IBM Rational

System Architect

Handbuch zur Erweiterbarkeit mit VBA

Release 11.3.1

Lesen Sie vor Verwendung dieser Informationen die "Bemerkungen" im Anhang auf Seite 20-1.

Diese Ausgabe gilt für IBM® Rational® System Architect®, Version 11.3.1 und für alle nachfolgenden Releases und Bearbeitungen, sofern in neuen Ausgaben nicht anders angegeben.

Inhalts-verzeichnis

Inhaltsverzeichnis ..... **i**

Einführung **1-1**
Rational System Architect automatisieren 1-2
Rational System Architect mit VBA programmieren 1-3
Makro ausführen 1-4
Makroprojekte 1-5
Auf den VBA-Editor zugreifen 1-7
Objektbrowser 1-11

Automatisierung und Rational System Architect **2-1**
Automatisierung 2-2
Anpassbare Lösungen 2-8
Automatisierte Lösung mit Rational System Architect planen 2-10

Rational System Architect-Objektmodell **3-1**
Objektmodellklassen 3-4

Anwendungsklasse **4-1**
Attribute 4-3
Methoden 4-5

Die Enzyklopädieklasse **5-1**
Attribute 5-3
Methoden 5-6
Arbeitsbereichsmethoden 5-21
Aufrufe für Arbeitsbereichsanwendungsklassen 5-23
Ereignisse für Arbeitsbereichsanwendungsklassen 5-24
Beziehungsmessdaten 5-25

Die Diagrammklasse **6-1**
Attribute 6-3
Methoden 6-9
Diagrammfelder 6-21
Diagrammessdaten 6-28

Die Symbolklasse 7-1
Attribute 7-3
Methoden 7-14
Symbolfelder 7-23
Symbolmessdaten 7-32

Die Definitionsklasse 8-1
Attribute 8-3
Methoden 8-8
Definitionsfelder 8-16
Definitionsmessdaten 8-18

Die MetaModel-Klasse 9-1
Attribute 9-2

Die MetaClass-Klasse 10-1
Attribute 10-2

MetaItem-Klasse 11-1
Attribute 11-2

MetaProperty-Klasse 12-1
Attribute 12-3

MetaKeyedBy-Klasse 13-1
Attribute 13-2

Rational System Architect-Objektgruppen 14-1
SAObjects-Klasse 14-3
SACollection-Klasse 14-6
OfCollection-Klasse 14-8

Rational System Architect-Ereignisse 15-1
Anwendungsereignisse 15-2
Richtlinien für das programmgestützte Hinzufügen von Makroelementen zu Menüs 15-9

Rational System Architect-Beziehungen 16-1
Beziehungstypen 16-2

Rational System Architect-Feldtypen 17-1
Feldtypen 17-2

Rational System Architect-Fehler 18-1
Fehler behandeln 18-2
Rational System Architect-Fehler 18-3

IBM Unterstützung 19-1

IBM Rational-Softwareunterstützung kontaktieren 19-2

Anhang: 20-1
Bemerkungen 20-2
Marken 20-5

1

Einführung

Einführung
In diesem Kapitel sind die Grundlagen der Arbeit mit Rational System Architect und Visual Basic for Applications (VBA) beschrieben. Sie erhalten Informationen zur Verwendung von Makros, des VBA-Editors sowie des Objektbrowsers.

In diesem Kapitel enthaltene Themen	Seite
Rational System Architect automatisieren	1-2
Rational System Architect mit VBA programmieren	1-3
Beispielmakro ausführen	1-4
Makroprojekte	1-5
Auf den VBA-Editor zugreifen	1-7
Objektbrowser	1-11

Rational System Architect automatisieren

Rational System Architect ist VBA-aktiviert, was es dem Benutzer ermöglicht, die Umgebung von Rational System Architect programmgestützt und andere Anwendungen mithilfe der OLE-Automatisierung zu steuern.

Microsoft VBA und die zugehörige Entwicklungsumgebung werden zusammen mit Rational System Architect installiert. Die Programmierumgebung, die Fehlerbehebungsumgebung, die Sprache und die Hilfefunktion sind dieselben wie in anderen VBA-aktivierten Anwendungen, z. B. Microsoft Office-Produkten.

Mithilfe der Automatisierung können Sie andere Anwendungen auf zwei Weisen in Rational System Architect integrieren. Sie können Rational System Architect auf die folgenden Weisen verwenden:

- Als Automatisierungssteuereinheit; Sie können ein OLE-Automatisierungsobjekt von einem Rational System Architect-Script aus aufrufen.
- Als Automatisierungsserver; Sie können das zugehörige OLE-Automatisierungsobjekt von einer anderen OLE-kompatiblen Anwendung aus aufrufen.

Makros in Rational System Architect können für Folgendes verwendet werden:

- Erweiterung der Funktionalität von Rational System Architect – Automatisierung der Darstellung und der Regelprüfung von Diagrammen.
- Erstellung von Diagrammen, Symbolen und Definitionen aus Informationen aus anderen Anwendungen.

Erfassen Sie Ereignisse, die in Rational System Architect auftreten, und speichern Sie sie in einer Datei oder in einer Datenbank.

Rational System Architect mit VBA programmieren

Rational System Architect speichert den aktuellen Status der zugehörigen VBA-Projektumgebung in der Initialisierungsdatei SA2001.INI (die entweder im Ordner "Windows" oder "WinNt" enthalten ist). Der Abschnitt **Macros** kann Folgendes enthalten:

```
[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
```

Der oben angezeigte Auszug gibt an, dass die permanente Projektdatei SAAUTO.MAC geladen wird, wenn Rational System Architect startet. (Sie ist ausgeblendet und schreibgeschützt.) Die temporäre Projektdatei SAMPLE.MAC ist derzeit aktiv und wird ebenfalls im schreibgeschützten Modus geladen. Falls eine der temporären Projektdateien das AutoExec-Modul enthält, werden die Makros automatisch ausgeführt. Die Standardeinstellung für das Öffnen neuer Projekte ist "Schreibgeschützt".

Makro ausführen

Gehen Sie wie folgt vor, um Makros direkt im Dialogfenster **Makros** auszuführen:

1. Wählen Sie im Menü **Tools** das Untermenü **Makros** aus.
2. Wählen Sie die Option **Makro ausführen…** aus.

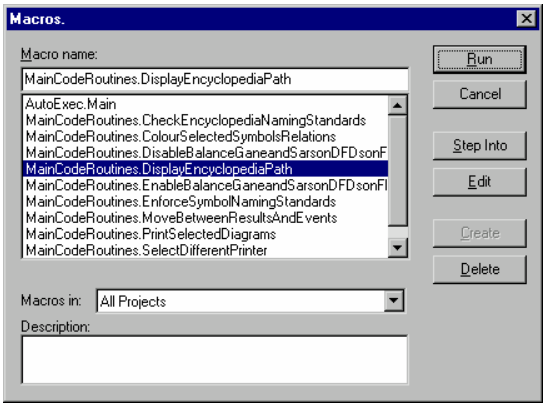


Abbildung 2-1. Dialogfenster "Makros"

Ein Makro kann auch in der Symbolleiste ausgeführt werden.

Gehen Sie wie folgt vor, um ein Makro zu einer Symbolleiste hinzuzufügen:

- Klicken Sie mit der rechten Maustaste auf die Menüleiste oder eine beliebige Symbolleiste und wählen Sie die Option **Anpassen…** aus.
- Wählen Sie die Registerkarte **Befehle** und anschließend die Option **Makros** aus.
- Blättern Sie durch die verfügbaren Makros und **ziehen und übergeben** Sie das erforderliche Makro auf eine Symbolleiste.

Makroprojekte

Andere temporäre Projektdateien können zu dem Musterprojekt hinzugefügt werden.

Gehen Sie wie folgt vor, um auf das Dialogfenster **Makroprojekte** zuzugreifen:

1. Öffnen Sie das Menü **Tools**.
2. Wählen Sie das Untermenü **Makros** aus.
3. Wählen Sie die Option **Makroprojekte…** aus.

Es können neue Projektdateien zur Liste der verfügbaren Projekte hinzugefügt oder vorhandene Projekte entfernt werden.

Das Kontrollkästchen **Automatische Makros aktivieren** kann abgewählt werden, um die AutoExec-Makros zu inaktivieren.

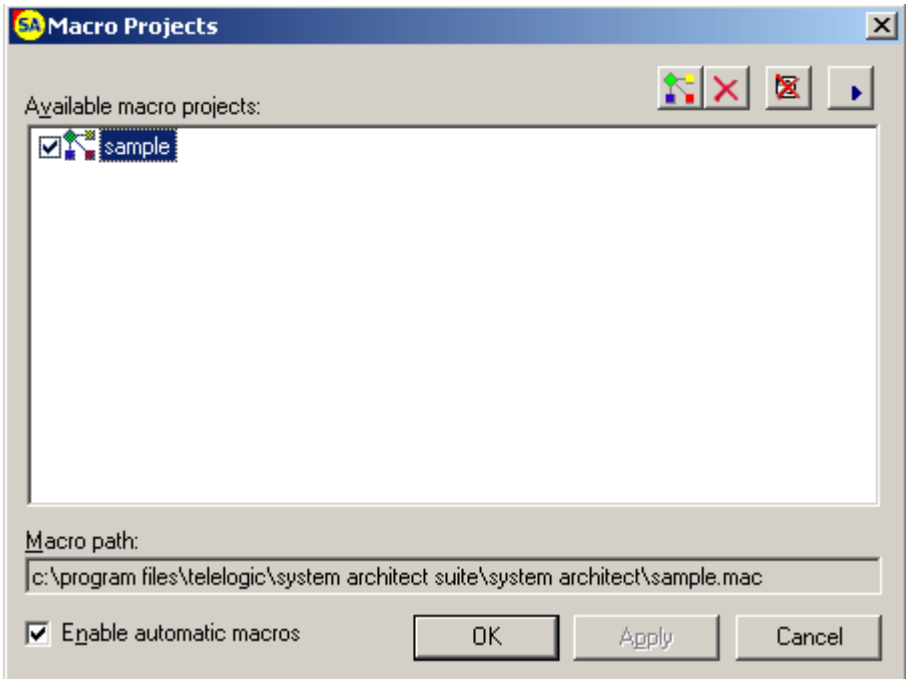


Abbildung 2-2. Dialogfenster "Makroprojekte"

Neues Projekt hinzufügen

Wenn Sie auf die Befehlsschaltfläche **Hinzufügen** klicken, wird das Dialogfenster **Makroprojekt öffnen** anzeigt.

Die Projektdateien von Rational System Architect können anhand der Erweiterung `.MAC` erkannt werden.

Sie können ein vorhandenes Projekt auswählen und öffnen oder ein neues Projekt erstellen, indem Sie einen Dateinamen eingeben, der nicht in der Liste aufgeführt ist.

Wenn Sie Änderungen am Projekt vornehmen müssen, stellen Sie sicher, dass das Kontrollkästchen zum schreibgeschützten Öffnen abgewählt ist, bevor Sie die Projektdatei öffnen.

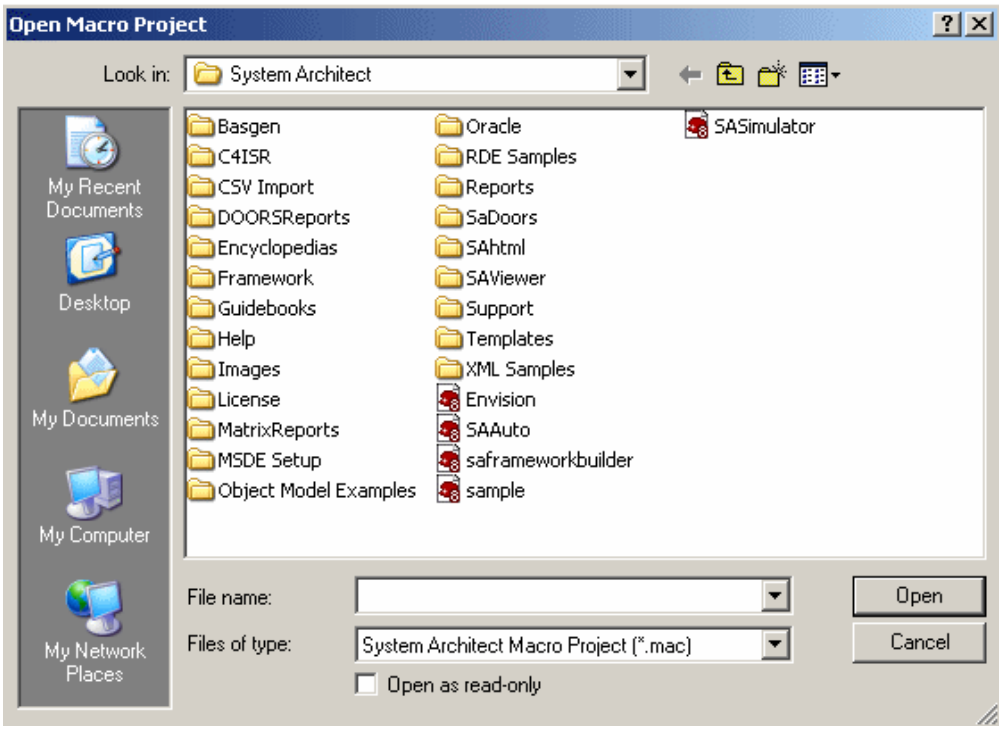


Abbildung 2-3. Dialogfenster "Makroprojekt öffnen"

Auf den VBA-Editor zugreifen

Um ein Makro zu erstellen oder ein vorhandenes Makro zu ändern, müssen Sie den VBA-Editor öffnen.

Sie können den VBA-Editor auf mehrere Arten öffnen:

- Wählen Sie im Menü **Tools** das Untermenü **Makros** und anschließend die Option "VBA-Editor" aus.
- Drücken Sie die Tastenkombination Alt+F11.
- Klicken Sie im Dialogfenster **Makros** auf die Befehlsschaltfläche **Bearbeiten**.
- Klicken Sie im Dialogfenster **Makroprojekte** auf die Schaltfläche zum Anwenden des Befehls (sofern aktiv) und anschließend auf die Befehlsschaltfläche **Makro ausführen…**. Hierdurch wird das Dialogfenster **Makros** geöffnet, das die Befehlsschaltfläche **Bearbeiten** enthält.

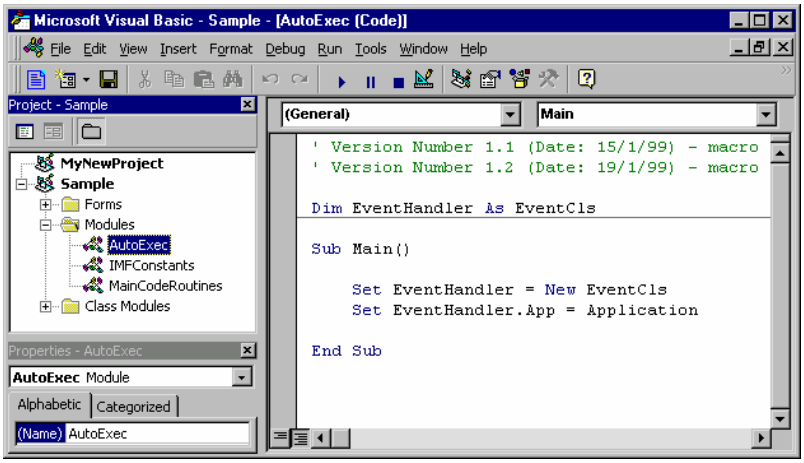


Abbildung 2-4. VBA-Editor

Projektexplorer

Der VBA-Editor besteht aus mehreren Fenstern. Es ist wahrscheinlich, dass das Projektfenster angezeigt wird, wenn Sie zum ersten Mal auf den VBA-Editor zugreifen.

