイルで調べることができます。

パラメーター

プロパティーの実名が定義ダイアログに表示される名前と異なることがあります。例えば、サービス・タイム・プロファイルの「Time Units」プロパティーの名前が変更されていて、実際のプロパティーが「Duration Time Units」になっている場合などがあります。このことは、saprops.cfg でサービス・タイム・プロファイルの定義を調べて、プロパティー名が実際は「Duration Time Units」であるにもかかわらず、ラベル名が「Time Units」になっていることを確認した場合にのみわかります。

```
PROPERTY "Duration Time Units" { EDIT Text LISTONLY List "Time Units" LABEL "Time
      Units" DEFAULT "Hour"  LENGTH 20 READONLY }
```

構文

```
Definition Object.SetProperty Name, value
```

```
Definition Object
```

使用: 必須

データ型: オブジェクト

インスタンス化された任意の Definition クラス

```
Name
```

使用: 必須

データ型: 文字列

saprops.cfg に格納されているプロパティー名

値

使用: 必須

データ型: 文字列

定義プロパティーの値

定義フィールド

定義フィールド・プロパティーには、定義に関して保持されるいくつかのプロパティーを含めることができます。一般には、ユーザーが直接入力できない、通常使用中に取得される情報が格納されます。オブジェクト・モデル内には、*DEFFLD* という列挙型があります。この列挙型は、**GetField(FieldID as *DEFFLD*)** および **SetField(FieldID as *DEFFLD*, Value as String)** 操作にパラメーターとして渡されます。これを使用すると、VBA プログラマーは低レベル定義フィールドの読み取りと更新の両方を実行できるようになります。

DEFFLD 定数	説明	データ型
DEFNFLD_SAAUDITID	定義の監査 ID。	読取専用
DEFNFLD_SAIDENTITY	定義のデータ・ディクショナリー ID。	読取専用
DEFNFLD_SAIDENTITY4	定義のデータ・ディクショナリー ID。	4 バイトの 2 進数 読取専用
DEFNFLD_SAISUNDEFINED	定義が未定義の場合は「T」、定義が定義されている場合は「F」を戻します。	読取専用
DEFNFLD_SALOCK	定義をロックします。	「T」 = ロック状態 「F」 = ロック解除状態
DEFNFLD_SAMAJORTYPE	メジャー・タイプ (「Definition」など)	読取専用
DEFNFLD_SAMAJORTYPENUMBER	メジャー・タイプ番号 (「3」など)	読取専用
DEFNFLD_SANAME	定義名。	文字列
DEFNFLD_SATYPE	SA の定義タイプ (UML Class、Entity など)	読取専用
DEFNFLD_SATYPENUMBER	定義タイプを表す内部定数。詳細については、SA ディレクトリーの DEFNS.BAS ファイルを参照してください。	読取専用

DEFFLD 定数	説明	データ型
DEFNFLD_SAUPDATEDATE	最終更新日。	読取専用
DEFNFLD_SAUPDATETIME	最終更新時刻。	読取専用

定義メトリック

従来、メトリックはさまざまな Rational System Architect レポートにおいて、リストの作成、ルール・チェックの実行、および計算を行うために使用されてきました。現在は、Definition クラスの GetMetric メソッドを呼び出して、ユーザーがメトリックを個別に実行できるようになりました。SA オブジェクト・ブラウザーの DEFINITIONMETRIC 列挙型リストには、すべての定義メトリックの一覧があります。次の表に、使用可能なすべての定義メトリック、およびその説明を示します。

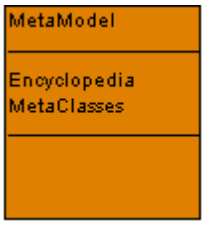
定義メトリック	説明
DEFMETBOTTOM	派生 T/F ブール値フィールドが最下位レベルになります。式でない定義の場合、およびデータ要素やデータ構造が含まれていない式の場合、この値は True です。
DEFMETCURRENT	T/F ブール値フィールド。現在表示されているダイアグラムに定義のシンボルが配置されている場合は、True を返します。
DEFMETEXPRESSION	定義の式で使用されている、間違った式構文、または未定義のデータ要素やデータ構造のリストを作成します。
DEFMETISFOREIGNKEY	定義が外部キーの場合、True を返します。
DEFMETKEYCOMPNBR	定義の主キーのコンポーネント番号を戻します。コンポーネント番号 (@1、@2 など) は、ブラウザー詳細内の定義の属性リストを展開して表示することもできます。
DEFMETNORMALIZE1	定義が第 1 正規形であるかどうかを確認します。繰り返しグループが含まれていないエンティティーは、第 1 正規形です。
DEFMETNORMALIZE23	定義が第 2 および第 3 正規形であるかどうかを確認します。エンティティーが第 1 正規形に従っていて、それぞれの非キー属性の機能が完全に主キーに従属している場合、そのエンティティーは第 2 正規形になっています。エンティティーが第 2 正規形に従っていて、各非キー属性が主キーのみに従属している場合、そのエンティティーは第 3 正規形になっ

定義メトリック	説明
	ています。
DEFMETPARENTSLASHDATA	属性のキー設定元に関する情報が格納された、定義の外部キー・スラッシュ・データを戻します。外部キー・スラッシュ・データは、ブラウザー詳細内の定義の属性リストを展開して表示することもできます。スラッシュ・データは、次の例のような形式で表示されます。 Row_Number / FKFROM “Stock_Location.Row_Number(stores)” /
DEFMETREFERENCE	定義が他のオブジェクトから参照されている場合は、True を返します。
DEFMETSELECTED	定義によって指定された選択済みシンボルがある場合は、True を返します。
DEFMETSEQINPARENTSLIST	親オブジェクトのプロパティー・セットを調べ、子オブジェクトのリストを検索します。このリストに定義が出現する回数を戻します。
DEFMETSYNCHRONIZE	定義を別のオブジェクトと同期する必要があることが指定されている場合、このメトリックは同期を実行します。同期は SA2001.INI エディターで設定できます。
DEFMETUPDATEUSES	データ・ディクショナリーの関係テーブル (RELATN.DBF) を更新します。戻り値はありません。
DEFNMETCHARCOUNT	定義の記述プロパティーの文字数を戻します。
DEFNMETLINECOUNT	定義の記述プロパティーの行数を戻します。
DEFNMETWORDCOUNT	定義の記述プロパティーのワード数を戻します。

9

MetaModel クラス

はじめに
MetaModel
オブジェクトは、エンサイクロペディアがメタクラス・オブジェクトにアクセスできるようにします。



属性

Encyclopedia

目的
メタモデル・オブジェクトの親 Encyclopedia オブジェクトを参照します。

パラメーター
読取専用

MetaClasses

目的
エンサイクロペディアが MetaClass
オブジェクトのコレクションにアクセスできるようにします。

パラメーター
データ型: SA Collection

読取専用

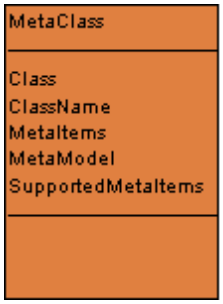
例

```
Dim coll As SACollection, i As Integer
Set coll = sa.Encyclopedia.metamodel.MetaClasses
  For i = 1 To coll.Count
    Debug.Print coll.Item(i).Class
  Next i
```

10

MetaClass クラス

はじめに
MetaClass
オブジェクトは、リポジトリー内のオブジェクトのクラス・レベル情報を戻します。



属性

Class

目的
リポジトリー内のオブジェクトのクラス番号。戻り値は、ダイアグラムの場合は
1、シンボルの場合は 2、定義の場合は 3 です。メジャー・タイプ番号とも呼ばれます。

パラメーター
データ型: Long

読取専用

例

```
Dim coll As SACollection, i As Integer
Set coll = sa.Encyclopedia.metamodel.MetaClasses
  For i = 1 To coll.Count
    Debug.Print coll.Item(i).Class
  Next i
```

ClassName

目的
クラスの名前。エンサイクロペディアで有効なクラス名は、Diagrams、Symbols、Definitio
ns です。メジャー・タイプとも呼ばれます。

パラメーター
データ型: 文字列

読取専用

例

```
Dim coll As SACollection, i As Integer
Set coll = sa.Encyclopedia.metamodel.MetaClasses
  For i = 1 To coll.Count
    Debug.Print coll.Item(i).ClassName
  Next i
```