Gehen Sie wie folgt vor, um das Fenster **Projekt** anzuzeigen:

1. Wählen Sie im Menü **Anzeigen** die Option **Projekt**, **Explorer** aus.
2. Das Musterprojekt enthält drei **'Ordnergruppen'**, **Formular**, **Modul** und **Klasse**.
3. Sie können die einzelnen Ordner öffnen, um den jeweiligen Inhalt anzuzeigen, indem Sie auf das 'Plus'-Symbol oder doppelt auf den Ordner oder den Namen des Ordners klicken.

Der Inhalt des jeweiligen Ordners wird in alphabetischer Reihenfolge angezeigt, aber die Ordner können so festgelegt werden, dass eine vollständige alphabetische Liste aller Elemente angezeigt wird.

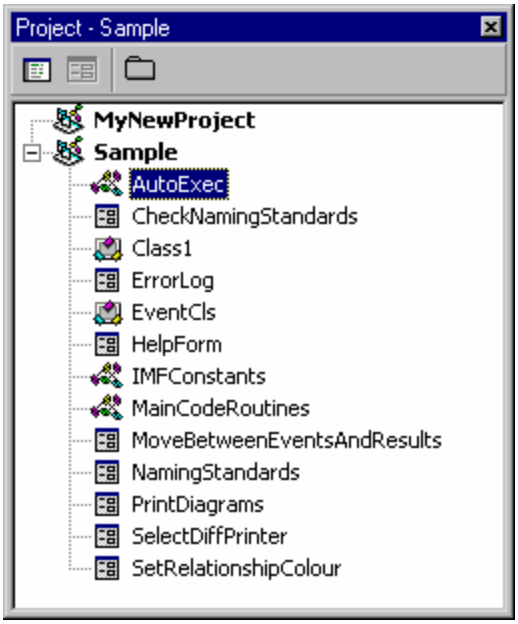


Abbildung 2-5. Projektfenster

Fenster "Eigenschaften"

Wenn Sie auf ein beliebiges **Modul** klicken, zeigt das Fenster **Eigenschaften** den Modulnamen an. Der Name kann dann im Fenster **Eigenschaften** geändert werden.

Wenn Sie doppelt auf einen **Modulnamen** klicken, wird der zugehörige Code angezeigt.

Wenn Sie doppelt auf den Namen eines **Formulars** klicken, wird das **Formularobjekt** angezeigt; das Fenster "Eigenschaften" zeigt eine alphabetische Liste der Formular-eigenschaften an.

Wenn Sie mit der rechten Maustaste auf den Namen eines Formulars klicken, können Sie den **Formularcode anzeigen**.

Das Fenster **Eigenschaften** kann auch nach Kategorie angezeigt werden.

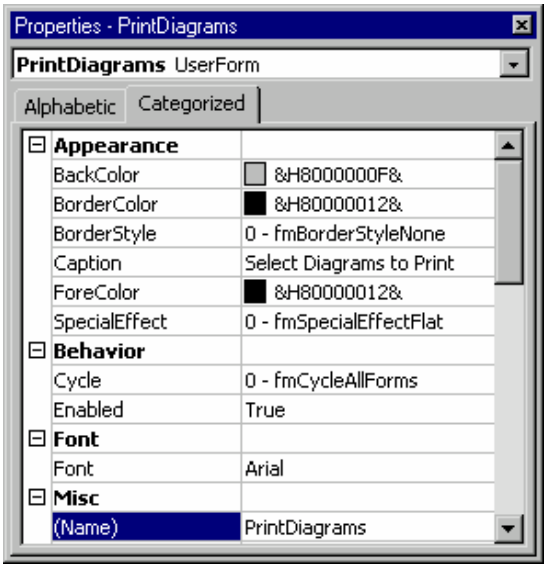


Abbildung 2-6. Fenster "Eigenschaften"

Module und Formulare einfügen

Sie können zusätzliche **Module** und **Formulare** mithilfe des Menüs **Einfügen** zu einem beliebigen Projekt hinzufügen.

Sie können ausgewählte **Module** oder **Formulare** entfernen und optional **exportieren**, indem Sie die Option **Entfernen** im Menü **Datei** verwenden.

Wenn Sie mit der rechten Maustaste auf das Fenster **Projekt** klicken, können Sie auch auf die Optionen **Einfügen** und **Entfernen** zugreifen.

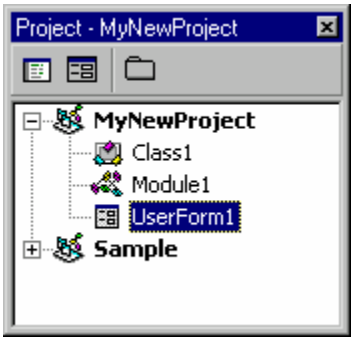


Abbildung 2-7. Projektfenster

Objektbrowser

Der Objektbrowser ist eine sehr nützliche Funktion des VBA-Editors, mit der Sie die verfügbaren Objektbibliotheken, Typenbibliotheken und dynamischen DLLs (Dynamic Link Libraries) abfragen können.

Auf Bibliotheksdateien verweisen

Gehen Sie wie folgt vor, um zusätzliche Bibliotheken für den Objektbrowser verfügbar zu machen:

- Wählen Sie im Menü **Tools** die Option **Verweise…** aus.

Es wird davon ausgegangen, dass die SA2001-Bibliothek bereits in der Liste der ausgewählten Verweise enthalten ist.

Sie können andere Verweise auswählen, wenn Sie in der Liste der verfügbaren Verweise nach unten blättern.

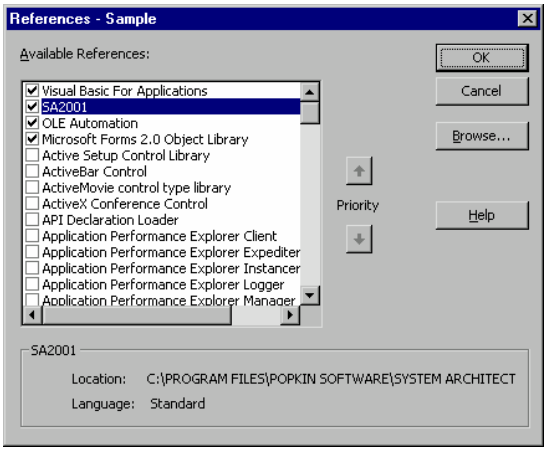


Abbildung 2-8. Verweise auswählen

Drücken Sie die Funktionstaste "F2" oder gehen Sie wie folgt vor, um den Objektbrowser anzuzeigen:

- Wählen Sie im Menü **Anzeigen** die Option für den **Objektbrowser** aus.

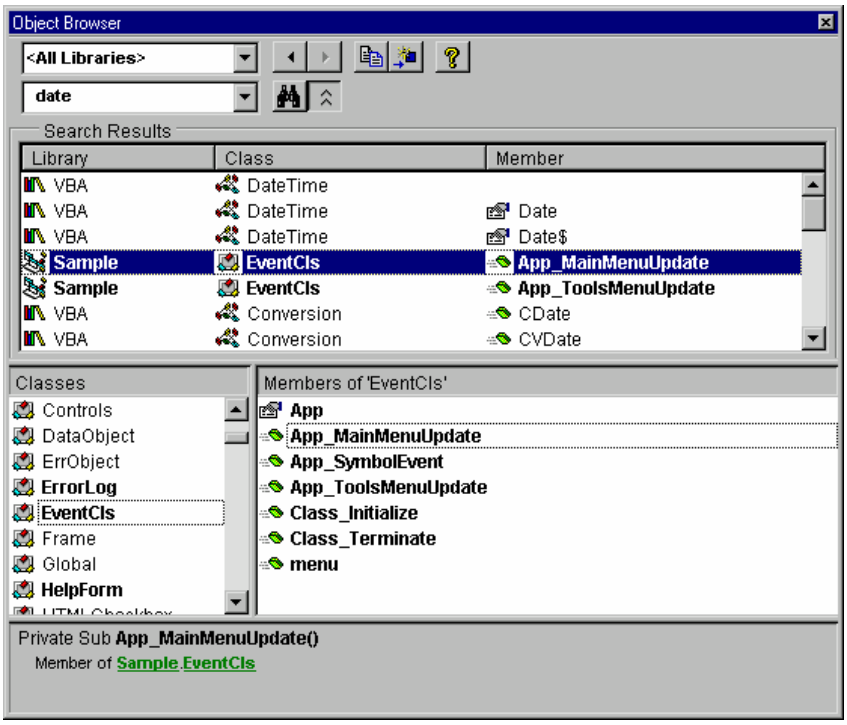


Abbildung 2-9. Objektbrowser

Der Objektbrowser kann für bestimmte Bibliotheken gefiltert und, wie oben dargestellt, zur Suche von Elementen verwendet werden, deren Klassen- oder Teildateinamen einen bestimmten Text enthalten.

Wenn ein Projekt im Projektexplorerfenster ausgewählt wird, markiert der Objektbrowser alle Elemente, die Teil dieses Projekts sind.

Klassen, Konstanten, Aufzählungstypen, Ereignisse, globale Variablen, Methoden, Module, Eigenschaften und benutzerdefinierte Typen werden angezeigt; ihnen wird ein entsprechendes Symbol vorangestellt.

2

Automatisierung und Rational System Architect

Einführung

Die Automatisierung ist die Fähigkeit einer Anwendung, Objekte zu deklarieren und zu verwenden, die von anderen Anwendungen erstellt werden. Wenn Sie Visual Basic for Applications (VBA) verwenden, kann Code geschrieben werden, um auf die Objekte einer oder mehrerer dieser Anwendungsobjekte in einem einzelnen Programm zuzugreifen.

Durch Automatisierung kann eine vollständig integrierte Lösung erstellt werden, die auf der Funktionalität einer Vielzahl von Produkten basiert. In Rational System Architect ist VBA integriert, wodurch es anderen Anwendungen ermöglicht wird, seine Funktionalität zu nutzen. In den folgenden Abschnitten des Handbuchs finden Sie Erläuterungen zu dieser Funktionalität.

In diesem Kapitel finden Sie Beschreibungen zur Automatisierung und Informationen dazu, wie diese Sie beim Bereitstellen einer anpassbaren Lösung unterstützen kann. Anschließend finden Sie ausführliche Informationen dazu, wie Rational System Architect so konfiguriert werden kann, dass es die erforderliche Funktionalität für die Lösung bereitstellt.

In diesem Kapitel enthaltene Themen	Seite
Automatisierung	2-2
Anpassbare Lösungen	2-8
Automatisierte Lösung mit Rational System Architect planen	2-10

Automatisierung

Mithilfe der Automatisierung kann ein Programmierer Objekte von anderen Anwendungen in die aktuelle Anwendung aufnehmen, um eine integrierte Lösung bereitzustellen. Beispiel: Rational System Architect kann zur Erstellung eines Unternehmensdatenmodells verwendet werden, das von einer Anzahl von Benutzern erstellt wird; andere Personen möchten die Daten jedoch mit einem speziell entwickelten Word-Bericht oder einem Excel-Arbeitsblatt anzeigen. Mithilfe der Automatisierung kann ein integrierter Word- oder Excel-Bericht direkt in einem benutzerdefinierten Menü in Rational System Architect ausgeführt werden. Die Automatisierung ermöglicht es dem Benutzer, Tools zu verwenden, die für bestimmte zu integrierende Tasks erstellt wurden.

Die Automatisierung wurde früher auch manchmal als OLE-Automatisierung oder als ActiveX-Automatisierung bezeichnet.

Automatisierungscontroller und -server

Die Automatisierung ist die einzige Möglichkeit zum Steuern anderer Anwendungsobjekte. Technisch gesehen enthält eine Anwendung den VBA-Code, der die Automatisierung steuert, und die andere stellt die Objekte für die Verwendung zur Verfügung. Der Treiber dieses Prozesses wird als Automatisierungscontroller bezeichnet, der Anbieter als Automatisierungserver.

Auf die Typenbibliothek verweisen

VBA verfügt über einen Mechanismus, um zu erkennen, welches Objekt innerhalb eines beliebigen VBA-Programms verfügbar ist. VBA folgert, wenn Sie den Code in Rational System Architect erstellen, dass die zugehörige Bibliothek mit Automatisierungsobjekten verfügbar ist, aber es folgert nicht, dass auch andere Bibliothektstypen verfügbar sind. Der Benutzer muss dann einen Verweis auf die erforderlichen Bibliotheken erstellen.

Bei der **Typenbibliothek** handelt es sich um eine Datenbank, die Informationen zu allen Objekten enthält, die für die Automatisierung in einer beliebigen Anwendung verfügbar sind. Diese Informationen enthalten Details zu Objekten, Attributen, Ereignissen und Methoden, die in der Anwendung verfügbar sind. Dabei handelt es sich in der Regel um eine zur gleichen Zeit wie die Anwendung separat installierte Datei, aber in manchen Fällen kann sie als Teil der ausführbaren Hauptdatei bereitgestellt werden.

Um einen Verweis auf diese Typenbibliotheksinformationen anzugeben, muss der Benutzer den VBA-Editor aufrufen. Dieser kann im entsprechenden Anwendungsmenü oder durch gleichzeitiges Drücken der Tasten **Alt+F11** aufgerufen werden. Wählen Sie im VBA-Editor die Optionen **Tools | Verweise** aus.

In diesem Beispiel sind die einzigen Objekte, auf die verwiesen wird, die VBA- und Rational System Architect-Standardobjekte. Um andere Anwendungen hinzuzufügen, blättern Sie nach unten und wählen Sie die entsprechende Anwendung aus, z. B. Microsoft Excel 9.0-Objektbibliothek, die dann alle Komponenten der Excel-Typenbibliothek in der aktuellen Anwendung enthält.

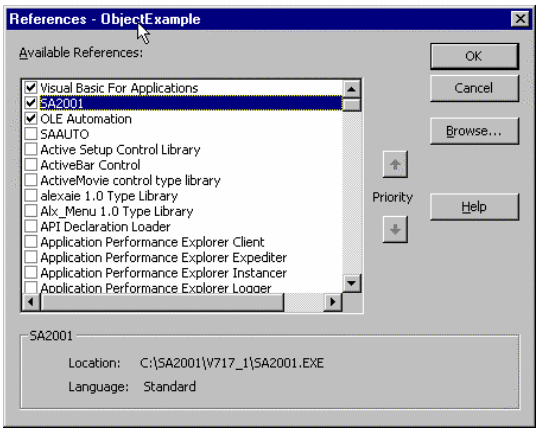


Abbildung 2-1. Typenbibliothek auswählen

Automatisierungsobjekte anzeigen

Sobald ein Verweis auf eine andere Anwendung festgelegt wurde, kann eine Liste der Objekte, Attribute und Methoden aller Typenbibliotheken, auf die derzeit verwiesen wird, angezeigt werden. Der VBA-Editor verfügt über einen eigenen Objektbrowser zum Anzeigen all dieser Eigenschaften.

Um auf den VBA-Objektbrowser zuzugreifen, drücken Sie entweder die Taste F2 oder rufen Sie die Menüoption **Anzeigen** und anschließend die Option für den Objektbrowser im VBA-Editor auf.

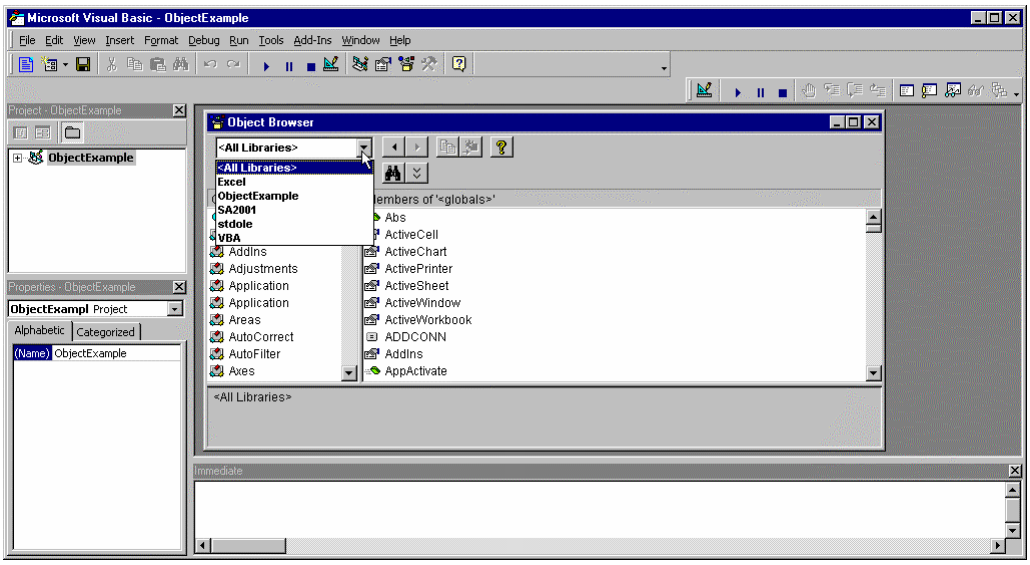


Abbildung 2-2. VBA-Objektbrowser

Der Objektbrowser enthält eine Liste aller Bibliotheken, die in den Typenbibliotheks-verweisen ausgewählt wurden, auch wenn eine einzelne Objektgruppe ausgewählt werden kann. In dem unten angezeigten Beispiel wird "SA2001" als Typenbibliothek ausgewählt, die angezeigt werden soll. Dabei ist speziell die Definitionsklasse der Automatisierungsobjekte von Interesse.

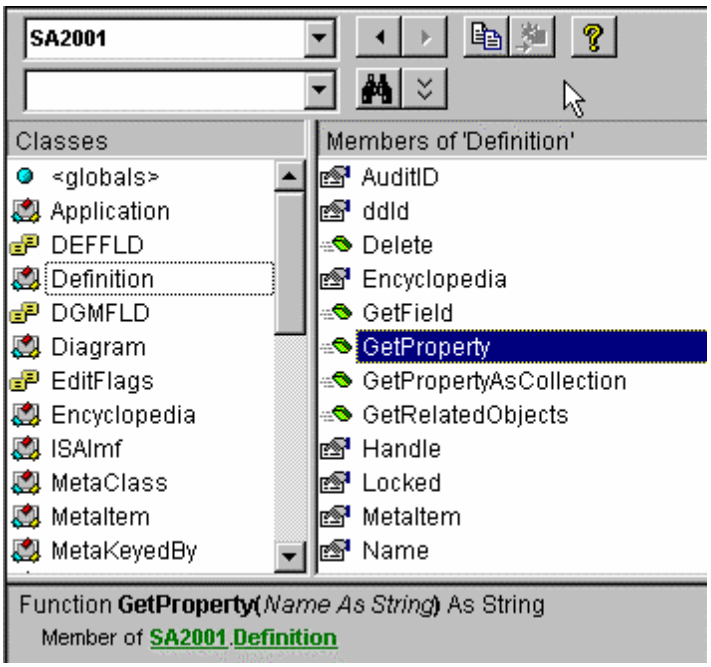


Abbildung 2-3. SA2001-Typenbibliothek durchsuchen

Die Mitglieder der Definitionsklasse sind aufgelistet und **GetProperty** ist ausgewählt. Die Parameter, z. B. Typ und Rückgabetyp, sind im unteren Suchfenster detailliert aufgeführt. Durch die Deklarierung eines Verweises auf ein Automatisierungsobjekt werden all diese internen Anwendungsinformationen für den VBA-Programmierer zum Erstellen einer integrierten Lösung verfügbar.

Instanz der Anwendung erstellen

Auch wenn in VBA auf die erforderlichen Anwendungen verwiesen wird, können die Anwendungsobjekte erst dann gesteuert werden, wenn Code geschrieben wird. Im nächsten Abschnitt enthalten Sie Informationen dazu, wie eine neue Instanz einer Anwendung so konfiguriert werden kann, dass sie auf die zugehörigen Automatisierungsobjekte zugreift, und wie sie deklariert wird.

Eine Instanz ist eine Sitzung der erforderlichen Anwendung. Um Automatisierungsobjekte verwenden zu können, muss die Anwendung im Speicher resident sein (muss jedoch nicht eingeblendet sein). Um auf Objekte zuzugreifen, verwendet VBA die Anweisungen "Dim" und "Set"; sie deklarieren die Instanz eines Automatisierungsobjekts wie bei einer Deklaration in einem integrierten Typ. Der einzige Unterschied bei der Automatisierung ist, dass an

dem Punkt, an dem die Objekte verwendet werden sollen, eine neue Instanz des Objekts mithilfe der Set-Anweisung erstellt werden muss.

Der folgende Code deklariert die Variable *ExObj* als Excel-Anwendung. Beachten Sie den vollständigen server.class-Namen der Deklaration – dieser stellt sicher, dass die Anwendungsklasse, auf die verwiesen wird, die Excel-Anwendungsklasse ist. Andere Anwendungen können ebenfalls über Klassen verfügen, die als "application" bezeichnet werden.

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

Die folgende Zeile erstellt eine neue Instanz der Excel-Anwendung und bestimmt auch den Punkt im Code, an dem die Instanziierung stattfindet.

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

Die Variable *ExObj* kann nun auf die Excel-Automatisierungsobjekte zugreifen, z. B. das Standarddialogfenster **Öffnen** in Excel mithilfe des Automatisierungsobjekts **GetOpenFilename()** gefiltert nach Textdateien mit dem folgenden Code anzeigen.

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

Auf diese Weise werden alle Excel-Automatisierungsobjekte neben den Rational System Architect-Objekten zur Verfügung gestellt.

Anwendungsinstanz freigeben

Automatisierungsobjektinstanzen werden mit dem Befehl "set" erstellt, mit Automatisierungsfunktionsaufrufen bearbeitet und anschließend beendet. Das Beenden der Automatisierungsklasse stellt sicher, dass die Ressourcen aus dem Speicher freigegeben werden.

Der folgende Code sollte verwendet werden, wenn das Automatisierungsobjekt nicht mehr verwendet wird. Dadurch wird der Automatisierungsserver effektiv geschlossen.

```
Set Exobj = Nothing
```

Zusammenfassung

Dim Object = Server.class	Server deklarieren
Set Object = New Server.class	Instanziierung des Servers
Automation Code	VBA-Code, der auf Server-automatisierungsobjekte zugreift
Dim Object = Nothing	Automatisierungsserver aus dem Speicher freigeben

Anpassbare Lösungen

Im vorherigen Abschnitt erhielten Sie eine Einführung in die Konzepte der Automatisierung, in der beschrieben wurde, wie Objekte einiger Anwendungen in andere integriert werden können. Alle Produkte, die VBA und die Automatisierung unterstützen, sind im Prinzip anpassbare Lösungen. Das heißt, dass jedes Produkt, das VBA unterstützt, beliebig geändert werden kann. Das gilt jedoch nicht nur für die Integration in andere Anwendungen. Ein Großteil der Arbeit bei einer Anpassung einer Anwendung wird auf das Ändern der eigentlichen Funktionsweise des Produkts verwendet.

Verschiedene Benutzer eines Produkts haben möglicherweise unterschiedliche Vorstellungen von einer Anpassung – Menüoptionen ändern, Integration in Office-Produkte, Automatisierung sich wiederholender Tasks – und all diese Vorstellungen können durch die Verwendung von VBA und der Automatisierung umgesetzt werden. In der folgenden Tabelle sind fünf potenzielle Gründe für die Anpassung eines Produkts aufgeführt. Sie können mithilfe von VBA alle umgesetzt werden.

Kategorie	Was bedeutet das?	Rational System Architect-Beispiel
Änderung des Anwendungs-verhaltens	Änderung der Funktions-weise von Anwendungen, um diese an die Geschäfts-regeln und -prozesse eines Unternehmens anzupassen	Beim Erstellen eines Symbols in einem Diagramm werden die Standardbenennungsregeln des Unternehmens überprüft und der Benutzer mit einem Flag versehen, wenn diese nicht eingehalten werden.
Automatisierung von sich wiederholenden Tasks	Kombination mehrerer allgemeiner manueller Tasks in einer Abfolge von Aktionen, die immer wieder ausgeführt werden können	Drucken einer benutzer-definierten Diagrammabfolge.
Erweiterung der Anwendungs-funktionalität	Hinzufügen von Funktionen zu einer Anwendung, die ohne Vorbereitungs- oder Anpassungsaufwand nicht verfügbar sind	Automatische Erstellung eines Prozesszuordnungsdiagramms, das von einer Prozessgrafik sowie von den zugehörigen Rollen und Organisations-einheiten abgeleitet ist.

Kategorie	Was bedeutet das?	Rational System Architect-Beispiel
Integration in andere Anwendungen	Steuerung einer anderen Anwendung, um die Funktionalität zu nutzen, die in der Regel nicht verfügbar ist	Erstellung eines Excel-Arbeitsblatts mit Informationen zu Prozessen im Vergleich zu Entitäten-CRUD.
Zugriff auf Unternehmensdaten	Austausch von Daten mit fernen Datenbanken und Anwendungen, die in der Regel nicht über Datenbankzugriff verfügen	Automatisierter Import von Informationen aus verschiedenen Quellen, die in der Regel nicht direkt importiert werden können.

Die Kombination von Rational System Architect mit VBA stellt dem Benutzer eine Standard-IDE (Integrated Development Environment - integrierte Entwicklungsumgebung) zur Erstellung dieser anpassbaren Lösungen zur Verfügung, indem alle Programmierungsentwicklungen von Microsoft integriert werden, z. B. IntelliSense und Microsoft-Formulare.

Automatisierte Lösung mit Rational System Architect planen

Die Verwendungsmöglichkeiten der Automatisierung reichen von der Verringerung der Anzahl der sich wiederholenden Tasks bis zum Erstellen einer vollständig integrierten Anwendung. Im nächsten Abschnitt erhalten Sie Informationen zu den Änderungen, die Sie an Rational System Architect vornehmen können.

Steuerungsverhalten

Rational System Architect reagiert auf bestimmte Ereignisse, die während des Betriebs auftreten. Diese Ereignisse können Code auslösen, der zum Automatisieren bestimmter Tasks verwendet werden kann.

Die von Rational System Architect unterstützten Ereignisse umfassen den Systemstart und den Systemabschluss des Produkts, das Öffnen und Schließen der Enzyklopädie, das Öffnen von Diagrammen, das Speichern und Schließen, das Ändern von Prüf-IDs sowie eine Anzahl von Symbolereignissen. Zu den Symbolereignissen gehören das Platzieren im Diagramm, das Benennen, das Verbinden und das Trennen einer Verbindung (mit einem Liniensymbol) sowie das Löschen.

Diese Funktionalität ermöglicht einen Echtzeitbetrieb von Rational System Architect, der mithilfe von VBA geändert werden kann.

Darstellung steuern

Ab Rational System Architect 10.1 können die Benutzer Menüs in Rational System Architect anpassen und Befehle aus der vorhandenen Menüstruktur entfernen oder hinzufügen. Ein Benutzer kann z. B. einen Befehl zum Ausführen eines Makros verwenden. Vor Rational System Architect 10.1 (z. B. 10.0 und davor) konnten Befehle nur über Rational System Architect-VBA zu Menüoptionen hinzugefügt werden. Beispiel: Ein Menü, das Optionen zu einer bestimmten Methodik enthält, konnte während der Codeausführung aktualisiert werden. Diese Menüoptionen wurden dann nur für ein bestimmtes Diagramm angezeigt, abhängig von der ausgewählten Methodik.

In jedem Fall können Menüoptionen und Kontextmenüs zu Menüs hinzugefügt und Bitmaps diesen Optionen zugeordnet werden. Eine häufige Verwendungsmöglichkeit dieser Technik ist das Hinzufügen eines Makros zu einer Menüoption und das Bereitstellen einer Bitmap, die das Makro darstellt. Diese kann dann sowohl in einer Symbolleiste als auch im Menü platziert werden.

Informationen zum Anpassen von Makros für Rational System Architect 10.1, die für die Verwendung in Menüoptionen von Rational System Architect bis Version 10.0 erstellt wurden, finden Sie im Konvertierungshandbuch auf der IBM Unterstützungswebsite.

Tasks automatisieren

Ein VBA-Makro kann zur Berichterstellung zu Informationen im Repository und zur Umsetzung von Konsistenzprüfungen erstellt werden. Alle Objekte und Eigenschaften im Datenverzeichnis können, basierend auf den Regeln, die im Code festgelegt sind, erstellt, gelesen, aktualisiert oder gelöscht werden. Einfache, sich wiederholende Tasks, z. B. das Aktualisieren eines Werts eines Verzeichnisobjekts in einem anderen Verzeichnisobjekt, können somit automatisiert werden.

Steuerelemente erzwingen

Durch Verwendung der ereignisgesteuerten Mitglieder des Rational System Architect-VBA-Modells können vordefinierte Standards in Echtzeit auf das Modell angewendet werden. Dabei kann es sich um einen Benennungsstandard oder um das Ausfüllen eines obligatorischen Felds handeln.

Schnittstelle zwischen externen Anwendungen

VBA kann, basierend auf Werten in anderen Anwendungen, zum Importieren, Exportieren, Lesen, Erstellen, Ändern, Aktualisieren und Löschen von Modellobjekten verwendet werden. Beispielfunktionen umfassen das Erstellen von Diagrammen, Symbolen und Definitionen, das Erstellen von Beziehungen zwischen Symbolen und das Ändern der Datenverzeichniseigenschaften.