MetaItems

目的
このプロパティーは、すべてのメタ項目が MetaItems
クラスにアクセスできるようにします。

パラメーター
データ型: SACollection

読取専用

例

```
Dim coll As SACollection, i As Integer
Set coll = sa.Encyclopedia.metamodel.MetaClasses
  For i = 1 To coll.Count
    Debug.Print coll.Item(i).MetaItems.Count
  Next i
```

MetaModel

目的
親 MetaModel クラスにアクセスできるようにします。

パラメーター
読取専用

SupportedMetaItems

目的
このプロパティーは、このエンサイクロペディアに対してオンになっているメタ項目のみ
にアクセスできるようにします。

パラメーター
データ型: SACollection

読取専用

例

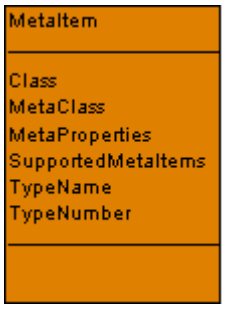
```
Dim sa As SA2001.Application
Set sa = New SA2001.Application
```

```
Debug.Print
  sa.Encyclopedia.MetaModel.MetaClasses.Item(1).SupportedMetaItems.Count
Set sa = Nothing
```

11

MetaItem クラス

はじめに
このオブジェクトによって、エンサイクロペディア内の個々のオブジェクト・タイプに関する情報が提供されます。この情報は、特定のオブジェクト・タイプに関する saprops.cfg および usrprops.txt 内の情報に対応しています。



属性

Class

目的
オブジェクトのクラス。有効な値は、ダイアグラムの場合は 1、シンボルの場合は 2、定義の場合は 3 です。メジャー・タイプ番号とも呼ばれます。

パラメーター
データ型: Long

読取専用

例
```
Dim sa As SA2001.Application
Set sa = New SA2001.Application
With
  sa.Encyclopedia.MetaModel.MetaClasses.Item(1).Suppo
  rtedMetaItems.Item(1)
    Debug.Print .Class
End With
```

MetaClass

目的
このプロパティーは、親 MetaClass オブジェクトにアクセスできるようにします。

パラメーター
読取専用

MetaProperties

目的

このプロパティーは、このオブジェクト・タイプでコレクションとしてサポートされているすべてのプロパティーにアクセスできるようにします。

パラメーター
データ型: SACollection

読取専用

例

```
Dim coll As SACollection, i As Integer
Set coll = Definition.MetaItem.MetaProperties
  For i = 1 To coll.Count
    Debug.Print coll.Item(i).Name
  Next i
```

SupportedMetaItems

目的
このプロパティーは、このエンサイクロペディアに対してオンになっているメタ項目のみにアクセスできるようにします。

パラメーター
データ型: SACollection

読取専用

TypeName

目的
オブジェクト・タイプの名前。

パラメーター
データ型: 文字列

読取専用

例

```
Dim sa As SA2001.Application
Set sa = New SA2001.Application
```

```
With
  sa.Encyclopedia.MetaModel.MetaClasses.Item(1).Suppo
  rtedMetaItems.Item(1)
    Debug.Print .TypeName
End With
```

TypeNumber

目的

Rational System Architect
によってこのオブジェクト・タイプに割り当てられている数値定数。

パラメーター

データ型: Long

読取専用

例

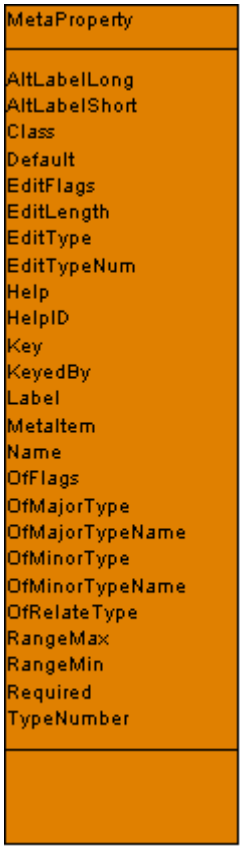
```
Dim sa As SA2001.Application
Set sa = New SA2001.Application
With
  sa.Encyclopedia.MetaModel.MetaClasses.Item(1).Suppo
  rtedMetaItems.Item(1)
    Debug.Print .TypeNumber
End With
```

12

MetaProperty クラス

はじめに

MetaProperty
オブジェクトを使用すると、エンサイクロペディア内の特定のオブジェクトの各プロパティーに関する情報を取り出すことができます。このオブジェクトは、saprops.cfg ファイルおよび usrprops.txt ファイルの Property キーワードに対応しています。



属性

以下のプロパティーを取得するには、次の例を以下に示すいずれかのプロパティーの名前に置き換えて使用します。

```
Dim sa As SA2001.Application
Set sa = New SA2001.Application
With
  sa.Encyclopedia.MetaModel.MetaClasses.Item(1).Suppo
  rtedMetaItems.Item(1).MetaProperties.Item(1)
    Debug.Print .Name
End With
Set sa = Nothing
```

AltLabelLong
プロパティーの Rational System Architect 代替長ラベル。

データ型: 文字列

読取専用

AltLabelShort
プロパティーの Rational System Architect 代替短ラベル。

データ型: 文字列

読取専用

Class
プロパティーの親オブジェクト・クラス・タイプ。メジャー・タイプ番号とも呼ばれます。

データ型: Long

読取専用

Default
初回使用時のプロパティーのデフォルト値セット。

データ型: 文字列

読取専用

EditFlags
プロパティーについて記述する必要があるプロパティーの数。

データ型: Long

読取専用

EditLength
プロパティーの編集の長さ。

データ型: Long

読取専用

EditType
SAEditTypes のリストは、以下の表を参照してください。

データ型: 文字列

読取専用

EditTypeNum
SAEditType の数値定数のリストは、以下の表を参照してください。

データ型: Long

読取専用

Help
このプロパティーのヘルプ・テキスト・セット。

データ型: 文字列

読取専用

HelpID
ヘルプ・テキストの ID 番号。

データ型: Long

読取専用

Key
プロパティーがキーであるかどうか。

データ型: ブール値

読取専用

KeyedBy
プロパティーに別のプロパティーによるキーが付いているかどうか。

データ型: SACollection

読取専用

Label
プロパティーの Rational System Architect 標準ラベル。

データ型: 文字列

読取専用

MetaItem
親 MetaItem オブジェクトにアクセスできるようにします。

読取専用

Name
プロパティー名。

データ型: 文字列

読取専用

OfFlags
プロパティーのフラグの数。

データ型: Long

読取専用

OfMajorType
親オブジェクト・クラスの数値定数。

データ型: Long

読取専用

OfMajorTypeName
親オブジェクト・クラスのタイプ名。

データ型: 文字列

読取専用

OfMinorType
現行の親オブジェクトの数値タイプ定数。

データ型: Long

読取専用

OfMinorTypeName
現行の親オブジェクトのタイプ名。

データ型: 文字列

読取専用

OfRelateType
親オブジェクトの関係番号。関係番号のリストについては、第 15 章を参照してください。

データ型: Long

読取専用

RangeMax
最大編集範囲値。

データ型: Long

読取専用

RangeMin
最小編集範囲値。

データ型: Long

読取専用

Required
オブジェクトの必須プロパティー。

データ型: Long

読取専用

TypeNumber
親オブジェクト・クラスのタイプ番号。

データ型: Long

読取専用

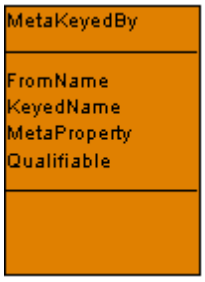
EditType	番号	説明
テキスト	1	テキストとして定義されたプロパティーは、リストや、ユーザーの入力した英数字である可能性があります。
日付	2	プロパティーの長さは 10 文字で、Windows に設定された日付形式に基づきます。
数値	3	プロパティーの値が数値でなければならないことを指定します。
ブール	4	プロパティーは、真 (T) または偽 (F) の 2 つのうちいずれかの値を持ち、定義ダイアログ・ボックスにチェック・ボックスとして表示されます。
式	5	プロパティーの値を + 符号で区切られた一連のストリングとして入力する必要があります。このワードは EXPRESSIONOF で置き換えられています。
ミニスペック	6	値がプロセス・シンボルの処理ロジックを表すプロパティー。 ミニスペックは、一般に構造化英語と呼ばれる公式の構文を使用して記述されます。
Time	7	プロパティーには、Windows に定義されている時刻形式に適合する表記のタイム・スタンプが含まれます。
ListOf	8	現行の定義タイプと、箇条書きのプロパティーのすべての項目の ListOf 設定に従って命名された定義タイプとの間に、1 対 1 または 1 対多の関係を形成します。
ExpressionOf	9	現行の定義タイプとプロパティーのすべての項目の ExpressionOf 設定に従って命名された定義タイプとの間に、1 対 1 または 1 対多の関係を形成します。