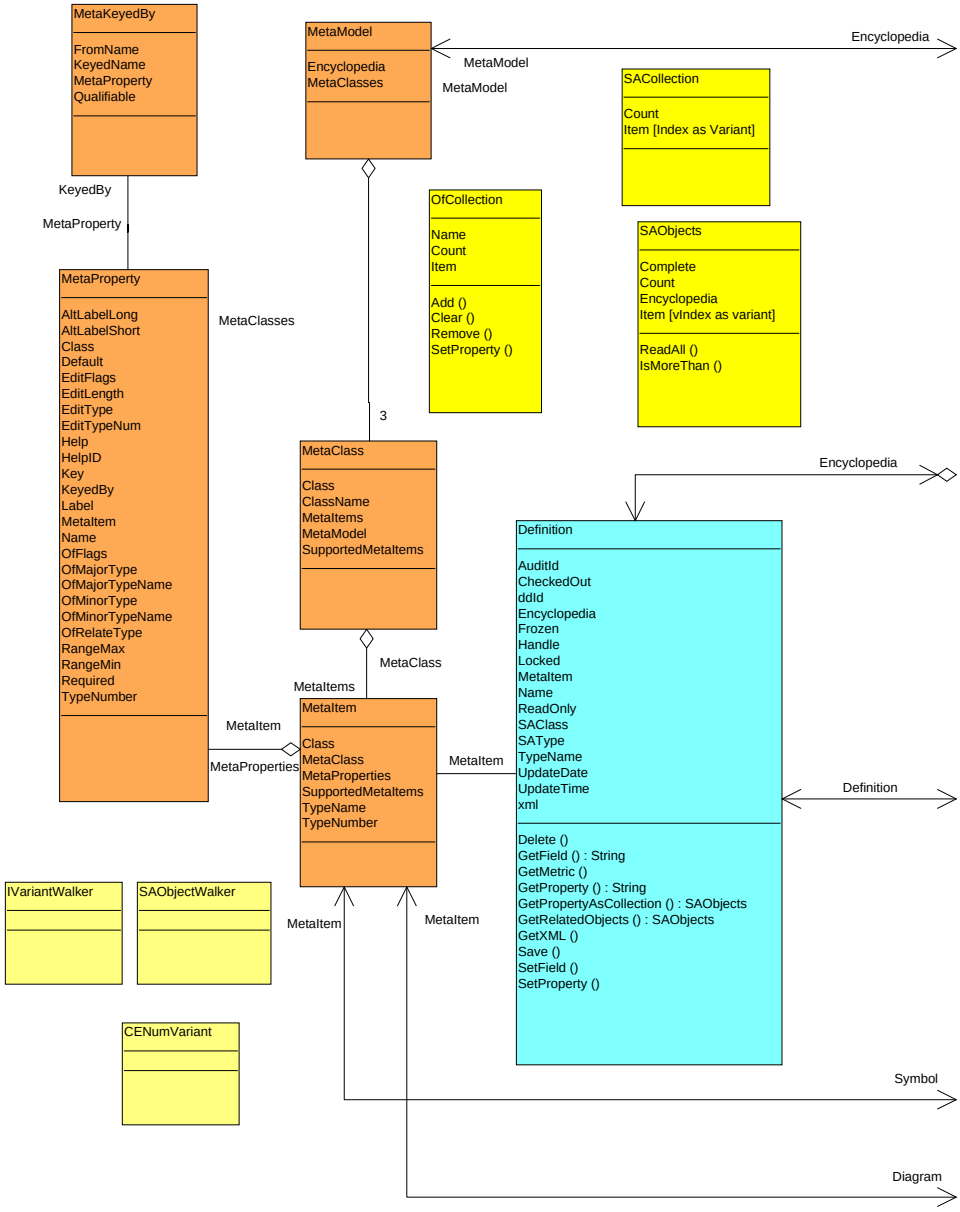
3

Rational System Architect-Objektmodell

Die für VBA in Rational System Architect verfügbaren Objekte können im Objektbrowser im VBA-Editor angezeigt werden. Die einzelnen Objekttypen werden als Klasse definiert, die eine Liste der unterstützten Eigenschaften und Methoden enthält.

Es ist einfacher, das gesamte Modell anzuzeigen, indem Sie ein Klassendiagramm verwenden. Die folgende Abbildung zeigt einen Auszug aus dem Rational System Architect-Objektmodell an, das in Rational System Architect mithilfe der UML-Klassendiagrammnotation gezeichnet wurde.

In diesem Kapitel enthaltene Themen	Seite
Rational System Architect-Objektmodell	3-2
Objektmodellklassen	3-4


MetaKeyedBy
FromName
KeyedName
MetaProperty
Qualifiable
KeyedBy
MetaProperty
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
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 ()
3
MetaClass
Class
ClassName
MetaItems
MetaModel
SupportedMetaItems
MetaClass
MetaItems
MetaItem
Class
MetaClass
MetaProperties
SupportedMetaItems
TypeName
TypeNumber
MetaItem
MetaProperties
MetaItem
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 ()
Definition
IVariantWalker
SAObjectWalker
MetaItem
MetaItem
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 ()
MetaModel
Encyclopedia
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

Objektmodellklassen

Im Folgenden finden Sie die Rational System Architect-Objektmodellklassen mit jeweils verschiedenen Verwendungsbeispielen.

Klasse	Verwendungsbeispiele
Application	Rational System Architect-Ereignisse Menübearbeitung
Encyclopedia	Diagramme und Definitionsobjekte erstellen Diagramm- und Definitionsobjekte abrufen
Diagram	Symbolobjekte erstellen und abrufen Diagrammeigenschaftenbearbeitung
Symbol	Symboleigenschaftenbearbeitung Symbolverbindungsinformationen Untergeordnete Diagrammobjekte abrufen
Definition	Definitionseigenschaftenbearbeitung Bearbeitung zugehöriger Definitionen
SACollection	Gruppe von Rational System Architect-Eigenschaften
SAObjects	Gruppe von Rational System Architect-Objekten (Diagramme, Symbole und Definitionen)
OfCollection	Sammlung von OneOf- oder ListOf-Diagrammen oder -Definitionen.
MetaModel	Rational System Architect-

Klasse	Verwendungsbeispiele
MetaClass MetaItem	Metamodellobjekte (unterstützte Diagramm-, Symbol- und Definitionstypen)
MetaKeyedBy	Nach Definitionen verschlüsseltes Rational System Architect-Metamodell (verschlüsselt nach Definitionen und den zugehörigen Strukturen)
MetaProperty	Rational System Architect-Metamodelleigenschaftengruppen (unterstützte Eigenschaften und deren Strukturen innerhalb der jeweiligen Definitionstypen)

4

Anwendungsklasse

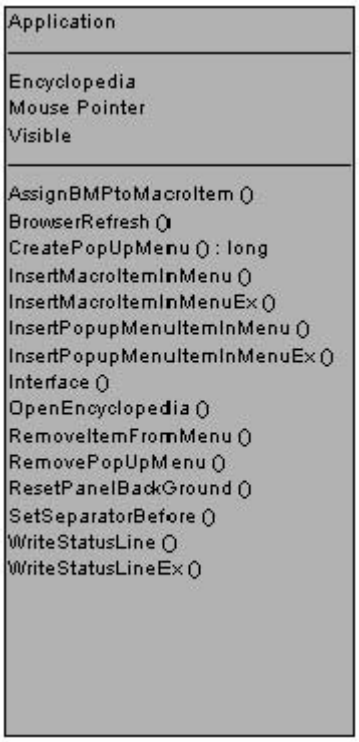
In diesem Kapitel enthaltene Themen	Seite
Attribute	4-3
Methoden	4-5

Einführung

Hierbei handelt es sich um das Rational System Architect-Anwendungsobjekt, mit dem Sie die Benutzerschnittstelle steuern können. Dabei handelt es sich um die höchste Ebene im Objektmodell.

Eine Instanz des Anwendungsobjekts wird wie folgt instanziiert.

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



Attribute

Encyclopedia

Zweck
Hierbei handelt es sich um das Enzyklopädieobjekt, das den Zugriff auf die Enzyklopädieklassenattribute und -methoden erleichtert.

Parameter
Schreibgeschützt

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

MousePointer

Zweck
Damit kann der Benutzer den Mauszeigertyp steuern, der dem Benutzer bei der Verwendung der Anwendung angezeigt wird.

Parameter
Datentyp: Ganzzahl

Beispiel
Der derzeitige Wert des Mauszeigers kann wie folgt zurückgegeben werden:

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

Um den Mauszeiger als Typ “HourGlass” festzulegen, geben Sie "11" als Wert ein.

```
oApplication.Mousepointer  = 11
```

Eine vollständige Liste der gültigen Werte ist in den VBA-Hilfedateien enthalten.

Visible

Zweck
Bestimmt, ob die Anwendung ausgeführt wird. Wenn Sie für diesen Wert “False” (Falsch) festlegen, wird die Anwendung heruntergefahren.

Parameter
Datentyp: boolesch

Beispiel
```
oApplication.Visible = False
```

Methoden

AssignBMPtoMacroItem

Zweck
Verbindet eine Bitmap, z. B. ein Menüsymbol, mit einem VBA-Makro.

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`MacroName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Beliebiges gültiges Makroprojekt

Syntax: “Projektname, Modulname, Subroutinenname()”

`BMPFileName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Dateipfad und Dateiname der BMP (z. B. “C:\Windows\world.bmp”)

BrowserRefresh

Zweck
Aktualisiert den Rational System Architect-Browser. Dies erzwingt die Anzeige aller Elemente, die seit der letzten Aktualisierung zur Enzyklopädie hinzugefügt wurden.

Syntax

```
Application Object.BrowserRefresh
```

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

CreatePopUpMenu

Zweck

Erstellt ein Kontextmenü, das in die Benutzerschnittstelle eingefügt werden kann. Ein Bitmapsymbol kann mit dem Kontextmenü verbunden werden.

Syntax

```
Application Object.CreatePopUpMenuPopUpName[, BMPFileName]
```

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`PopUpName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des erstellten Kontextmenüs

`BMPFileName`

Verwendung: optional

Datentyp: Zeichenfolge

Dateipfad und Dateiname der BMP (z. B. “C:\Windows\world.bmp”)

InsertMacroItemInMenu

Zweck

Erstellt eine Menüoption im Rational System Architect-Menü, die auf eine vorhandene Makrosubroutine verweist.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`MacroName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Beliebiges gültiges Makroprojekt

Syntax: “Projektname, Modulname, Subroutinenname()”

`MacroItemCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des Makroelements, das in ein SA-Menü eingefügt wird

`InMenuTitleCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des vorhandenen SA-Kontextmenüs, in dem das Makroelement platziert wird

`BeforeMenuItemCaption`

Verwendung: optional

Datentyp: Zeichenfolge

Name der vorhandenen SA-Menüoption, vor der das Makroelement platziert wird

Hinweis: Bei fehlender Angabe wird das Makro unten im Kontextmenü platziert.

InsertMacroItemInMenuEx

Zweck

Erstellt eine Menüoption im Rational System Architect-Menü, die auf eine vorhandene Makrosubroutine verweist. Diese Methode ist eine Erweiterung der InsertMacroItemInMenu-Methode.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`MacroName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Beliebiges gültiges Makroprojekt

Syntax: “Projektname, Modulname, Subroutinenname()”

`MacroItemCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des Makroelements, das in ein SA-Menü eingefügt wird

`InMenuTitleCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des vorhandenen SA-Kontextmenüs, in dem das Makroelement platziert wird

`BeforeMenuItemCaption`

Verwendung: optional

Datentyp: Zeichenfolge

Name der vorhandenen SA-Menüoption, vor der das Makroelement platziert wird

Hinweis: Bei fehlender Angabe wird das Makro unten im Kontextmenü platziert.

`Tag`

Verwendung: optional

Datentyp: Zeichenfolge

Dies ermöglicht es mehreren Menüoptionen, über denselben Namen zu verfügen und auf dieselbe Subroutine zu verweisen, indem alle Menüoptionen mit einem eindeutigen Tag versehen werden. Wenn eine der Menüoptionen aufgerufen wird, teilt der Tag der Subroutine mit, welcher Teil des Codes ausgeführt werden soll. Beispiel: Ein Benutzer kann ein Makro schreiben, das verschiedene Word-Berichte erstellt. Anstatt separate Subroutinen für die einzelnen Typen von Word-Berichten schreiben zu müssen, kann der Benutzer den Code in eine Subroutine schreiben und verschiedene Tags in den jeweiligen Menüoptionen angeben, die auf verschiedene Funktionen im Code verweisen.

`bAfterSepartor`

Verwendung: optional

Datentyp: boolesch

Wird nur verwendet, wenn die vorhandene Menüoption, vor der der Benutzer das Makro platziert, davor über eine Trennlinie verfügt. Wenn "True" (Wahr) eingegeben ist, dann wird das Makroelement nach der Trennlinie platziert. Wenn "False" (Falsch) oder nichts eingegeben ist, dann wird das Makroelement automatisch vor der Trennlinie platziert.

InsertPopUpMenuItemInMenu

Zweck

Erstellt eine Kontextmenüoption in einer vorhandenen Rational System Architect-Menüoption.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`PopUpName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des erstellten Kontextmenüs

`InMenuTitleCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des vorhandenen SA-Kontextmenüs, in dem ein neues Kontextmenü platziert wird

`BeforeMenuItemCaption`

Verwendung: optional

Datentyp: Zeichenfolge

Name der vorhandenen SA-Menüoption, vor der das neue Kontextmenü platziert wird

Hinweis: Bei fehlender Angabe wird das neue Kontextmenü unten im vorhandenen Kontextmenü platziert.

InsertPopUpMenuItemInMenuEx

Zweck

Erstellt eine Kontextmenüoption in einer vorhandenen Rational System Architect-Menüoption. Diese Methode ist eine Erweiterung der InsertPopUpMenuItemInMenu-Methode.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`PopUpName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des erstellten Kontextmenüs

`InMenuTitleCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des vorhandenen SA-Kontextmenüs, in dem ein neues Kontextmenü platziert wird

`BeforeMenuItemCaption`

Verwendung: optional

Datentyp: Zeichenfolge

Name der vorhandenen SA-Menüoption, vor der das neue Kontextmenü platziert wird

Hinweis: Bei fehlender Angabe wird das neue Kontextmenü unten im vorhandenen Kontextmenü platziert.

`bAfterSepartor`

Verwendung: optional

Datentyp: boolesch

Wird nur verwendet, wenn die vorhandene Menüoption, vor der der Benutzer das Makro platziert, davor über eine Trennlinie verfügt. Wenn "True" (Wahr) eingegeben ist, dann wird das Makroelement nach der Trennlinie platziert. Wenn "False" (Falsch) oder nichts eingegeben ist, dann wird das Makroelement automatisch vor der Trennlinie platziert.

Schnittstelle

Zweck
Diese Methode wird selten verwendet, kann jedoch eine Instanz einer Schnittstelle mithilfe einer Zeichenfolge anstatt eines expliziten Verweises aufrufen.

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

OpenEncyclopedia

Zweck
Öffnet eine vorhandene Rational System Architect-Enzyklopädie.

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`EncyclopediaPath`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Dateipfad einer vorhandenen Enzyklopädie

Bei (EncyclopediaPath) handelt es sich um eine UDL-Datei. UDL-Dateien werden indirekt im folgenden Pfad erstellt: C:\Dokumente und Einstellungen\<Benutzername>\Local Settings\Application Data\Telelogic\System Architect\Temp UDL files. Diese UDL-Dateien können z. B. wie folgt benannt sein: SA_563.udl. Rational System Architect muss geöffnet sein, damit Sie diesen Pfad anzeigen können.

OpenEncyclopediaUsingConnectionString

Zweck
Öffnet eine vorhandene Rational System Architect-Enzyklopädie mithilfe einer Verbindungszeichenfolge.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`strConnection`

Verwendung: erforderlich

Datentyp: Zeichenfolge

```
strConnection ist eine Zeichenfolge, die den Inhalt der
  UDL-Datei darstellt.
```

`Beispiel:`

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

OpenEncyclopediaUsingDisplayName

Zweck
Öffnet eine vorhandene Rational System Architect-Enzyklopädie mithilfe des Anzeigenamens.

Syntax
Application Object.OpenEncyclopediaUsingDisplayName(strDisplayName)

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`strDisplayName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

strDisplayName ist der Name, so wie er in der SA-Beschriftungsleiste angezeigt wird, z. B. connection-name(encyc-name).

Beispiel:

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

RemoveItemFromMenu

Zweck

Diese Methode entfernt eine Menüoption aus einer benannten Rational System Architect-Menüoption oder einem Rational System Architect-Kontextmenü.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`ItemCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name der Menüoption, die aus einem vorhandenen Kontextmenü entfernt wird

`FromMenuTitleCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des vorhandenen Kontextmenüs, aus dem die Menüoption entfernt wird

RemovePopUpMenu

Zweck

Das angegebene Kontextmenü wird aus dem Rational System Architect-Menüsystem entfernt.

Syntax

`Application Object.`**`RemovePopUpMenu`**`(PopUpName)`

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`PopUpName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des Kontextmenüs, das entfernt wird

ResetPanelBackGround

Zweck

Setzt die Hintergrundfarbe der Statusleistenanzeige zurück.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`Panel`

Verwendung: erforderlich

Datentyp: lang

Anzeigen sind die "Teilfenster" oder Abschnitte der Statusleiste - 1 auf der linken bis 4 auf der rechten Seite (2 und 3 werden nur angezeigt, wenn ein Symbol ausgewählt ist).

SetSeparatorBefore

Zweck

Platziert in einem angegebenen Menü eine Trennleiste vor einer Menüoption.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`ItemCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name der Menüoption, vor der die Trennleiste platziert wird.

`FromMenuTitleCaption`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des vorhandenen SA-Kontextmenüs, in dem das Trennzeichen platziert wird.

`bHasSeparator`

Verwendung: erforderlich

Datentyp: boolesch

Legt mit dem Wert "True" (Wahr) oder "False" (Falsch) fest, ob das Trennzeichen angezeigt wird.

WriteStatusLine

Zweck

Ermöglicht ein Relais von Kurznachrichten an den Benutzer, um ihn zu informieren, während Code in der Statusleiste von Rational System Architect ausgeführt wird (Leiste in der unteren linken Ecke).

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`TextToShow`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Text, der in der Statusleiste angezeigt wird.

WriteStatusLineEx

Zweck

Ermöglicht ein Relais von Kurznachrichten an den Benutzer, um ihn zu informieren, während Code in der Statusleiste von Rational System Architect ausgeführt wird (Leiste in der unteren linken Ecke). Diese ist eine Erweiterung der WriteStatusLin-Methode.

Syntax

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

`Application Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Anwendungsklasse

`Panel`

Verwendung: erforderlich

Datentyp: lang

Anzeigen sind die "Teilfenster" oder Abschnitte der Statusleiste - 1 auf der linken bis 4 auf der rechten Seite (2 und 3 werden nur angezeigt, wenn ein Symbol ausgewählt ist).

`TextToShow`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Text, der in der Statusleiste angezeigt wird.

`BackColor`

Verwendung: erforderlich

Datentyp: lang

Hintergrundfarbe der Statusleiste

`ForeColor`

Verwendung: erforderlich

Datentyp: lang

Vordergrundfarbe der Statusleiste

5

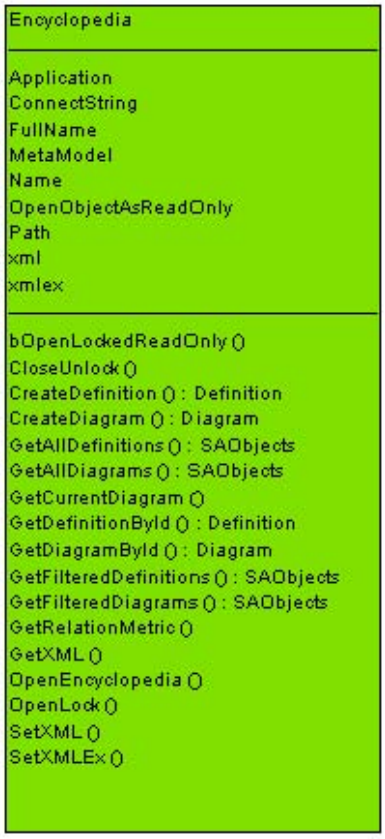
Die Enzyklopädieklasse

Themen in diesem Kapitel	Seite
Attribute	5-3
Methoden	5-6
Arbeitsbereichsmethoden	5-21
Aufrufe für Arbeitsbereichsanwendungsklassen	5-23
Ereignisse für Arbeitsbereichsanwendungsklassen	5-24
Beziehungsmessdaten	5-25

Einführung

Hierbei handelt es sich um das Enzyklopädieobjekt, das den Zugriff auf die Enzyklopädie-attribute und -methoden ermöglicht, wie unten dargestellt.

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



Attribute

Application

Zweck
Das Anwendungsobjekt gibt das übergeordnete Anwendungsobjekt des aktuellen Enzyklopädieobjekts zurück.

Parameter
Schreibgeschützt

ConnectString

Zweck
Die zum Herstellen der Verbindung zu einer Enzyklopädie erforderlichen Informationen.

Parameter:
Datentyp: Zeichenfolge

Schreibgeschützt

FullName

Zweck
Der Name der aktuellen Enzyklopädie, einschließlich des vollständigen Pfads.

Parameter
Schreibgeschützt

Datentyp: Zeichenfolge

MetaModel

Zweck
Hierbei handelt es sich um die MetaModel-Klasse, die den Zugriff auf alle MetaModel-Attribute ermöglicht.

Parameter
Schreibgeschützt

Name

Zweck
Gibt den Namen der aktuellen Enzyklopädie zurück.

Parameter
Schreibgeschützt

Datentyp: Zeichenfolge

OpenObjectsAsReadOnly

Zweck
Legt fest, ob alle Objekte aus dem SA-Objektmodell schreibgeschützt geöffnet werden müssen.

Parameter
Datentyp: boolesch

Path

Zweck
Dies ist der Pfad der aktuellen Enzyklopädie.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

Xml

Zweck
Die XML-Zeichenfolge der Enzyklopädie. Bearbeitet durch die GetXML- und die SetXML-Methode.

Parameter
Datentyp: Zeichenfolge

XmlEx

Zweck
Die XML-Zeichenfolge der Enzyklopädie. Bearbeitet durch die SetXMLEx-Methode.

Parameter
Datentyp: Zeichenfolge

Methoden

bOpenLockedReadOnly

Zweck
Durch diese Methode wird "True" zurückgegeben, wenn für das OpenObjectsAsReadOnly-Attribut der Wert "True" festgelegt wurde oder wenn die Enzyklopädie schreibgeschützt geöffnet wurde.

CloseUnlock

Siehe unten den Eintrag zu "OpenLock".

CreateDefinition

Zweck
Erstellt eine Instanz der Definitionsklasse mit einem angegebenen Definitionsnamen und -typ.

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name der neuen Definition

`SAType`

Verwendung: erforderlich

Datentyp: lang

Typ der Rational System Architect-Definition, die erstellt wird (z. B. DFXPROCESS oder 3)

Anmerkung: Eine vollständige Liste aller SA-Definitionen und der zugehörigen internen Konstantennamen und -nummern finden Sie in der Datei DEFNS.BAS im Rational System Architect-Verzeichnis.

Anmerkung: Um erfolgreich eine SA-Definition zu erstellen, müssen Sie die Save-Methode der Definition aufrufen. Andernfalls wird die neue Definition beim Schließen der Enzyklopädie gelöscht.

CreateDiagram

Zweck

Erstellt eine Instanz der Diagrammklasse mit einem angegebenen Diagrammnamen und -typ.

Syntax

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des neuen Diagramms

`SAType`

Verwendung: erforderlich

Datentyp: lang

Typ des Rational System Architect-Diagramms, das erstellt wird (z. B. GTCATPROCESSFLOW oder 89)

Anmerkung: Eine vollständige Liste aller SA-Diagramme und der zugehörigen internen Konstantennamen und -nummern finden Sie in der Datei DIAGRAMS.BAS im Rational System Architect-Verzeichnis.

GetAllDefinitions

Zweck

Dadurch werden alle Definitionen in der Enzyklopädie als Objektgruppe von Definitionen zurückgegeben.

Regeln

Eine SAObjects-Variable muss als Objektgruppe von Definitionen dimensioniert und festgelegt sein.

Beispiel

```
Dim oCollectionofDefinitions As SAObjects

Set oCollectionofDefinitions =
  oEncyclopedia.GetAllDefinitions

Call oCollectionofDefinitions.ReadAll
```

Die SAObjects-Objektgruppe wird erst dann vollständig ausgefüllt, wenn das Flag "Complete" für die Objektgruppe "True" ist. "GetAllDefinitions" sollte in Verbindung mit der ReadAll- oder der IsMoreThan-Methode verwendet werden.

GetAllDiagrams

Zweck
Dadurch werden alle Diagramme in der Enzyklopädie als Objektgruppe von Diagrammen zurückgegeben.

Regeln
Eine SAObjects-Variable muss als Objektgruppe von Diagrammen dimensioniert und festgelegt sein.

Beispiel
```
Dim oCollectionofDiagrams As SAObjects

Set oCollectionofDiagrams =
  oEncyclopedia.GetAllDiagrams

Call oCollectionofDiagrams.ReadAll
```

Die SAObjects-Objektgruppe wird erst dann vollständig ausgefüllt, wenn das Flag "Complete" für die Objektgruppe "True" ist. "GetAllDiagrams" sollte in Verbindung mit der ReadAll- oder der IsMoreThan-Methode verwendet werden.

GetCurrentDiagram

Zweck
Durch diese Methode wird das derzeit geöffnete Diagramm als Diagrammobjekt zurückgegeben.

Regeln
Ein Diagrammobjekt muss als das derzeit geöffnete Diagramm dimensioniert und festgelegt sein. Siehe folgendes Beispiel.

Beispiel
```
Dim OCurrentDiagram As Diagram

Set OCurrentDiagram = oEncyclopedia.GetCurrentDiagram
```

GetDefinitionById

Zweck
Durch diese Methode wird eine Definition als Definitionsobjekt von der zugehörigen angegebenen ID zurückgegeben.

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`Id`

Verwendung: erforderlich

Datentyp: lang

Alle in Rational System Architect gespeicherten Definitionen sind intern eindeutig durch die Verwendung einer Datenverzeichnis-ID gekennzeichnet.

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

GetDiagramById

Zweck
Alle in Rational System Architect gespeicherten Diagramme sind intern eindeutig durch die Verwendung einer Datenverzeichnis-ID gekennzeichnet. Durch diese Methode wird ein Diagramm als Diagrammobjekt von der angegebenen ID zurückgegeben.

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`Id`

Verwendung: erforderlich

Datentyp: lang

Alle in Rational System Architect gespeicherten Diagramme sind intern eindeutig durch die Verwendung einer Datenverzeichnis-ID gekennzeichnet.

Beispiel

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

GetFilteredDefinitions

Zweck
Gibt eine gefilterte Definitionsobjektgruppe einer Enzyklopädie zurück.

Parameter
Datentyp: SAObjects

Syntax

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`WildCardName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Filterkriterien (z. B. "C*" = alle Definitionen, die mit "C" beginnen)

Anmerkung: Bei der Suche mit Platzhalterzeichen muss die Groß-/Kleinschreibung beachtet werden.

`SAType`

Verwendung: erforderlich

Datentyp: lang

Typ der Rational System Architect-Definition, die abgerufen wird (z. B. DFXPROCESS oder 3)

Anmerkung: Eine vollständige Liste aller SA-Definitionen und der zugehörigen internen Konstantennamen und -nummern finden Sie in der Datei DEFNS.BAS im Rational System Architect-Verzeichnis.

Beispiel

Durch den folgenden Befehl werden alle Prozessdefinitionen zurückgegeben, die mit "C" beginnen.

```
Dim oCollectionofDefinitions As SAObjects

Set oCollectionofDefinitions =
  oEncyclopedia.GetFilteredDefinitions("C*",
  DFXPROCESS)

Call oCollectionofDefinitions.ReadAll
```

Die SAObjects-Objektgruppe wird erst dann vollständig ausgefüllt, wenn das Flag "Complete" für die Objektgruppe "True" ist. "GetFilteredDefinitions" sollte zusammen mit der ReadAll- oder der IsMoreThan-Methode verwendet werden.

GetFilteredDiagrams

Zweck

Gibt eine gefilterte Diagrammobjektgruppe einer Enzyklopädie zurück.

Parameter

Datentyp: SAObjects

Syntax

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`WildCardName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Filterkriterien (z. B. "C*" = alle Diagramme, die mit "C" beginnen)

Anmerkung: Bei der Suche mit Platzhalterzeichen muss die Groß-/Kleinschreibung beachtet werden.

`SAType`

Verwendung: erforderlich

Datentyp: lang

Typ des Rational System Architect-Diagramms, das abgerufen wird (z. B. GTCATPROCESSFLOW oder 89)

Anmerkung: Eine vollständige Liste aller SA-Diagramme und der zugehörigen internen Konstantennamen und -nummern finden Sie in der Datei DIAGRAMS.BAS im Rational System Architect-Verzeichnis.

Beispiel

Durch den folgenden Befehl werden alle Gane & Sarson-Diagramme, die mit "Pr" beginnen, zurückgegeben.

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

Die SAObjects-Objektgruppe wird erst dann vollständig ausgefüllt, wenn das Flag "Complete" für die Objektgruppe "True" ist. "GetFilteredDiagrams" sollte zusammen mit der ReadAll- oder der IsMoreThan-Methode verwendet werden.

GetRelationMetric

Zweck

Ruft Beziehungsinformationen oder -verhalten zwischen zwei Rational System Architect-Objekten in der Enzyklopädie ab.

Syntax

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`SAObject1`

Verwendung: erforderlich

Datentyp: Objekt

Eines der beiden erforderlichen Rational System Architect-Objekte, die zum Ausführen der Beziehungsmessdaten erforderlich sind.

`SAObject2`

Verwendung: erforderlich

Datentyp: Objekt

Eines der beiden erforderlichen Rational System Architect-Objekte, die zum Ausführen der Beziehungsmessdaten erforderlich sind.

`Relation`

Verwendung: erforderlich

Datentyp: RELATETYPE

Der Beziehungstyp, der zwischen den beiden parametrisierten SA-Objekten oben vorhanden ist. In Kapitel 16 finden Sie eine vollständige Liste aller Rational System Architect-Beziehungstypen und die zugehörigen Beschreibungen.

`Depth`

Verwendung: erforderlich

Datentyp: lang

Die Anzahl der Beziehungen zwischen den beiden parametrisierten SA-Objekten oben. Wenn z. B. Objekt 1 eine Datenstruktur ist, die Objekt 2 enthält, das ein Datenelement ist, ist die Tiefe zwischen den beiden Objekten 1.

`Metric`

Verwendung: erforderlich

Datentyp: RELATIONMETRIC

Beziehungsmessdaten; eine vollständige Liste aller Beziehungsmessdaten finden Sie unten.

`FieldType`

Verwendung: erforderlich

Datentyp: FLDTYPE

Feldtyp; eine vollständige Liste der Rational System Architect-Feldtypen finden Sie in Kapitel 17.

`NbrChars`

Verwendung: optional

Datentyp: lang

Die Anzahl der von SA zurückgegebenen Zeichen, die vor dem Dezimalzeichen angezeigt werden.

`NbrDec`

Verwendung: optional

Datentyp: lang

Die Anzahl der von SA zurückgegebenen Zeichen, die nach dem Dezimalzeichen angezeigt werden.

GetXML

Zweck

Exportiert die XML-Zeichenfolge der Enzyklopädie in eine gültige XML-Datei.

Syntax

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`StrXML`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Wenn für `bToFile` "True" festgelegt ist, ist dies der Name einer gültigen XML-Datei, in die SA die XML der Enzyklopädie exportiert. Wenn für `bToFile` "False" festgelegt ist, agiert "strXML" als XML-Zeichenfolge.

`bToFile`

Verwendung: erforderlich

Datentyp: boolesch

Ist "True" festgelegt, erstellt die Methode die Datei, die im Paramater "strXML" angegeben ist. Bei "False" füllt die Methode den Parameter "strXML" mit der XML-Zeichenfolge der Enzyklopädie aus.

OpenLock…CloseUnlock-Anweisung

Zweck

Die OpenLock- und die CloseUnlock-Methode steuern den Sperrstatus der aktuellen Rational System Architect-Enzyklopädie. Dadurch wird festgelegt, ob die Enzyklopädie für den schreibgeschützten Zugriff, den Lese-Schreib-Zugriff oder den Aktualisierungszugriff gesperrt ist, während VBA-Operationen stattfinden.