EditType	番号	説明
OneOf	10	現行の定義タイプと、プロパティーの項目の OneOf 設定に従って命名された定義タイプとの間に、1 対 1 の関係を形成します。
TemplateOf	11	トリガー・テンプレートの構文。
ParmListOf	12	現行の定義タイプと、その名前およびパラメーターによって指定され、箇条書きのプロパティーのすべての項目の ParmListOf 設定に従って命名された定義タイプとの間に、1 対 1 または 1 対多の関係を形成します。
ParmOneOf	13	現行の定義タイプと、その名前およびパラメーターによって指定され、プロパティーの項目の OneOf 設定に従って命名された定義タイプとの間に、1 対 1 の関係を形成します。

13

MetaKeyedBy クラス

はじめに
ここでは、以下の MetaKeyedBy クラスとその属性を示します。



この章のトピック	ページ
属性	13-2

属性

FromName

目的
MetaProperty のキー修飾の From Name プロパティー。

パラメーター
データ型: 文字列

読取専用

KeyedName

目的
MetaProperty のキー付きの名前。

パラメーター
データ型: 文字列

読取専用

MetaProperty

目的
親 MetaProperty クラスにアクセスできるようにします。

パラメーター
読取専用

Qualifiable

目的
MetaProperty が修飾可能であるかどうか (つまり、MetaProperty がキー構造への参照を保持するかどうか) の値。

パラメーター
データ型: ブール値

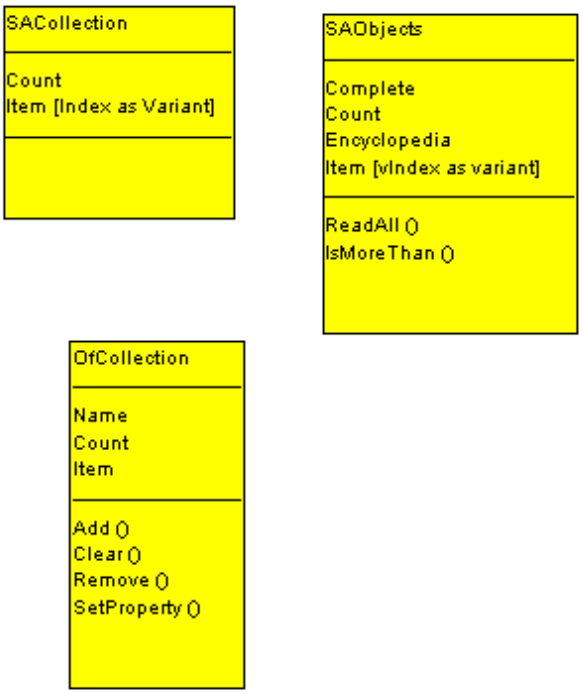
読取専用

14

Rational System Architect のコレクション

はじめに

以下の図に、Rational System Architect コレクション・クラスとその属性を示します。



SAObjects クラス

このコレクション・クラスは、類似した Rational System Architect オブジェクト (ダイアグラム、シンボル、または定義) のグループの操作に使用します。

Complete

目的
読み取り可能なメンバーがすべてコレクションに読み取られた場合に True を示すブール値。

パラメーター
データ型: ブール値

読取専用

例
```
Dim coll As SAObjects, i As long
Set coll = sa.Encyclopedia.GetAllDiagrams
i = 1
  Do While coll.IsMoreThan(i)
     i = i +1
     Debug.Print coll.Item(i).Name
  Loop
Debug.Print coll.Complete
Debug.Print coll.Count
```

Count

目的
コレクション・メンバーの数を表す値。

パラメーター
データ型: Long

読取専用

例

```
Dim coll As SAObjects, i As Integer
Set coll = sa.Encyclopedia.GetAllDiagrams
coll.ReadAll
  For i = 1 To coll.Count
     Debug.Print coll.Item(i).Name
  Next i
```

Encyclopedia

目的

コレクション内の現行のオブジェクトの Encyclopedia クラスへの逆方向の参照。

パラメーター

読取専用

Item(Index)

目的

コレクションの各メンバーは、コレクション内に、コレクションの作成時に派生した特定の指標番号を持っています。項目変数は、コレクションの指標に基づくオブジェクトです。その指標番号または名前 (わかっている場合)
で、このオブジェクトを参照することができます。

パラメーター

データ型: ブール値

読取専用

例

```
Dim coll As SAObjects, i As Integer
Set coll = sa.Encyclopedia.GetAllDiagrams
coll.ReadAll
  For i = 1 To coll.Count
     Debug.Print coll.Item(i).Name
  Next i
```

IsMoreThan(Index)

目的
コレクションに現行の指標値よりも多くの項目がある場合に True を返します。コレクションを 1 つずつ読み取る場合に使用します。

パラメーター
データ型: ブール値

例
```
Dim coll As SAObjects, i As long
Set coll = sa.Encyclopedia.GetAllDiagrams
i = 1
  Do While coll.IsMoreThan(i)
     i = i +1
  Loop
```

ReadAll

目的
タイプの読み取り可能なすべてのオカレンスをコレクション内に読み取ります。

例
```
Dim coll As SAObjects, i As Integer
Set coll = sa.Encyclopedia.GetAllDiagrams
coll.ReadAll
  For i = 1 To coll.Count
     Debug.Print coll.Item(i).Name
Next i
```

SACollection クラス

このコレクション・クラスは、類似した Rational System Architect
プロパティーのグループの操作に使用します。

Count

目的
コレクション・メンバーの数を表す数値を戻します。

パラメーター
データ型: Long

読取専用

例
```
Dim coll As SACollection, i As Integer
Set coll = Definition.MetaItem.MetaProperties
  For i = 1 To coll.Count
    Debug.Print coll.Item(i).Name
  Next i
```

Item(Index)

目的
コレクションの各メンバーは、コレクション内に、コレクションの作成時に派生した特定の指標番号を持っています。項目変数は、コレクションの指標に基づくオブジェクトです。その指標番号または名前 (わかっている場合)
で、このオブジェクトを参照することができます。

パラメーター
データ型: ブール値

読取専用

例
```
Dim coll As SACollection, i As Integer
Set coll = Definition.MetaItem.MetaProperties
```

```
  For i = 1 To coll.Count
Debug.Print coll.Item(i).Name
  Next i
```

OfCollection クラス

このコレクション・クラスは、以下の表に示す特別なプロパティー・タイプのコンポーネントであるダイアグラムまたは定義のグループの操作に使用します。

プロパティー・タイプ	説明
ListOf	現行の定義タイプと、箇条書きのプロパティーのすべての項目の ListOf 設定に従って命名された定義タイプとの間に、1 対 1 または 1 対多の関係を形成します。
OneOf	現行の定義タイプと、プロパティーの項目の OneOf 設定に従って命名された定義タイプとの間に、1 対 1 の関係を形成します。

Name

目的
プロパティー・リストの名前。

パラメーター
データ型: 文字列

読取専用

例

```
Dim oDef as Definition, coll as OfCollection
Set coll = oDef.GetPropertyAsCollection("Operations")
  Debug.Print coll.Name
```

Count

目的
コレクション・メンバーの数を表す数値。

パラメーター
データ型: Long

読取専用

例

```
Dim oDef as Definition, coll as OfCollection, i as integer
Set coll = oDef.GetPropertyAsCollection("Operations")
  Debug.Print coll.Name
  For i = 1 to coll.Count
     Debug.Print coll.Item(i).Name
  Next i
```

Item

目的

コレクションの各メンバーは、コレクション内に、コレクションの作成時に派生した特定の指標番号を持っています。項目変数は、コレクションの指標に基づくオブジェクトです。その指標番号または名前 (わかっている場合)
で、このオブジェクトを参照することができます。

パラメーター

データ型: ブール値

読取専用

例

```
Dim oDef as Definition, coll as OfCollection, i as integer
Set coll = oDef.GetPropertyAsCollection("Operations")
  Debug.Print coll.Name
  For i = 1 to coll.Count
     Debug.Print coll.Item(i).Name
  Next i
```