Wenn eine OpenLock-Methode in einem bestimmten Modus ausgeführt wird, muss eine CloseUnlock-Methode später im Code im selben Modus ausgeführt werden.

Die OpenLock- und die CloseUnlock-Methode können mehrfach im Code ausgeführt werden, wenn in der Enzyklopädie verschiedene Ebenen von Sperrung erforderlich sind.

Wenn die OpenLock- und die CloseUnlock-Methode nicht im VBA-Code ausgeführt werden, führt Rational System Architect bei Bedarf jedes Mal, wenn eine Objektmodellmethode ausgegeben wird, eine eigene Sperrung durch. Dies kann die Leistung des Makros beeinträchtigen.

Die beiden Methoden geben einen booleschen Wert zurück, der angibt, ob der Aufruf erfolgreich war.

Syntax

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`LockMode`

Verwendung: erforderlich

Data Type: EncyLockMode

Sperrstatus der aktuellen Rational System Architect-Enzyklopädie.

EncyLockMode	Bedeutung
NETOPENREAD	Schreibgeschützt
NETOPENREADWRITE	Lese-Schreib-Zugriff
NETOPENUPDATE	Aktualisierungszugriff, während VBA-Anwendungen ausgeführt werden

Beispiel

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

SetXML

Zweck

Importiert eine XML-Datei in die Enzyklopädie.

Syntax

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`StrXML`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Wenn für "bFromFile" der Wert "True" festgelegt ist, ist dies der Name einer gültigen XML-Datei, aus der SA den XML-Code importiert. Wenn für "bFromFile" der Wert "False" festgelegt ist, ist dies die XML-Zeichenfolge der Enzyklopädie.

`bFromFile`

Verwendung: erforderlich

Datentyp: boolesch

Wenn der Wert "True" festgelegt ist, importiert die Methode XML aus der Datei, die im Parameter "strXML" angegeben ist. Wenn "False" festgelegt ist, füllt die Methode den Parameter "strXML" mit der XML-Zeichenfolge der Enzyklopädie aus.

`bValidate`

Verwendung: erforderlich

Datentyp: boolesch

Wenn "True" festgelegt ist, wird die XML-Zeichenfolge vom Parser überprüft.

SetXMLEx

Zweck

Syntax

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

`Encyclopedia Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Enzyklopädieklasse.

`StrXML`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Wenn für "bFromFile" der Wert "True" festgelegt ist, ist dies der Name einer gültigen XML-Datei, aus der SA den XML-Code importiert. Wenn für "bFromFile" der Wert "False" festgelegt ist, ist dies die XML-Zeichenfolge der Enzyklopädie.

`ICollision`

Verwendung: erforderlich

Datentyp: lang

Kollisionsoptionen	Beschreibung
0	Vorhandene Definition oder vorhandenes Diagramm nie überschreiben
1	Wenn die Definition vorhanden ist, alle Eigenschaften dafür abrufen, sie löschen, erneut erstellen und die Eigenschaften erneut ausfüllen
2	Einzelne Felder aktualisieren, wenn Daten angegeben werden
3	Einzelne Felder aktualisieren - Feld löschen, wenn keine Daten vorhanden sind
256	Vorhandenes Diagramm immer ersetzen

`bFromFile`

Verwendung: erforderlich

Datentyp: boolesch

Wenn der Wert "True" festgelegt ist, importiert die Methode XML aus der Datei, die im Parameter "strXML" angegeben ist. Wenn "False" festgelegt ist, füllt die Methode den Parameter "strXML" mit der XML-Zeichenfolge der Enzyklopädie aus.

`bValidate`

Verwendung: erforderlich

Datentyp: boolesch

Wenn "True" festgelegt ist, wird die XML-Zeichenfolge vom Parser überprüft.

Arbeitsbereichsmethoden

Das Hinzufügen von Arbeitsbereichen zu Rational System Architect V 11.3 beeinträchtigt keine vorhandenen Makros, da kein Arbeitsbereichsobjekt in das Objektmodell eingeführt wurde. Rational System Architect ermöglicht nur den Zugriff auf jeweils einen Arbeitsbereich, weshalb das Arbeitsbereichsobjekt fast vollständig dem Enzyklopädieobjekt zugeordnet ist. Die folgenden Erweiterungen wurden zum Manipulieren von Arbeitsbereichen durch den Benutzer am Objektmodell vorgenommen.

IsEncyWorkspaceEnabled

Zweck
Gibt "True" zurück, wenn die aktuelle Enzyklopädie Arbeitsbereiche unterstützt.

GetWorkspaceID

Zweck
Gibt die ID des aktuellen Arbeitsbereichs zurück.

SetWorkspaceID

Zweck
Ändert den aktuellen Arbeitsbereich.

GetWorkspaceTree

Zweck
Gibt eine XML-Verzeichnisstruktur mit Arbeitsbereichsnamen, -IDs und Ausgangsdatenstatus zurück.

GetWorkspaceName

Zweck
Gibt den Namen des aktuellen Arbeitsbereichs zurück.

IsWorkspaceReadOnly

Zweck
Gibt "True" zurück, wenn der aktuelle Arbeitsbereich schreibgeschützt ist.

Aufrufe für Arbeitsbereichsanwendungsklassen

Anwendungsklassenaufrufe für Arbeitsbereiche sind unten aufgeführt.

OpenEncyclopediaUsingConnectionStringAndWorkspace

Zweck
Version von OpenEncyclopediaUsingConnectionString mit Arbeitsbereichsunterstützung.

OpenEncyclopediaUsingDisplayNameAndWorkspace

Zweck
Version von OpenEncyclopediaUsingDisplayName mit Arbeitsbereichsunterstützung.

Ereignisse für Arbeitsbereichsanwendungs-klassen

Anwendungsklassenereignisse für Arbeitsbereiche sind unten aufgeführt.

WorkspaceOpen

Zweck
Ereignis, das beim Ändern des Arbeitsbereichs auftritt.

WorkspaceBeforeOpen

Zweck
Ereignis, das vor dem Ändern des Arbeitsbereichs auftritt und ein Abbrechen ermöglicht. Beachten Sie, dass alle Objektmodellreferenzen auf Definitionen und Diagramme wie beim Ändern der Enzyklopädie ungültig werden.

Anmerkung: Dies gilt nur bei Rational System Architect ab Version 11.3.0.2.

Beziehungsmessdaten

Beziehungsmessdaten weichen von Diagramm-, Symbol- und Definitionsmessdaten darin ab, dass es sich dabei um Elemente interner Funktionalität handelt, die Informationen zur Beziehung zwischen zwei Rational System Architect-Objekten in der Enzyklopädie abfragen. Für alle Beziehungsmessdaten muss deklariert werden, welche beiden Objekte untersucht werden sollen und welche Beziehung zwischen ihnen besteht. Je nachdem, welche Beziehungsmessdaten verwendet werden, sind nur bestimmte Rational System Architect-Objekte mit bestimmten Beziehungen gültig.

Um auf diese Beziehungsmessdaten zuzugreifen, muss der Benutzer die GetRelationMetric-Methode in der Enzyklopädieklasse aufrufen. Ein Verzeichnis aller Beziehungsmessdaten finden Sie in der RELATIONMETRIC-Aufzählungsliste im SA-Objektbrowser. Im Folgenden finden Sie eine Tabelle aller verfügbaren Beziehungsmessdaten mit den zugehörigen Beschreibungen und Parametern.

Beziehungsmessdaten	Beschreibung	Parameter
RELMETCRUD	Gibt die Kombination der Buchstaben CRUD (Create, Read, Update, Delete - Erstellen, Lesen, Aktualisieren, Löschen) neben dem Prozessnamen im Datenspeicher als Zeichenfolge zurück.	SAObjects: Data Store, Process
RELMETDEPTH	Gibt die Anzahl der "Uses"-Beziehungen zwischen zwei Objekten als Zahl zurück.	SAObjects: Muss mindestens eine "Uses"- oder "Used by"-Beziehung aufweisen.
RELMETICOMROLE	Gibt die Beziehungsrolle [input (Eingabe), control (Steuerung), mechanism (Mechanismus) oder boundary (Grenze)] zwischen einem ICOM-Pfeil und dem damit verbundenen Funktions-/Aktivitätssymbol als Zeichenfolge zurück.	SAObjects: ICOM Arrow, Function/Activity RelTypes: RELCONNSTART, RELSTARTAT, RELCONNEND, RELENDAT

Beziehungsmessdaten	Beschreibung	Parameter
RELMETINPUT	Überprüft, ob ein Datenflussymbol in andere Symbole oder Diagramme "fließt". Gibt ein boolesches Feld zurück.	SAObjects: Datenfluss-, Diagramm- oder Knotensymbol RelType: RELCONNSTART, RELSTARTAT, RELCONNEND, RELENDAT, RELDIAGRAMCON, RELCONDIAGRAM
RELMETOUTPUT	Überprüft, ob ein Datenflussymbol aus einem anderen Symbol oder Diagramm "fließt". Gibt ein boolesches Feld zurück.	SAObjects: Datenfluss-, Diagramm- oder Knotensymbol RelType: RELCONNSTART, RELSTARTAT, RELCONNEND, RELENDAT, RELDIAGRAMCON, RELCONDIAGRAM
RELMETSTATETABLE	Überprüft, ob der Name des Ereignisses, das an eine Ausgabe-Übergangslinie angehängt ist, die mit einem Status verbunden ist, im Statusdefinitionsdialogfenster genannt wird. Bei "True" wird der Statusname zurückgegeben.	SAObjects: Shlaer State, Shlaer Transition line RelType: RELCONNEND, RELCONNSTART

6

Die Diagrammklasse

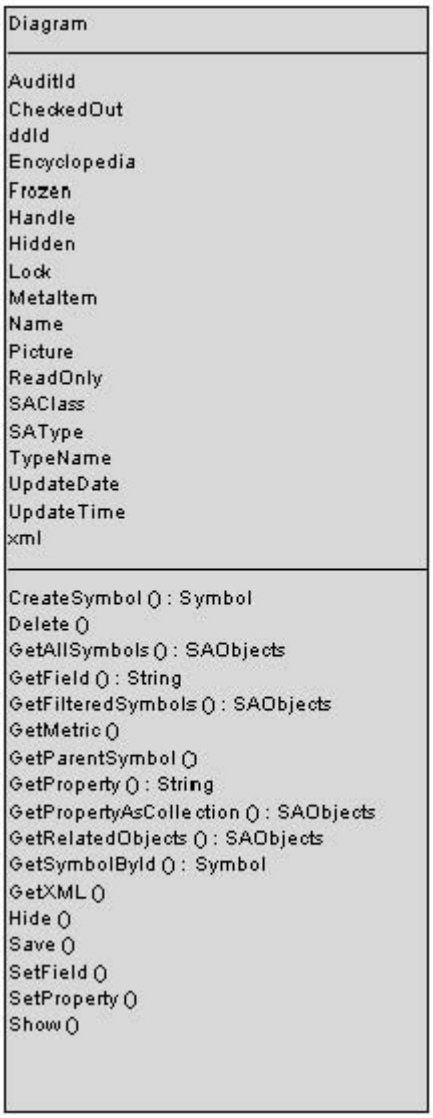
In diesem Kapitel enthaltene Themen	Seite
Attribute	6-3
Methoden	6-9
Felder	6-21
Messdaten	6-28

Einführung

Dies ist eine Instanz eines Diagramms, das in der Enzyklopädie enthalten ist.

Gehen Sie wie folgt vor, um das derzeit aktive Diagramm als Objektverwendung zurückzugeben:

```
Dim oApplication As
  SA2001.Application

Dim oDiagram As Diagram

Set oApplication = New
  Application

Set oDiagram =
  oApplication.oEncyclopedia.
  GetCurrentDiagram
```



Attribute

AuditID

Zweck
Alle in Rational System Architect gespeicherten Elementdiagramme sind mit der ID der Person versehen, die das Diagramm erstellt oder zuletzt geändert hat. Die ID ist im Diagramm als AuditId codiert.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

CheckedOut

Zweck
Wenn "True" festgelegt ist, wird das Diagramm für alle Benutzer, bis auf den mit der AuditID, die es ausgecheckt hat, schreibgeschützt.

Parameter
Datentyp: boolesch

ddID

Zweck
Alle in Rational System Architect gespeicherten Diagramme sind intern eindeutig durch die Verwendung einer Datenverzeichnis-ID gekennzeichnet. Durch diese Methode wird die ID eines Diagramms zurückgegeben.

Parameter
Datentyp: lang

Encyclopedia

Zweck
Ermöglicht den Zugriff mit den übergeordneten Attributen und Methoden der Enzyklopädie.

Parameter
Schreibgeschützt

Frozen

Zweck
Der Benutzer muss über die Berechtigung zum Blockieren verfügen, um dieses Attribut festzulegen. Wenn "True" festgelegt ist, ist das Diagramm für alle Benutzer schreibgeschützt, auch für den mit der AuditID, der das Diagramm blockiert hat.

Parameter
Datentyp: boolesch

Handle

Zweck
Hierbei handelt es sich um die Speicherkennung des Diagramms, die nur zur Laufzeit verfügbar ist. Diese Kennung ist nicht eindeutig und kann bei jedem Zugriff darauf unterschiedlich sein.

Parameter
Datentyp: lang

Schreibgeschützt

Beispiel

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

Hidden

Zweck
Gibt den Wert "True" oder "False" zurück, um anzugeben, ob ein Diagramm geschlossen oder geöffnet ist.

Parameter
Datentyp: boolesch

Schreibgeschützt

Locked

Zweck
Gibt den Wert "True" oder "False" zurück, um anzugeben, ob ein Diagramm gesperrt oder nicht gesperrt ist, das heißt, ob es gerade von einem Benutzer verwendet wird oder nicht.

Parameter
Datentyp: boolesch

Schreibgeschützt

MetaItem

Zweck
Ermöglicht den Zugriff mit der MetaItem-Klasse und den zugehörigen Attributen.

Parameter
Schreibgeschützt

Name

Zweck
Der Name des angegebenen Diagrammobjekts.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

Picture

Zweck
Nach dem Speichern eines Diagramms erstellt Rational System Architect eine Windows-Metadatei (.wmf) des Diagramms, die im Enzyklopädieverzeichnis gespeichert wird. Dieses Attribut ermöglicht es dem Benutzer, auf die stdPicture-Attribute und -Methoden zuzugreifen; es handelt sich um ein OLE-Automatisierungsobjekt, das Daten zum Inhalt der Abbildung enthält.

Parameter
Datentyp: stdPicture

Schreibgeschützt

ReadOnly

Zweck
Gibt "True" zurück, wenn das Diagramm schreibgeschützt geöffnet wurde.

Parameter
Datentyp: boolesch

Schreibgeschützt

SAClass

Zweck
Der Klassentyp des Diagramms. Wird auch als "Major Type Number" (Nummer des übergeordneten Typs) bezeichnet.

Parameter
Datentyp: lang

Schreibgeschützt

Anmerkung: Dadurch wird für ein Diagramm immer "1" zurückgegeben.

SAType

Zweck
Die konstante Ganzzahl des Diagramms. Alle Diagramme in Rational System Architect weisen eine eindeutige numerische Konstanten-ID auf.

Parameter
Datentyp: lang

Schreibgeschützt

TypeName

Zweck
Der Typ des Diagramms als Zeichenfolge, z. B. "Entity Relation" (Entitätenbeziehung).

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

UpdateDate

Zweck
Das Datum, an dem das Diagramm zuletzt geändert wurde.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

UpdateTime

Zweck
Die Uhrzeit, zu der das Diagramm zuletzt geändert wurde.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

xml

Zweck
Die XML-Zeichenfolge des Diagramms. Bearbeitet durch die GetXML-Methode.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

Methoden

CreateSymbol

Zweck
Erstellt eine Instanz der Symbolklasse mit einem bestimmten Namen und Typ.

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse, zu der das Symbol hinzugefügt wird.

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Name des neuen Symbols

`SAType`

Verwendung: erforderlich

Datentyp: lang

Typ des Rational System Architect-Symbols, das erstellt wird
(z. B. ETCATELEMBUSPROC oder 445)

Anmerkung: Eine vollständige Liste aller SA-Symbole und der zugehörigen internen Konstantennamen und -nummern finden Sie in der Datei SYMBOLS.BAS im Rational System Architect-Verzeichnis.

Anmerkung: Um ein SA-Symbol in einem Diagramm erfolgreich zu erstellen, müssen Sie die Save-Methode der Diagrammklasse aufrufen. Andernfalls wird das neue Symbol aus dem Diagramm gelöscht, wenn die Enzyklopädie geschlossen wird.

Delete

Zweck
Löscht ein durch ein Diagrammobjekt angegebenes Diagramm.

GetAllSymbols

Zweck
Durch diese Methode werden alle Symbole, die im angegebenen Diagrammobjekt enthalten sind, als SAObjects-Objektgruppe zurückgegeben. Die SAObjects-Objektgruppe wird erst dann vollständig ausgefüllt, wenn das Flag "Complete" für die Objektgruppe "True" ist. "GetAllSymbols" sollte zusammen mit der ReadAll- oder der IsMoreThan-Methode verwendet werden.

Regeln
Eine SAObjects-Variable muss als Objektsammlung von Symbolen dimensioniert und festgelegt sein. Siehe folgendes Beispiel.

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

GetField

Zweck
Hierbei handelt es sich um Merkmale des Diagramms, wie z. B. "Symbol Grid Size" (Größe des Symbolrasters), "Line Grid Size" (Größe der Rasterlinie), "Level Number" (Ebenennummer), von denen einige festgelegt werden können, wie z. B. "Diagram Name" (Diagrammname) und andere nicht, wie z. B. "Diagram Type" (Diagrammtyp).

Syntax
```
Diagram Object. GetField FieldID
```

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`FieldID`

Verwendung: erforderlich

Datentyp: DGMFLD

Diagrammfeld; eine vollständige Liste aller Diagrammfelder finden Sie unten.

GetFilteredSymbols

Zweck

Um die in einem Diagramm enthaltenen Symbole zu filtern, geben Sie die Filterkriterien als Platzhalterzeichen für das erste Argument an und geben Sie den Typ des betreffenden Symbols als zweites Argument an; ein Teil dieses Arguments kann auch “” sein. Dadurch wird eine SAObjects-Objektgruppe zurückgegeben. Die SAObjects-Objektgruppe wird erst dann vollständig ausgefüllt, wenn das Flag "Complete" für die Objektgruppe "True" ist. "GetFilteredSymbols" sollte zusammen mit der ReadAll- oder der IsMoreThan-Methode verwendet werden.

Parameter

Datentyp: SAObjects

Syntax

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`WildCardName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Filterkritereien (z. B. "C" = Alle Symbole, die mit "C" beginnen)

Anmerkung: Bei der Suche mit Platzhalterzeichen muss die Groß-/Kleinschreibung beachtet werden.

`SAType`

Verwendung: erforderlich

Datentyp: lang

Typ des Rational System Architect-Symbols, das erstellt wird (z. B. ETCATELEMBUSPROC oder 445)

Anmerkung: Eine vollständige Liste aller SA-Symbole und der zugehörigen internen Konstantennamen und -nummern finden Sie in der Datei SYMBOLS.BAS im Rational System Architect-Verzeichnis.

Beispiel

Bei diesem Beispiel werden alle Entitätensymbole im Diagramm zurückgegegeben, die mit dem Buchstaben "P" beginnen.

```
Dim oDiagram as Diagram, oCollectionofSymbols As
  SAObjects

Set oCollectionofSymbols =
  oDiagram.GetFilteredSymbols("P", ETECACTIVITY)

Call oCollectionofSymbols.ReadAll
```

Get Metric

Zweck

Ruft bestimmte Listen, Berechnungen und Teile von interner Funktionalität, die Diagramme betreffen, ab.

Syntax

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`Metric`

Verwendung: erforderlich

Datentyp: DIAGRAMMETRIC

Diagrammmessdaten; eine vollständige Liste aller Diagrammmessdaten finden Sie unten.

`FieldType`

Verwendung: optional

Datentyp: FLDTYPE

Feldtyp; eine vollständige Liste der Rational System Architect-Feldtypen finden Sie in Kapitel 17.

`NbrChars`

Verwendung: optional

Datentyp: lang

Wenn der Feldtyp eingegeben wurde, teilt dieser Parameter SA mit, wie viele Zeichen vor dem Dezimalzeichen zurückzugeben sind.

`NbrDec`

Verwendung: optional

Datentyp: lang

Wenn der Feldtyp eingegeben wurde, teilt dieser Parameter SA mit, wie viele Stellen nach dem Dezimalzeichen zurückzugeben sind.

GetParentSymbol

Zweck

Ein Diagramm kann das untergeordnete Element eines übergeordneten Symbols sein, wie z. B. in einem Datenflussdiagramm. Durch diese Methode wird das übergeordnete Symbolobjekt für das angegebene Diagrammobjekt zurückgegeben.

GetProperty

Zweck

Dadurch wird der Eigenschaftsinhalt für eine beliebige angegebene Diagrammeigenschaft zurückgegeben.

Parameter

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einem elementaren Geschäftsprozess eines Prozessgrafikdiagramms ist z. B. die Eigenschaft "Locations" (Positionen) eine Umbenennung - die Eigenschaft heißt eigentlich "Location Types" (Standorttypen). Dies können Sie nur wissen, wenn Sie die Definition eines elementaren Geschäftsprozesses in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Location Types" heißt, aber als "Locations" bezeichnet wurde.

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

Syntax

```
Diagram Object.GetProperty Name
```

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

GetPropertyAsCollection

Zweck

Manche Eigenschaften definieren Beziehungen zu anderen Eigenschaften. Eine Prozessgrafik verweist z. B. auf einen Prozessthread über die zugehörige Eigenschaft "Process Thread". Durch diese Methode wird eine Objektgruppe von OneOf- und ListOf-Diagrammen oder -Definitionen zurückgegeben. Weitere Informationen zu OneOf- und ListOf-Eigenschaftstypen finden Sie in Kapitel 14.

Parameter

Data Type: OfCollection

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einem elementaren Geschäftsprozess eines Prozessgrafikdiagramms ist z. B. die Eigenschaft "Locations" (Positionen) eine Umbenennung - die Eigenschaft heißt eigentlich "Location Types" (Standorttypen). Dies können Sie nur wissen, wenn Sie die Definition eines elementaren Geschäftsprozesses in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Location Types" heißt, aber als "Locations" bezeichnet wurde.

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

Syntax

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`PropName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

Beispiel

```
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

Zweck

Durch diese Methode wird eine SAObjects-Objektgruppe zugehöriger Objekte für das aktuelle Diagrammobjekt zurückgegeben.

Syntax

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`RelType`

Verwendung: erforderlich

Datentyp: RELATETYPE

SA-Beziehung; eine vollständige Liste aller Beziehungen finden Sie in Kapitel 16.

GetSymbolById

Zweck
Alle in Rational System Architect gespeicherten Diagramme sind intern eindeutig durch die Verwendung einer Datenverzeichnis-ID gekennzeichnet. Durch dieses Methode wird ein Symbol als ein Objekt von der zugehörigen ID zurückgegeben.

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`Id`

Verwendung: erforderlich

Datentyp: lang

Alle in Rational System Architect gespeicherten Symbole sind intern eindeutig durch die Verwendung einer Datenverzeichnis-ID gekennzeichnet.

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

GetXML

Zweck
Exportiert die XML-Zeichenfolge des Diagramms in eine gültige XML-Datei.

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`StrXMLTextOut`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Eine gültige XML-Datei, in die SA die XML-Zeichenfolge des Diagramms exportiert.

Hide

Zweck
Wird verwendet, um eine Instanz eines Diagramms, die derzeit geöffnet ist, zu schließen.

Syntax

```
Call oDiagram.Hide
```

Save

Zweck
Wird verwendet, um eine Instanz eines Diagramms zu speichern.

Syntax

```
Call oDiagram.Save
```

SetField

Zweck
"SetField" ermöglicht das Festlegen eines angegebenen Werts für ein Diagrammfeld.

Syntax

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`FieldID`

Verwendung: erforderlich

Datentyp: DGMFLD

Diagrammfeld; eine vollständige Liste aller Diagrammfelder finden Sie unten.

`Value`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Wert des Diagrammfelds

SetProperty

Zweck

Damit ein Wert für eine Diagrammeigenschaft festgelegt werden kann, muss der Name der Eigenschaft als erstes Argument und der Wert als zweites Argument festgelegt werden. Die Eigenschaftsnamen finden Sie in den Dateien SAPROPS.CFG und USRPROPS.TXT.

Parameter

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einem elementaren Geschäftsprozess eines Prozessgrafikdiagramms ist z. B. die Eigenschaft "Locations" (Positionen) eine Umbenennung - die Eigenschaft heißt eigentlich "Location Types" (Standort-typen). Dies können Sie nur wissen, wenn Sie die Definition eines elementaren Geschäftsprozesses in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Location Types" heißt, aber als "Locations" bezeichnet wurde.

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

Syntax

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

`Diagram Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Diagrammklasse

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

`Value`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Wert der Diagrammeigenschaft

Show

Zweck

Durch diese Methode wird das Diagramm in einer Rational System Architect-Anzeige geöffnet.

Syntax