Add

目的

プロパティー・リストにダイアグラム・オブジェクトまたは定義オブジェクトを追加します。

構文

```
OfCollection Object.Add(Item[, Before[, After]]
```

`OfCollection Object`

使用: 必須

データ型: オブジェクト

インスタンス化された任意の OfCollection クラス

`Item`

使用: 必須

データ型: 文字列

プロパティー・リストに追加するダイアグラム・オブジェクトまたは定義オブジェクト。

`Before`

使用: オプション (After パラメーターが設定されている場合は設定できません)

データ型: Long

コレクション・オブジェクト・カウント内のダイアグラム・オブジェクトまたは定義オブジェクトの数。この後にダイアグラム・オブジェクトまたは定義オブジェクトが追加されます。

`After`

使用: オプション (Before パラメーターが設定されている場合は設定できません)

データ型: Long

コレクション・オブジェクト・カウント内のダイアグラム・オブジェクトまたは定義オブジェクトの数。この前に、新しいダイアグラム・オブジェクトまたは定義オブジェクトが追加されます。

例

```
Dim i As Integer, oDef As Definition, coll As OfCollection
Set coll = oDef.GetPropertyAsCollection("Location Types")
  coll.Add Chr(34) & "Regional Office" & Chr(34)
  coll.SetProperty
  oDef.Save
```

Clear

目的
コレクション内のすべての項目の OfCollection オブジェクトを消去します。

例
```
Dim i As Integer, oDef As Definition, coll As OfCollection
Set coll = oDef.GetPropertyAsCollection("Location Types")
  coll.Clear
  oDef.Save
```

Remove

目的
コレクションからダイアグラム・オブジェクトまたは定義オブジェクトを除去します。

構文
```
OfCollection Object.Remove(Index)
```

OfCollection Object

使用: 必須

データ型: オブジェクト

インスタンス化された任意の OfCollection クラス

索引

使用: 必須

データ型: Long

コレクション・オブジェクト・カウント内の除去するダイアグラム項目または定義項目の数。

例
```
Dim i As Integer, oDef As Definition, coll As OfCollection
Set coll = oDef.GetPropertyAsCollection("Location Types")
  coll.Remove(2)
  coll.SetProperty
```

```
  oDef.Save
```

SetProperty

目的

OfCollection
オブジェクト内にプロパティー・リストを保存します。また、定義オブジェクトにコレクションが保存されるプロパティー・リストが含まれるため、定義オブジェクト自体が保存されることが重要です。

例

```
Dim i As Integer, oDef As Definition, coll As OfCollection
Set coll = oDef.GetPropertyAsCollection("Location Types")
  coll.Add Chr(34) & "Regional Office" & Chr(34)
  coll.SetProperty
  oDef.Save
```

15

Rational System Architect のイベント

はじめに

ユーザーが実行したアクションや、そのアクションに対する Rational System Architect の応答に対して、なんらかの制御を実行する必要がある場合があります。例えば、特定の操作を禁止し、警告を出して、動作を強制したり、追加の操作を実行したりする必要があります。このような場合、アクションの発生時に、明確にそのアクションを識別して、適切に対応する必要があります。

特定の時点で発生するアクションは「イベント」と呼ばれます。Rational System Architect では、イベントの発生時にイベントを処理することができます。

Rational System Architectでは、シンボルがダイアグラムに配置された、ダイアグラムが保存された、エンサイクロペディアが開いた、などの多くのイベントが識別されます。以下に、識別されるすべてのイベントと渡される変数の詳細をリストします。

アプリケーション・イベント

AuditIDChanged

目的
AuditID が変更され、新規 AuditId が NewAuditID として渡されます。監査 ID は**「ファイル」、「監査 ID...」**メニューで変更できます。

構文
AuditIDChanged(NewAuditID)

NewAuditID

使用: 必須

データ型: 文字列

AuditID プロパティーの変更後のユーザー名。

例

DiagramClose

目的
ダイアグラムが閉じました。

構文
DiagramClose(hdgm)

hdgm

使用: 必須

データ型: Long

ダイアグラム・ハンドルが hDgm として渡されます。

例

DiagramOpen

目的

ダイアグラムが開きました。

構文

DiagramOpen(hdgm)

hDgm

使用: 必須

データ型: Long

ダイアグラム・ハンドルが hDgm として渡されます。

例

DiagramSave

目的

ダイアグラムが保存されました。

例

EncyClose

目的

エンサイクロペディアが閉じました。

例

EncyOpen

目的

エンサイクロペディアが開きました。

例

MainMenuUpdate

目的

メインメニューが更新されました。「ファイル」メニューおよびその他の関連メニュー項目です。

例

MethodMenuUpdated

目的
「描画」メニューが更新されました。新規メソッド・タイプにより、強制的にメニューが再描画され、新規メニュー項目が追加された場合です。

例

ReportsMenuUpdate

目的
「レポート」メニューが更新されました。メソッド固有のレポートまたはルール・チェックを反映するため、Rational System Architect により「レポート」メニューが変更された場合です。

例

ToolsMenuUpdate

目的
「ツール」メニューが更新されました。使用中の特定のメソッドに適用可能なツール・オプションを反映するため、「ツール」メニューが更新された場合です。

例

ShowNode

目的
ブラウザーの VBA フィルターがアクティブのときに、イベントがトリガーされました。RetCode を設定して、この項目を表示/非表示にします。

構文

ShowNode(TabIndex, TabName, ddId, Major, Minor, Name, Memo, LockFlags, RetCode)

TabIndex

使用: 必須

データ型: 整数

タブ名	タブの指標番号
すべてのメソッド	1
データ・モデリング	2
ビジネス・プロセス	3
OO レガシー	4
アプリケーション	5
構造化	6
組織	7
技術	8
場所	9
ビジネスの方向づけ	10
UML	11
XML	12

TabName

使用: 必須

データ型: 文字列

上記の表を参照してください。

ddId

使用: 必須

データ型: Long

`Major`

使用: 必須

データ型: Long

`Minor`

使用: 必須

データ型: Long

`Name`

使用: 必須

データ型: 文字列

`Memo`

使用: 必須

データ型: 文字列

`LockFlags`

使用: 必須

データ型: Long

`RetCode`

使用: 必須

データ型: NodeStatus

ノードの状況	番号
StatusDONTSHOW	0
StatusSHOW	1
StatusDONTCARE	2

例

ShutDown

目的
Rational System Architect がシャットダウンされました。

例

StartUp

目的
Rational System Architect が開始されました。

例

SymbolEvent

目的
シンボルがダイアグラムに配置されました。

構文
SymbolEvent(hDgm, hSym, hSymOther, SymEvent, lData)

hDgm

使用: 必須

データ型: Long

ダイアグラムのハンドル。

hSym

使用: 必須

データ型: Long

シンボルのハンドル。

hSymOther

使用: 必須

データ型: Long

ライン・シンボル (ノード・シンボルの接続先/切断先) のハンドル。

SymEvent

使用: 必須

データ型: SYMEVENTS

SYMEVENTS	番号	説明
ADDCONN	64	ノード・シンボルがライン・シンボルと接続されました。
BREAKCONN	65	ノード・シンボルとライン・シンボルとの接続が切断されました。
SYM_DESELECTED	257	シンボルが選択解除されました。
SYM_SELECTED	256	シンボルが選択されました。
TRANSFORMED	66	シンボルが別のシンボルに変換されました。特定のシンボルが選択されている場合にそれを右クリックして「変換」を選択すると実行されます。

Ldata

使用: 必須

データ型: Long

メニューにマクロ項目をプログラムで追加するためのガイドライン

Rational System Architect V10.1 までで、以前の固定フォーマットのメニューおよびツールバーが、ほぼすべて解放されました。つまり、現在では、ユーザーがメニューをカスタマイズすることができ、カスタマイズしたものをそのまま使用することができます。唯一の例外は「描画」メニューとツールバーです。これらについては、デフォルトの描画ツールのコマンドやボタンのリストより後の部分しかカスタマイズできません。

この変更の目的は、SA カタログ・マネージャーを使用して、特定のユーザー・グループが、作り替えられたメニューおよびツールバーの構成を使用できるようにすることです。

Rational System Architect V10.0 以前のマクロ・メニュー・コードは、Rational System Architect が特定のメニューを破棄することを考慮して作成され、それを強制的に再描画する傾向がありました。メニューが破棄されるので、セッション中にメニュー項目が表示されないようにする必要があったユーザーは、プログラムでメニューから項目を実際に除去せずに済みました。典型的な例は、ダイアグラムが閉じた場合です。この場合には、「ツール」メニューは破棄され、再作成されます。したがって、ユーザーは、その時点で必要な場合に、メニュー項目を再挿入するだけで済みました。現在では、Rational System Architect でメニューが破棄されなくなったため、これらのメニュー項目は、マクロ・コードを変更して除去 (非表示に) しない限り、表示されたままになります。Rational System Architect 10.0 以前に作成されたマクロの変更については、IBM サポート・サイトで Rational System Architect の変換マニュアル Conversion.pdf を参照してください。

Rational System Architect V10.1 以降のメニューにマクロを追加するためのガイドラインについては、このセクションの以降の部分で説明します。

Rational System Architect 標準メニューにマクロを追加する方法

マクロ項目の追加をサポートするためのメインとなる SA2001.Application メソッドは、以下のとおりです。

InsertMacroItemInMenu(MacroName as String, MacroItemCaption as String, InMenuTitleCaption as String, [BeforeMenuItemCaption as string]) as long

この関数は、InMenuTitleCaption というタイトルの SA/ユーザー・メニューに対して、キャプション MacroItemCaption を使用して、BeforeMenuItemCaption というタイトルのメニュー項目の前に MacroName ("<Project>.<Module>.<Subroutine>" から構成される) を追加します。Before 項目が指定されていない場合は、メニューの最後に項目が追加されます。Before 項目が #TOP# である場合、項目はメニューの先頭に挿入されます。

成功した場合は関数からの戻りはゼロであり、成功しなかった場合はゼロ以外になります。

つまり、以下のようになります。

```
Set App = New SA2001.Application
```

x = App.InsertMacroItemInMenu("MyProject.MyModule.MySub", "テスト項目(&T)", "ツール(&T)","マクロ(&C)")

これにより、項目「テスト項目」が、「ツール」メニューの「マクロ」ポップアップ項目の前に挿入されます。この項目がクリックされると、プロジェクト「MyProject」のモジュール「MyModule」で、サブルーチン「MySub」が実行されます。

これを成功させるには、指定したサブルーチンが既に存在している必要があります。また、メニューのキャプションが正確に指定されている必要があります。

例えば、 ツール は、ストリングの ツール(&T)

アンパーサンドは、ニーモニック・キーを意味します。これには大/小文字の区別があります。

サブルーチンの名前の後には、() は付けないでください。

メニュー項目に BMP を追加するには、AssignBMPtoMacroItem メソッドを使用します。

AssignBMPtoMacroItem(MacroName as String, BMPFileName as String) as long

成功した場合この関数からの戻りはゼロであり、成功しないかった場合はゼロ以外になります。

この関数は、メニューにマクロ項目を追加する前に一度だけ実行する必要があります。

以下に例を示します。

Set App = New SA2001.Application

x = App.AssignBMPtoMacroItem("MyProject.MyModule.MySub", "C:\Piccy.BMP")

x = App.InsertMacroItemInMenu("MyProject.MyModule.MySub", "テスト項目(&T)", "ツール(&T)","マクロ(&C)")

これにより、piccy.bmp が MySub マクロに追加され、そのマクロが前の例のようにメニューに追加されます。メニューのどの部分でもこのメニュー項目が表示されるようにすると、カスタマイズ・メニュー (メニューの右クリック) オプションに Piccy.BMP およびキャプション「テスト項目(&T)」が含まれます。

イベントの使用

以下の SA2001.Application イベントは、メニューの可視性を変更するタイミングを決定するマクロ・コードで役立ちます。

- MainMenuUpdate: メニュー全体が影響を受けたとき
- MethodMenuUpdate : 辞書タイプのメニュー項目が更新されたとき
- ToolsMenuUpdate: ツール・タイプのメニュー項目が更新されたとき
- ReportsMenuUpdate: レポート・タイプのメニュー項目が更新されたとき
- App_ShutDown: SA がシャットダウンしたとき

メニューのユーザー項目を保持するためにイベントを使用する必要はなくなりました。ただし、メニューから項目を除去する場合は、そのイベント処理の概要について、以下に示すコードを参照してください。(注: プロジェクトは「MyProject」とします。)

モジュール - AutoExec

```
' VBA ベースのイベント・ハンドラーを保守するメイン・モジュール
Dim EventHandler As EventCls
' イベント・ハンドラーを開始するメイン・サブルーチン
Sub Main()
  Set EventHandler = New EventCls    ' イベント・ハンドラーを作成
  Set EventHandler.App = Application ' イベント・ハンドラーを SA に接続
  ' 追加の開始処理がある場合はここで実行します
  ' BMP の追加やポップアップ・メニューの作成はここで行うことをお勧めします
  InitBMPs
  InitPopUps
End Sub

Sub InitBMPs()
  Dim x As Integer
  x = Application.AssignBMPtoMacroItem("MyProject.MyModule.PRINTDIAGRAMS",
"SAWORD.BMP")
End Sub

Sub InitPopUps()
  Dim x As Integer
  ' 注: ポップアップを作成すると、「RemovePopupMenu」で除去するまで存在しますが、
メニューから任意に追加/除去することもできます。

  ' ビットマップ付きのポップアップ・メニューを作成
  x = Application.CreatePopUpMenu("サンプル・マクロ","SAWORD.BMP")
  ' ポップアップに項目を追加
  x = Application.InsertMacroItemInMenu("MyProject.MyModule.PRINTDIAGRAMS", "ダイア
グラムの印刷(&P)", "サンプル・マクロ")
End Sub
```

クラス - EventCls

```
Public WithEvents App As Application ' イベントを発生させるアプリケーション

Private Sub Class_Initialize()
  ' ここでは何もする必要はありません
End Sub

Private Sub Class_Terminate()
  ' ここでは何もする必要はありません
End Sub

Private Sub App_ReportsMenuUpdate()
```

```
 '「レポート」メニューが再描画されます
  Dim x As Long
  x = App.SetSeparatorBefore("レポート生成(&R)...", "レポート(&R)", True)

 '「レポート」メニューの「レポート生成...」項目の前に、PrintDiagrams2 を呼び出す「すべてのダイアグラムの印刷」というメニュー項目を挿入
  x = App.InsertMacroItemInMenu("MyProject.MyModule.PRINTDIAGRAMS2", "すべてのダイアグラムの印刷(&D)", "レポート(&R)", "レポート生成(&R)...")

 '「レポート」メニューの「すべてのダイアグラムの印刷」項目の前に、ポップアップ・メニュー「サンプル・マクロ」を挿入
 ' 注: 特に理由がない場合は、ここでポップアップを破棄したり作成したりしないでください。初期化時に一度だけ行います。
  x = App.InsertPopupMenuItemInMenu("サンプル・マクロ", "レポート(&R)", "すべてのダイアグラムの印刷(&D)")
End Sub
```

モジュール - MyModule

```
Public Sub PrintDiagrams()
        ' 実行するユーザー・コード
        MsgBox "Print Diagrams"
End Sub
Public Sub PrintDiagrams2()
        ' 実行するユーザー・コード
        MsgBox "Print Diagrams2"
End Sub
```

ポップアップを追加する方法

ポップアップに使用する名前を指定して CreatePopUpMenu メソッドを実行します。必要に応じて BMP も指定します。ポップアップは SA のポップアップのルート・コレクションに追加されます。

追加されたポップアップは、SA を終了するときに RemovePopUpMenu を使用してイベント・ハンドラー・クラスで除去する必要はありません。

ポップアップにマクロ項目を追加する方法

SA メニューに追加する場合と同じ方法で追加します。

SA メニューにポップアップを追加する方法

Rational System Architect メニューにマクロ項目を追加する場合と同じ方法でメニューにポップアップを追加できます。ただし、InsertPopupMenuItemInMenu を使用し、マクロ名は指定しないでください。

以下に例を示します。

InsertPopupMenuItemInMenu(PopUpName as String, InMenuTitleCaption as String, [BeforeMenuItemCaption as string]) as long

メニュー項目にセパレーターを追加する方法

SetSeparatorBefore を使用して、メニュー名と項目名を指定し、セパレーターがある場合は True を、ない場合は False を設定します。

メニューから項目を除去する方法

RemoveItemFromMenu を使用して、メニュー名と項目名を指定すると、ポップアップまたはメニュー項目を除去できます。

注: 除去できるのは、追加した項目のみです。例えば、Rational System Architect メニュー項目は除去できません。項目は、実際に除去される訳ではなく、非表示になるだけです。このことを利用して、ユーザーはメニュー・システム全体にわたって項目をカスタマイズできます。メニュー名に“”を指定すると、メニュー名は無視されます。項目はメニュー・システム全体にわたって表示されません。

メニュー・システムに追加されたすべてのポップアップおよびツールは SA が再始動された後もそのまま残るため、不要になったポップアップ・メニューの除去が必要になる場合があります。App.RemovePopUpMenu(<PopupName>) メソッドを使用すると、メニュー・システム全体からポップアップが除去されます。

このメソッドをマクロで (例えば SA シャットダウン時に) 使用した場合、ポップアップおよびそのすべてのカスタマイズは SA のシャットダウンごとに除去されます。これには、このポップアップのカスタマイズが SA セッション間では保持されないという効果があります。

16

Rational System Architect の関係

はじめに

この章では、Rational System Architectリポジトリーのオブジェクト間に存在するすべての関係について説明します。SA オブジェクト・ブラウザーには、列挙型 RELATETYPE があります。この型は、VBA マクロで使用できる関係をすべて保持しています。

Diagram クラス、Symbol クラス、Definition クラスには、それぞれ、**GetRelatedObjects** メソッドがあります。このメソッドを実行するには、以降の表にリストされているものから、どの関係が存在するかを指定する必要があります。また、Encyclopedia クラスには、**GetRelationMetric** メソッドがあります。ユーザーは、この関係メトリックを実行するために、必須パラメーターとして関係タイプのいずれか 1 つを指定する必要があります。

関係タイプ

RELATETYPE	説明	番号
RELNULL	Null	0
RELNULL2	Null	1
RELDIAGRAMCON	ダイアグラムがシンボルを含んでいる	2
RELCONDIAGRAM	シンボルがダイアグラムに含まれている	3
RELSHOWTO	シンボルが (子) ダイアグラムに拡張されている	4
RELSHOWFROM	ダイアグラムが (親) シンボルから拡張されている	5
RELCONNSTART	ノード・シンボルがライン・シンボルの始点に接続されている	6
RELSTARTAT	ラインの始点がノード・シンボルに接続されている	7
RELCONNEND	ノード・シンボルがラインの終点に接続されている	8
RELENDAT	ラインの終点がノード・シンボルに接続されている	9
RELFLAGSENDS	モジュールがフラグ・シンボルに接続され、それを経由してデータを送信している	10
RELFLAGSTR	フラグ・シンボルがモジュールに接続され、そこからデータを受信している	11
RELFLAGRECVS	モジュールがフラグ・シンボルに接続され、そこからデータを受信している	12
RELFLAGEND	フラグ・シンボルがモジュールに接続され、それにデータを渡している	13
RELDFEELEMENT	式がデータ (要素または構造) を使用している	14

RELATETYPE	説明	番号
RELELEMENTDFE	データ (要素または構造) が式に使用されている	15
RELEXPLAINEDBY	シンボルがコメントによって説明されている	16
RELEXPLAINS	コメントがシンボルを説明している	17
RELPARTFULFILLS	シンボルが要求やテスト計画などを示している	18
RELPARTFULFILLEDBY	要求やテスト計画などがシンボルによって示されている	19
RELCOMPFULFILLS	シンボルが定義によって定義されている	20
RELDEFINEDBY	シンボルが定義によって定義されている	20
RELCOMPFULFILLEDBY	定義がシンボルを定義している	21
RELDEFINES	定義がシンボルを定義している	21
RELISQUALIFIEDBY	ライン・シンボルまたはノード・シンボルがフラグ・シンボルによって「修飾されて」いる	22
RELQUALIFIES	フラグ・シンボルがライン・シンボル (またはノード・シンボル) を「修飾して」いる	23
RELISA	定義が定義の「インスタンスである」	24
RELINSTBY	定義が定義によって「インスタンス化されて」いる	25
RELIDENTIFIES	定義が別の定義を「識別して」いる	26
RELKEYEDBY	定義が定義によって識別されている	27
RELEMBEDS	ノード・シンボルがシンボルを完全に組み込んでいる	28

RELATETYPE	説明	番号
RELISEMBEDDEDBY	ノード・シンボルがノード・シンボルによって完全に組み込まれている	29
RELISPARENTIN	定義が親子関係の定義の親である	30
RELHASPARENTOF	定義には親子関係の親定義がある	31
RELISCHILDIN	定義が親子関係の定義の子である	32
RELHASCHILDOF	定義には親子関係の子定義がある	33
RELCOMPRISES	定義が定義を含んでいる	34
RELISPARTOF	定義が定義の一部になっている	35
RELISCATZNOF	カテゴリー化定義が一般エンティティー定義をカテゴリー化している	36
RELHASCATZN	一般エンティティー定義にはカテゴリー化定義のカテゴリーがある	37
RELISCATGRYIN	エンティティー定義がカテゴリー化の 1 カテゴリーである	38
RELHASCATGRY0F	カテゴリー化にはエンティティー定義のカテゴリーがある	39
RELISCHILDOF	シンボルが階層ダイアグラムでのみ別のシンボルの子である	40
RELISPARENTOF	シンボルが階層ダイアグラムでのみ別のシンボルの親である	41
RELISFIRSTCHILDOF	シンボルが階層ダイアグラムでのみ別のシンボルの最初の子 (例えば、左端) である	42
RELHASFIRSTCHILD	シンボルには階層ダイアグラムでのみ最初の子である別のシンボルがある	43
RELISNEXTSIBLING	シンボルが階層ダイアグラムでのみ別の兄弟の次の兄弟である	44

RELATETYPE	説明	番号
RELISPRIORSIBLING	シンボルが階層ダイアグラムでのみ別のシンボルの前の兄弟である	45
RELISINDEXOF	データ・モデル - エンティティーやテーブルのアクセス・パスまたは索引	46
RELISINDEXEDBY	定義が別の定義によって索引付けされている	47
RELORIGINATESFROM	定義が定義から発生している (例えば、グラフィック画面ダイアグラム)	48
RELISORIGINOF	定義が定義の起点になっている	49
RELISBASEDON	定義が定義に基づいている (通常はデータ要素)	50
RELISBASISFOR	定義が派生定義の基底になっている	51
RELISLINKEDTO	シンボルが別のシンボルにリンクされている	52
RELISLINKEDWITH	シンボルが別のシンボルからリンクされている	53
RELUSER および RELUSERCOMPLEMENT	ユーザーおよびユーザー補足	54 から 83
RELPOPKIN および RELPOPKINCOMPLEMENT	IBM および IBM 補足	84 から 111
RELREPRESENTS	エクスプローラー・ダイアグラム・シンボルがオブジェクトを表している	112
RELISPRESENTEDBY	オブジェクトがエクスプローラー・ダイアグラム・シンボルによって表されている	113
RELPOPKIN および RELPOPKINCOMPLEMENT	IBM および IBM 補足	114 から 125
RELUSER および RELUSERCOMPLEMENT	ユーザーおよびユーザー補足	126 から 135

RELATETYPE	説明	番号
RELLINKS TO	オブジェクトが DOORS のオブジェクトにリンクされている	136
RELISLINKEDFROM	オブジェクトが DOORS のオブジェクトからリンクされている	137
RELISTOBESENTTO	オブジェクトが DOORS モジュールに送信される	138
RELISTORECEIVE	DOORS モジュールがオブジェクトを受信する	139
RELHASBEENSENTTO	オブジェクトが DOORS モジュールに送信された	140
RELHASRECEIVED	DOORS モジュールがオブジェクトを受信した	141

17

Rational System Architect のフィールド・タイプ

はじめに

この章では、Rational System Architect オブジェクト・ブラウザーで列挙タイプ FLDTYPE にリストされる使用可能なすべてのフィールド・タイプについて説明します。これらの定数によって、データが戻される形式が決定されます。通常、Rational System Architect では、フィールド・タイプが内部で設定されます。この列挙タイプは、Encyclopedia クラスの GetRelationMetric メソッドの必須パラメーターとして使用されます。また、Diagram クラス、Symbol クラス、および Definition クラスの GetMetric メソッドではオプション・パラメーターになります。

以下の表に、すべてのフィールド・タイプとその説明を示します。

フィールド・タイプ

フィールド・タイプ	番号	説明
FLDTYPAUTO	65	Rational System Architect で内部的に選択されるデフォルトのフィールド。
FLDTYPCHARACTER	67	256 文字以下のストリングを保持します。
FLDTYPDATE	68	日付フィールド (MM/DD/YYYY など)。
FLDTYPLOGICAL	76	ブール・フィールド
FLDTYPMEMO	77	メモ・フィールドには、4,095 文字以下のラージ・テキスト・ブロックが格納されます。
FLDTYPNUMERIC	78	フィールドの値が数値でなければならないことを指定します。
FLDTYPTIME	84	時間フィールド (時間:分:秒など)。

18

Rational System Architect のエラー

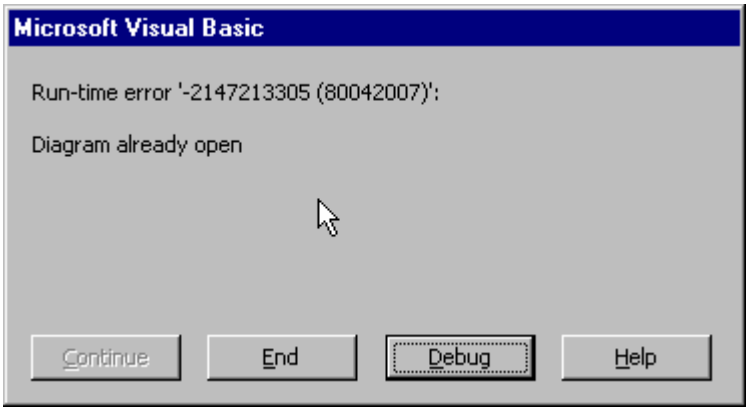
はじめに

この章では、Rational System Architect で生成されるエラーのうち、コード実行中にトラップできるものについて重点的に説明します。作成されたコード内のエラーは扱いません (例えば、変数のインスタンスが作成されていないことを示す Visual Basic エラー 91「オブジェクト変数または With ブロック変数が設定されていません。」など)。

エラー処理

Rational System Architect オブジェクト・モデルのプロパティーおよびメソッドは、一貫した方法でコードを実行しますが、基本データが常に予測どおりになるとは限らないため、VBA マクロの通常処理中にエラーが発生することがあります。通常処理中にさまざまな問題が発生し、これらのトラップや処理が必要になることがあります。トラップや処理を行わないと、マクロはコード実行を完了せずに、突然終了します。

コード実行中にエラーが発生すると、次のダイアログが表示されます。現在のダイアグラム・グラフィックを開くために、メソッド **Diagram.Show** が実行されました。しかし、このダイアグラムはコード内の別のポイントですでに開かれているため、エラーが発生しました。



ユーザーは、「終了 (End)」をクリックして、その時点でプログラムを終了したり、「デバッグ (Debug)」をクリックして、VBA エディター・インターフェースを開き、コード内のエラー発生場所を強調表示したりできます。詳細な説明を表示するには、「ヘルプ」ボタンをクリックします。

Rational System Architect のエラー

Rational System Architect には、識別できるさまざまなエラーがあります。VBA では、マクロ内で検出されたこのようなエラーを追跡し、VBA エラー・ハンドラーにメッセージを渡して、ユーザーにエラーを表示することができます。エラーごとに、キャッチされる問題は異なります。メソッドが実行されると、メソッドは呼び出しの状況を示す値を VBA に戻します。例外が発生すると、VBA はエラーを報告します。列挙型 SA2001Errors には、Rational System Architect のすべてのエラーの完全なリストが含まれています。

エラー	番号	説明
SAERR_BADDDID	8195	
SAERR_DIAGRAMNOTOPEN	8198	ダイアグラムが開いていません。ダイアグラムが開かれなかったか、または **.Hide** メソッドが以前に呼び出されています。
SAERR_DIAGRAMNOTSAME	8201	
SAERR_DIAGRAMOPEN	8199	ダイアグラムはすでに開いています。ダイアグラムが以前に開かれたか、または **.Show** メソッドが以前に呼び出されています。
SAERR_ENUMVARIANTERROR	8193	
SAERR_INVALIDCLASS	8202	
SAERR_INVALIDOBJECT	8194	
SAERR_INVALIDPROPERTY	8204	
SAERR_INVALIDTYPE	8207	
SAERR_NOTIMPLEMENTED	8196	
SAERR_OBJECTDOESNOTEXIST	8197	オブジェクトは、インスタンス化されなかったか、または以前に削除されました。
SAERR_OBJECTISLOCKED	8205	OpenLock メソッドが以前に呼び出されたか、またはオブジェクトが別のユーザーによって以前にチェックアウトまたはフリーズされました。
SAERR_OBJECTNOTFOUND	1025	

エラー	番号	説明
SAERR_OPENEDASREADONLY	8206	オブジェクトは、読み取り専用として開かれています。
SAERR_REQUIREDPROPERTYABSENT	8192	参照されているプロパティーは、ブランクであるか、または存在しません。
SAERR_SA20001_IMF_ERROR	4096	SAIMF エラーが発生しました。
SAERR_SANOTRUNNING	1024	Rational System Architect が開始されていないか、または以前にシャットダウンされました。
SAERR_SYMBOLHASNODIAGRAM	8200	シンボルがどのダイアグラムからも参照されていません。
SAERR_TOOMANYOBJECTS	8203	

19

IBM サポート

はじめに

問題のトラブルシューティングに役立つセルフ・ヘルプ方式の情報リソースおよびツールが多数あります。ご使用の製品に問題がある場合、以下の方法で対処できます。

ご使用の製品のリリース情報を参照し、既知の問題、回避策、およびトラブルシューティング情報を確認します。

問題の解決に利用できるダウンロードまたはフィックスがないか確認します。

使用可能な知識ベースを検索して、問題の解決策が文書化されていないか確認します。

それでも支援が必要な場合は、IBM®ソフトウェア・サポートへ連絡して問題を報告してください。

IBM Rational ソフトウェア・サポートへのお問い合わせ

セルフ・ヘルプ・リソースで問題を解決できない場合は、IBM® Rational® ソフトウェア・サポートにお問い合わせください。

注: 従来からの Telelogic のお客様は、すべてのサポート・リソースを 1 つの参照サイト (http://www.ibm.com/software/rational/support/telelogic/

前提条件

IBM Rational ソフトウェア・サポートに問題を送信するには、有効なパスポート・アドバンテージ®・ソフトウェア保守契約を締結している必要があります。パスポート・アドバンテージは、IBM の包括的なソフトウェア・ライセンス、およびソフトウェア保守 (製品のアップグレードおよび技術サポート) を提供するものです。パスポート・アドバンテージのオンライン登録は、http://www.ibm.com/software/lotus/passportadvantage/howtoenroll.html

- パスポート・アドバンテージについて詳しくは、パスポート・アドバンテージの FAQ (http://www.ibm.com/software/lotus/passportadvantage/brochures_faqs_quickguides.html) にアクセスしてください。
- さらに支援が必要な場合は、IBM 担当員にお問い合わせください。

問題をオンラインで (IBM の Web サイトから) IBM Rational ソフトウェア・サポートに送信するには、以下のようにしてください。

- IBM Rational ソフトウェア・サポートの Web サイトでユーザーとして登録します。登録について詳しくは、http://www.ibm.com/software/support/ を参照してください。
- サービス要求ツールの許可された呼び出し元としてリストに追加されていることを確認してください。

その他の情報

Rational ソフトウェア製品ニュース、イベント、およびその他の情報については、IBM Rational ソフトウェアの Web サイト (http://www.ibm.com/software/rational/.

問題の送信

問題を IBM Rational ソフトウェア・サポートに送信するには、以下のようにしてください。

1. 問題がビジネスに及ぼす影響を判別します。問題を IBM に報告する際には、重大度レベルを報告する必要があります。そのため、問題がビジネスに及ぼす影響を理解し、査定しておく必要があります。

重大度レベルを決定するには、以下の表を使用します。

重大度	説明
1	問題はビジネスに重大な影響を及ぼしている: プログラムを使用できないため、結果的にオペレーションに重大な影響を及ぼしている。この状態は、早急な解決策が必要です。
2	問題はビジネスにかなり影響を及ぼしている: プログラムは使用できるが、それはごく限られた部分だけである。
3	問題はビジネスにいくらか影響を及ぼしている: プログラムは使用できるが、それほど重要ではない機能 (オペレーション上重大ではない機能) が使用できない。
4	問題はビジネスにわずかに影響を及ぼしている: 問題がオペレーションに及ぼす影響がわずかであるか、または問題に対して適切な回避策が実施されている。

2. 問題を説明し、その背景となる情報を収集します。IBM に問題を説明する際には、できる限り具体的に説明してください。IBM Rational ソフトウェア・サポート・スペシャリストがお客様の問題を適切に解決できるように、関連するすべての背景情報を提供してください。

時間を節約するために、次の質問に対する答えを準備しておいてください。

- 問題が発生したときに実行していたソフトウェアのバージョンは何ですか。

- 正確な製品名とバージョンを判別するには、以下のうち該当するオプションを使用してください。
    - IBM Installation Manager を開始して、**「ファイル」 > 「インストール済みパッケージの表示 (View Installed Packages)」**をクリックします。パッケージ・グループを展開して、パッケージを選択し、パッケージ名とバージョン番号を確認します。
    - 製品を開始して、**「ヘルプ」 > 「バージョン情報」**をクリックし、表示される名称およびバージョン番号を確認します。

- オペレーティング・システムとバージョン番号は何ですか (Service Pack またはパッチを含む)。

- 問題の症状に関連するログ、トレース、およびメッセージはありますか。

- 問題を再現できますか。再現できる場合、問題を再現するための実行手順を教えてください。

- システムに何らかの変更を加えましたか。例えば、ハードウェア、オペレーティング・システム、ネットワーキング・ソフトウェア、またはその他のシステム・コンポーネントに変更を加えましたか。

3. 現在、問題の回避策をとっていますか。回避策をとっている場合には、問題を報告する際に、その回避策について説明できるようにしておいてください。

4. 次のいずれかの方法で、IBM Rational ソフトウェア・サポートに問題を送信します。

- オンライン: IBM Rational ソフトウェア・サポートの Web サイト (https://www.ibm.com/software/rational/support/) にアクセスします。Rational サポート・タスク・ナビゲーターで、「**Open Service Request**」をクリックします。電子問題報告ツールを選択して、Problem Management Record (PMR) を開き、問題についての説明を入力します。

- サービス要求を開く方法について詳しくは、http://www.ibm.com/software/support/help.html を参照してください。

- IBM サポート・アシスタントを使用して、オンライン・サービス要求を開くこともできます。詳しくは、http://www.ibm.com/software/support/isa/faq.html を参照してください。
- 電話による方法: お住まいの国または地域の電話番号については、各国の連絡先をリストした IBM ディレクトリー (http://www.ibm.com/planetwide/) にアクセスして、お住まいの国名または地域名をクリックしてください。

- IBM 担当員を通じて連絡する方法: オンラインまたは電話で IBM Rational ソフトウェア・サポートに連絡できない場合は、IBM 担当員にご連絡ください。必要に応じて、IBM 担当員がお客様に代わってサービス要求を開きます。国ごとの完全な連絡先情報については、http://www.ibm.com/planetwide/ を参照してください。

20

付録:

はじめに

この章では、IBM® Rational® System Architect®の法的使用および商標について記述しています。

特記事項

本書は米国 IBM が提供する製品およびサービスについて作成したものであり、 本書に記載の製品、サービス、または機能が日本においては提供されていない場合があります。日本で利用可能な製品、サービス、および機能については、日本 IBM の営業担当員にお尋ねください。 本書で IBM 製品、プログラム、またはサービスに言及していても、その IBM 製品、プログラム、または サービスのみが使用可能であることを意味するものではありません。これらに代えて、IBM の知的所有権を侵害することのない、機能的に同等の製品、プログラム、またはサービスを使用することができます。 ただし、IBM 以外の製品とプログラムの操作または サービスの 評価および検証は、お客様の責任で行っていただきます。

IBM は、本書に記載されている内容に関して特許権 (特許出願中のものを含む) を保有している場合があります。本書の提供は、お客様にこれらの特許権について 実施権を許諾することを意味するものではありません。 実施権についてのお問い合わせは、書面にて下記宛先にお送りください。

〒106-8711
東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産 知的財産権ライセンス渉外

2 バイト文字セット (DBCS) 情報に関するライセンス上の問い合わせについては、各国の IBM 知的財産部門に連絡するか、書面で下記宛先までお送りください。

〒106-8711 東京都港区六本木 3-2-12
日本アイ・ビー・エム株式会社
法務・知的財産
知的財産権ライセンス渉外

以下の保証は、国または地域の法律に沿わない場合は、適用されません。IBM および その直接または間接の子会社は、本書を特定物として現存するままの 状態で提供し、商品性の保証、特定目的適合性の保証および法律上の 瑕疵担保責任を含むすべての明示もしくは黙示の保証責任を負わないものと します。国または地域によっては、法律の強行規定により、保証責任の制限が禁じられる場合、強行規定の制限を受けるものとします。

この情報には、技術的に不適切な記述や誤植を含む場合があります。 本書は定期的に見直され、必要な変更は本書の次版に組み込まれます。 IBM は予告なしに、随時、この文書に記載されている製品またはプログラムに対して、 改良または変更を行うことがあります。

本書において IBM 以外の Web サイトに言及している場合がありますが、 便宜のため記載しただけであり、決してそれらの Web サイトを推奨するものでは ありません。それらの Web サイトにある資料は、この IBM 製品の資料の一部ではありません。それらの Web サイトは、お客様の責任でご使用ください。

IBM は、お客様が提供するいかなる情報も、お客様に対してなんら義務も負うことのない、 自ら適切と信ずる方法で、使用もしくは配布することができるものとします。

本プログラムのライセンス保持者で、(i) 独自に作成したプログラムと その他のプログラム (本プログラムを含む) との間での情報交換、 および (ii) 交換された情報の相互利用を可能にすることを目的として、 本プログラムに関する情報を必要とする方は、下記に連絡してください。

Intellectual Property Dept. for Rational Software
IBM Corporation
1 Rogers Street
Cambridge, MA 02142
U.S.A

本プログラムに関する上記の情報は、適切な使用条件の下で使用すること ができますが、有償の場合もあります。

本書で説明されているライセンス・プログラムまたはその他のライセンス資 料は、IBM 所定のプログラム契約の契約条項、IBM プログラムのご使用条件、 またはそれと同等の条項に基づいて、IBM より提供されます。

この文書に含まれるいかなるパフォーマンス・データも、管理環境下で 決定されたものです。 そのため、他の操作環境で得られた結果は、異なる可能性があります。 一部の測定が、開発レベルのシステムで行われた可能性がありますが、その測定値が、一般に利用可能なシステムのものと同じである保証はありません。 さらに、一部の測定値が、推定値である可能性があります。 実際の結果は、異なる可能性があります。お客様は、お客様の特定の環境に適したデータを確かめる必要があります。

IBM 以外の製品に関する情報は、その製品の供給者、出版物、 もしくはその他の公に利用可能なソースから入手したものです。IBM は、それらの製品のテストは行っておりません。したがって、 他社製品に関する実行性、互換性、またはその他の要求については確証できません。 IBM 以外の製品の性能に関する質問は、それらの製品の供給者にお願いします。

本書には、日常の業務処理で用いられるデータや報告書の例が含まれています。 より具体性を与えるために、それらの例には、個人、企業、ブランド、 あるいは製品などの名前が含まれている場合があります。 これらの名称はすべて架空のものであり、 名称や住所が類似する企業が実在しているとしても、それは偶然にすぎません。

著作権

本書には、様々なオペレーティング・プラットフォームでのプログラミング手法を例示するサンプル・アプリケーション・プログラムがソース言語で掲載されています。お客様は、サンプル・プログラムが書かれているオペレーティング・ プラットフォームのアプリケーション・プログラミング・インターフェースに 準拠したアプリケーション・プログラムの開発、使用、販売、配布を目的として、 いかなる形式においても、IBM に対価を支払うことなくこれを複製し、改変し、 配布することができます。 このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。 従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性が あることをほのめかしたり、保証することはできません。

それぞれの複製物、サンプル・プログラムのいかなる部分、またはすべての派生的創作物にも、次の ように、著作権表示を入れていただく必要があります。

この情報をソフトコピーでご覧になっている場合は、写真やカラーの図表は 表示されない場合があります。

商標

IBM、IBM ロゴ、ibm.com は、International Business Machines Corporation の米国およびその他の国における商標または登録商標です。その他の製品およびサービス名は、IBM または他の企業の商標です。現時点での IBM の商標リストについては、http://www.ibm.com/legal/copytrade.shtmlwww.ibm.com/legal/copytrade.html をご覧ください。

Microsoft および Windows は、Microsoft Corporation の米国およびその他の国における商標です。

他の会社名、製品名およびサービス名等はそれぞれ各社の商標です。