```
Call oDiagram.Show
```

Diagrammfelder

Die Diagrammfeldeigenschaft kann eine Anzahl von Eigenschaften zum Diagramm enthalten. Sie enthält in der Regel Informationen, die ein Benutzer nicht direkt eingeben kann, sondern die sich aus der normalen Verwendung ableiten. Im Objektmodell gibt es einen Aufzählungstyp namens *DGMFLD*. Dieser wird als Parameter an die Operationen **GetField(FieldID as *DGMFLD*)** und **SetField(FieldID as *DGMFLD*, Value as String)** übergeben. So kann der VBA-Programmierer Diagrammfelder auf unterer Ebene sowohl lesen als auch aktualisieren.

DGMFLD constant	Beschreibung	Datentyp
DIAGFLD_BBORDER	Aktiviert das Kontrollkästchen "Berichtsrahmen" auf der Seite "Seite einrichten". Dadurch kann der Benutzer einen Rahmen um den Bericht platzieren.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_BDGMBORDER	Wählt einen Rahmen aus der Dropdown-Liste für Diagrammrahmen im Fenster "Seite einrichten" aus.	“0” = Keine Form und kein Rahmen “1” = Einfacher Rahmen
DIAGFLD_BDGMPDEFAULT	Legt die Einstellungen im Fenster "Seite einrichten" als Standardeinstellungen fest.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_BORDEROFFSET	Legt den Wert "Abstand" im Fenster "Seite einrichten" fest. Durch dieses Feld wird der Rahmen um den angegebenen Wert vom Berichtstext abgerückt.	Zeichenfolge Numerischer Wert in hunderstel Zoll
DIAGFLD_BPRESENTATIONMENU	Aktiviert die automatische Aufnahme des Menüs "Darstellung" in die Zeichentoolbox. Die im Menü "Darstellung" enthaltenen zusätzlichen Symbole	“0” = Inaktiviert “1” = Aktiviert

DGMFLD constant	Beschreibung	Datentyp
	enthalten grundsätzliche Angaben zu Computer, Telefon, Person, Festplatte und Drucker. Die Symbole werden im Diagramm durch dieselbe Methode wie jedes andere Blocksymbol gezeichnet und können nach Wunsch benannt werden.	
DIAGFLD_BREADONLY	Legt Schreibschutz für das Diagramm fest.	“0” = False “1” = True
DIAGFLD_BSHOWGRID	Aktiviert die automatische Anzeige des zugrunde liegenden Rasters.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_BSHOWLINESHADOW	Aktiviert die automatische Anzeige eines Schattens um alle Liniensymbole.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_BSHOWNODESHADOW	Aktiviert die automatische Anzeige eines Schattens um alle Knotensymbole.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_BSHOWPAGES	Aktiviert das Kontrollkästchen "Seiten" im Fenster mit den Diagrammanzeigeoptionen. Wenn es aktiviert ist, werden die Seitendruckbereiche als gepunktete Linien angezeigt, sodass, wenn das Diagramm im Modus "Originalgröße" gedruckt wird, die Seiten-ränder vorangezeigt werden können.	“0” = Inaktiviert “1” = Aktiviert

DGMFLD constant	Beschreibung	Datentyp
DIAGFLD_BSHOWRULER	Aktiviert die Anzeige der Lineale an der X- und Y-Achse in cm oder Zoll, je nach den lokalen Einstellungen auf dem PC. Die X- und Y-Lineale werden mit der Zeichnung gespeichert, wenn diese Option aktiviert ist.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_BSHOWSCROLL	Aktiviert die automatische Anzeige der Schiebeleisten, die es dem Benutzer ermöglichen, in einem Diagramm zu navigieren, wenn es im aktuellen Anzeigemodus größer als eine Anzeige ist. Diese Option ist standardmäßig aktiviert.	“0” = Nicht aktiviert “1” = Aktiviert
DIAGFLD_BSHOWTEXTSHADOW	Aktiviert die automatische Anzeige eines Schattens um alle Textsymbole.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_BSNAPGRIDENT	Richtet alle Symbole an der nächsten Rasterlinie (normalerweise nicht sichtbar) im Diagramm aus, nachdem Sie die Rastereinstellung in eine gröbere oder feinere Rastereinstellung geändert haben.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_BSNAPGRIDLIN	Richtet alle Symbole an der nächsten Rasterlinie (normalerweise nicht sichtbar) im Diagramm aus, nachdem Sie die Rastereinstellung in eine gröbere oder feinere Rastereinstellung geändert haben.	“0” = Inaktiviert “1” = Aktiviert
DIAGFLD_CGRAPHNAME	Name des Diagramms	Zeichenfolge

DGMFLD constant	Beschreibung	Datentyp
DIAGFLD_CLEVELNUMBER	Ebenennummer des Diagramms	Schreibgeschützt Zeichenfolge
DIAGFLD_DDDIAGRAM_DDIDENTITY	Datenverzeichnis-ID für das Diagramm	Schreibgeschützt Numerisch
DIAGFLD_IDGMFORM	Wählt die Rahmenform aus der Dropdown-Liste mit Diagrammrahmen im Fenster "Seite einrichten" aus.	“0” = Keine Form und kein Rahmen “1” = IDEF0 betriebsfähig “2” = IDEF0-Veröffentlichung “3” = IDEF3 “4” = IDEF3 freigegeben “5” = SSADM Form
DIAGFLD_IGRAPHTYPE	SA-Typ des Diagramms	Zeichenfolge Interne Konstantennummer des Diagramms. Eine vollständige Liste finden Sie in der Datei DIAGRM.BAS im SA-Verzeichnis.
DIAGFLD_PGRIDNUMENT	Knotensymbol in Raster-punkten pro Zoll	“[Vertikal] [Horizontal]” **Anmerkung:** Muss zusammen mit DIAGFLD_PGRIDSIZEENT verwendet werden.
DIAGFLD_PGRIDNUMLIN	Liniensymbol in Raster-punkten pro Zoll	“[Vertikal] [Horizontal]” **Anmerkung:** Muss zusammen mit DIAGFLD_PGRIDSIZELIN verwendet werden.

DGMFLD constant	Beschreibung	Datentyp
DIAGFLD_PGRIDSIZEENT	Knotensymbol in Zoll pro Rasterpunkt	“[Vertikal] [Horizontal]” in hunderstel Zoll **Anmerkung:** Muss zusammen mit DIAGFLD_PGRIDNUMENT verwendet werden.
DIAGFLD_PGRIDSIZELIN	Liniensymbol in Zoll pro Rasterpunkt	“[Vertikal] [Horizontal]” in hunderstel Zoll **Anmerkung:** Muss zusammen mit DIAGFLD_PGRIDNUMLIN verwendet werden.
DIAGFLD_PGRIDUNIT100	Legt den Abstand fest, um den ein Objekt innerhalb des Rasters gezogen werden kann. Standardwert =“100 100”	“[Vertikal] [Horizontal]” in hunderstel Zoll
DIAGFLD_PSHADOWDELTA	Legt den Abstand fest, um den ein Schatten vom Symbol entfernt angezeigt wird. Standardwert = “20 10”	“[Vertikal] [Horizontal]” in hunderstel Zoll
DIAGFLD_RGBSHADOWCOLOR	Legt die Schattenfarbe fest.	“[RGB-Farbe]”
DIAGFLD_RMARGIN	Legt die Ränder im Fenster "Seite einrichten" fest.	“[Links] [Oben] [Rechts] [Unten]” in hunderstel Zoll.
DIAGFLD_SAAUDITID	Prüf-ID des Diagramms	Schreibgeschützt Zeichenfolge
DIAGFLD_SAIDENTITY	Datenverzeichnis-ID für das Diagramm	Schreibgeschützt Numerisch

DGMFLD constant	Beschreibung	Datentyp
DIAGFLD_SALOCK	Sperrt das Diagramm.	“0” = Nicht gesperrt “1” = Gesperrt
DIAGFLD_SAMAJORTYPE	Übergeordneter Typ (z. B. Diagramm)	Schreibgeschützt Zeichenfolge
DIAGFLD_SAMAJORTYPENUMBER	Übergeordnete Typennummer (z. B. 1)	Schreibgeschützt Numerisch
DIAGFLD_SANAME	Name des Diagramms	Schreibgeschützt Zeichenfolge
DIAGFLD_SANUMBER	Ebenennummer des Diagramms (nur IDEF0)	Schreibgeschützt Numerisch
DIAGFLD_SATYPE	Typ des Diagramms (z. B. "Prozessgrafik, "Entitätenbeziehung" usw.).	Schreibgeschützt Zeichenfolge
DIAGFLD_SATYPENUMBER	Interne Konstantennummer des Diagramms	Schreibgeschützt Numerisch
DIAGFLD_SAUPDATEDATE	Datum der letzten Aktualisierung	Schreibgeschützt Datumsfeld
DIAGFLD_SAUPDATETIME	Uhrzeit der letzten Aktualisierung	Schreibgeschützt Zeitfeld
DIAGFLD_USEDENTCOUNT	Anzahl der Symbole im Diagramm	Lang (hexadezimal) Schreibgeschützt

DGMFLD constant	Beschreibung	Datentyp
DIAGFLD_WBORDERPENSTYLE	Umrissstil des Rands im Fenster "Seite einrichten"	“[Nummer für SymPenStyle]” Eine vollständige Liste aller SA-Stifstile finden Sie in Kapitel 7.
DIAGFLD_WORIENTATION	Die Ausrichtung für den Diagrammdruck im Fenster "Seite einrichten".	“0” = Druckerstandard “1” = Hochformat “2” = Querformat “3” = Beste Anpassung

Diagrammessdaten

In der Vergangenheit wurden Messwerte verwendet, um Listen zu erstellen, Regelprüfungen auszuführen und Berechnungen für verschiedene Rational System Architect-Berichte bereitzustellen. Nun kann der Benutzer individuelle Messdaten ausführen, indem er die GetMetric-Methode in der Diagrammklasse ausführt. Ein Verzeichnis aller Diagramm-messdaten finden Sie in der DIAGRAMMETRIC-Aufzählungsliste im SA-Objektbrowser. Im Folgenden sehen Sie eine Tabelle aller verfügbaren Diagrammmessdaten und der zugehörigen Beschreibungen.

Diagrammmessdaten	Beschreibung
DIAGMETBALANCE	Vergleicht die Eingabe- und Ausgabelinien des Diagramms mit den Eingabe- und Ausgabelinien des zugehörigen übergeordneten Prozesses. Erstellt eine Liste nicht übereinstimmender Eingabe- und Ausgabelinien mit dem Namen und dem Symboltyp. (Siehe die Hilfedatei zur Neuausrichtung von übergeordneten Elementen.)
DIAGMETCHARCOUNT	Gibt die Anzahl der Zeichen in der Eigenschaft "Beschreibung" des Diagramms zurück.
DIAGMETCURRENT	Boolesches Feld (T/F). Gibt "True" zurück, wenn das Diagramm derzeit angezeigt wird.
DIAGMETELELMENTLIST	Erstellt eine Liste der Elemente der unteren Elemente für alle Symboldefinitionen im Diagramm. Ein Element ist ein Element der unteren Ebene, wenn es keine erweiternden Beziehungen aufweist.
DIAGMETINPUTLIST	Erstellt eine Liste von Elementen der unteren Ebene, die als Eingabe für alle Symboldefinitionen im Diagramm verwendet wird. Ein Element ist ein Element der unteren Ebene, wenn es keine erweiternden Beziehungen aufweist.
DIAGMETLEVELNUMBER	Gibt die Nummer, die die hierarchische Position des Diagramms beschreibt (z. B. 5.3, 5.3.1, uws.) als Zeichenfolge zurück.

Diagrammmessdaten	Beschreibung
DIAGMETLEVELNUMBERSORT	Gibt die Nummer, die die hierarchische Position des Diagramms beschreibt, als Zeichenfolge zurück. Jede Nummer enthält 3 Stellen (z. B. 003.005.002). So kann der Benutzer die Ergebnisse besser sortieren.
DIAGMETLINECOUNT	Gibt die Anzahl der Linien in der Eigenschaft "Beschreibung" eines Diagramms zurück.
DIAGMETOUTPUTLIST	Erstellt eine Liste der Elemente der unteren Ebene, die als Ausgabe für alle Symboldefinitionen im Diagramm verwendet wird. Ein Element ist ein Element der unteren Ebene, wenn es keine erweiternden Beziehungen aufweist.
DIAGMETREFERENCE	Gibt "True" zurück, wenn auf das Diagramm von einem anderen Objekt verwiesen wird.
DIAGMETRULES	Führt eine Regelprüfung auf Nichteinhaltung von Standardmethodikregeln für das Diagramm durch.
DIAGMETSELECTED	Gibt "True" zurück, wenn das Diagramm derzeit geöffnet ist und wenn darin Symbole ausgewählt sind.
DIAGMETTOP	Gibt "False" zurück, wenn das Diagramm das untergeordnete Element von Symbolen ist. Gibt "True" zurück, wenn das Diagramm keine Erweiterung eines Symbols ist.
DIAGMETUNMARKEDLIST	Erstellt eine Liste von Elementen der unteren Ebene, die von der Definition aller Liniensymbole im Diagramm verwendet wird, die weder als Eingabe noch als Ausgabe gekennzeichnet sind (keine Pfeilspitzen). Ein Element ist ein Element der unteren Ebene, wenn es keine erweiternden Beziehungen aufweist.
DIAGMETWORDCOUNT	Gibt die Anzahl der Wörter in der Beschreibungseigenschaft des Diagramms zurück.

7

Die Symbolklasse

In diesem Kapitel enthaltene Themen	Seite
Attribute	7-3
Methoden	7-14
Felder	7-23
Messdaten	7-32

Einführung

Hierbei handelt es sich um die Symbolklasse mit den auf der rechten Seite dargestellten Attributen und Methoden.

Attribute

ArrowAtEnd

Zweck
Dadurch werden die assoziativen Eigenschaften für ein Liniensymbol festgelegt. Am Linienende wird eine Pfeilspitze erstellt.

Parameter
Datentyp: boolesch

Beispiel

```
oSymbol.ArrowAtEnd = True
```

ArrowAtStart

Zweck
Dadurch werden die assoziativen Eigenschaften für ein Liniensymbol festgelegt. Am Linieanfang wird eine Pfeilspitze erstellt.

Parameter
Datentyp: boolesch

Beispiel

```
oSymbol.ArrowAtStart = True
```

AuditId

Zweck
Alle in Rational System Architect gespeicherten Elementsymbole sind mit der ID der Person, die das Symbol erstellt oder zuletzt geändert hat, gekennzeichnet. Die ID ist im Symbol als AuditId codiert.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

ddId

Zweck
Alle in Rational System Architect gespeicherten Symbole sind intern durch die Verwendung einer Datenverzeichnis-ID eindeutig gekennzeichnet. Durch diese Methode wird die ID eines Symbols zurückgegeben.

Parameter
Datentyp: lang

Schreibgeschützt

Definition

Zweck
Ermöglicht den Zugriff auf die Definitionsklasse des Symbols.

Parameter
Schreibgeschützt

Diagram

Zweck
Ermöglicht den Zugriff auf die Diagrammklasse, in der das Symbol erstellt wurde.

Parameter
Schreibgeschützt

Encyclopedia

Zweck
Ermöglicht den Zugriff auf die Enzyklopädieklasse, zu der das Symbol gehört.

Parameter
Schreibgeschützt

FillColor

Zweck
Mit dieser Eigenschaft kann die Füllfarbe für ein Symbol zurückgegeben werden. Der Wert für die Farbe ist ein OLE_COLOR-Wert.

Parameter
Ein OLE_COLOR-Wert ist ein Wert vom Typ "BGR" (Blue, Green, Red - Blau, Grün, Rot). Um einen BGR-Wert anzugeben, geben Sie für "blue", "green" und "red" in der folgenden Formel einen Wert an (jeweils im Bereich von 0 bis 255).

BGR value = (blue * 65536) + (green * 256) + red

FontColor

Zweck
Mit dieser Eigenschaft kann die Füllfarbe für eine Symbolschriftart zurückgegeben werden. Der Wert für die Farbe ist ein OLE_COLOR-Wert.

Parameter
Ein OLE_COLOR-Wert ist ein Wert vom Typ "BGR" (Blue, Green, Red - Blau, Grün, Rot). Um einen BGR-Wert anzugeben, geben Sie für "blue", "green" und "red" in der folgenden Formel einen Wert an (jeweils im Bereich von 0 bis 255).

BGR value = (blue * 65536) + (green * 256) + red

FromCardinality

Zweck
Für Beziehungslinien in einem Entitätenbeziehungsdiagramm oder Bedingungen in einem physischen Datenmodell kann die Kardinalität am Linienanfang über diese Eigenschaft bestimmt und festgelegt werden.

Die Konstanten werden wie folgt festgelegt oder zurückgegeben:

Konstante	Zahl	Bedeutung
CARDINALITYZERO	0	Null
CARDINALITYONLYONE	1	Nur ein
CARDINALITYZEROONE	2	Null oder ein
CARDINALITYONEMULT	3	Ein oder mehrere
CARDINALITYZEROONEMULT	4	Ein, null oder mehrere
CARDINALITYMULT	5	Viele
CARDINALITYUNKNOWN	6	Nicht festgelegt
CARDINALITYNOTUSED	7	Keine Kardinalität

Handle

Zweck
Hierbei handelt es sich um die Speicherkennung des Symbols, die nur zur Laufzeit verfügbar ist. Diese Kennung ist nicht eindeutig und kann bei jedem Zugriff darauf unterschiedlich sein.

Parameter
Datentyp: lang

Schreibgeschützt

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

LineStyle

Zweck
In Diagrammen gezeichnete Linien können einen von vielen Stilen, die bestimmt und festgelegt werden können, aufweisen.

Einige der allgemeinen Konstanten werden wie folgt festgelegt oder zurückgegeben:

Konstante	Zahl	Bedeutung
LSARC	4	Elliptische Bögen
LSAUTOSTROR	19	Automatisch gerade - rechtwinklig
LSTRAA	1	Gerade - beliebige Ausrichtung
LSTROR	3	Gerade - rechtwinklig (nicht automatisch)

MetaItem

Zweck
Ermöglicht den Zugriff auf die MetaItem-Attribute.

Parameter
Schreibgeschützt

Name

Zweck
Der Symbolname.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

PenColor

Zweck
Diese Eigenschaft gibt die Farbe des Symbolumrisses zurück oder legt sie fest. Der Wert für die Farbe ist ein OLE_COLOR-Wert.

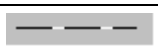
Parameter
Ein OLE_COLOR-Wert ist ein Wert vom Typ "BGR" (Blue, Green, Red - Blau, Grün, Rot). Um einen BGR-Wert anzugeben, geben Sie für "blue", "green" und "red" in der folgenden Formel einen Wert an (jeweils im Bereich von 0 bis 255).

BGR value = (blue * 65536) + (green * 256) + red

PenStyle

Zweck
Mit dieser Eigenschaft kann der Stil des Symbolumrisses zurückgegeben oder festgelegt werden.

Es gibt einen Bereich von Konstanten mit dem Präfix PEN. Diese entsprechen der Option "Format, Symbolstil, Umriss" in Rational System Architect.

Konstante	Zahl	Bedeutung
PENDASH	1	
PENDASH2DOT	4	
PENDASHDOT	3	
PENDOT	2	
PENNULL	5	Kein Umrissstil
PENSOLID1	16	
PENSOLID2	48	
PENSOLID3	64	

Konstante	Zahl	Bedeutung
PENSOLID4	128	
PENSOLID4A	384	

SAClass

Zweck
Der Klassentyp des Symbols. Wird auch als "Major Type Number" (Nummer des übergeordneten Typs) bezeichnet.

Parameter
Datentyp: lang

Schreibgeschützt

Anmerkung: Für Symbole ist dieser Wert immer "2".

SAType

Zweck
Die numerische Konstante des Symbols.

Parameter
Datentyp: lang

Schreibgeschützt

Selected

Zweck
Gibt an, ob das Symbol im Diagramm hervorgehoben ist.

Parameter
Datentyp: boolesch

ToCardinality

Zweck

Für Beziehungslinien in einem Entitätenbeziehungsdiagramm oder Bedingungen in einem physischen Datenmodell kann die Kardinalität am Linienende mit dieser Eigenschaft bestimmt und festgelegt werden.

Die Konstanten werden wie folgt festgelegt oder zurückgegeben:

Konstante	Zahl	Bedeutung
CARDINALITYMULT	0	Viele
CARDINALITYNOTUSED	1	Keine Kardinalität
CARDINALITYONEMULT	2	Ein oder mehrere
CARDINALITYONLYONE	3	Nur ein
CARDINALITYUNKNOWN	4	Nicht festgelegt
CARDINALITYZERO	5	Null
CARDINALITYZEROONE	6	Null oder ein
CARDINALITYZEROONEMULT	7	Ein, null oder mehrere

TunnelAtEnd

Zweck

Für Liniensymbole in IDEF0-Funktionsdiagrammen kann die "Tunnelung" am Ende eines Pfeils mit dieser Eigenschaft bestimmt und festgelegt werden.

Parameter

Datentyp: boolesch

TunnelAtStart

Zweck
Für Liniensymbole in IDEF0-Funktionsdiagrammen kann die "Tunnelung" am Anfang eines Pfeils mit dieser Eigenschaft bestimmt und festgelegt werden.

Parameter
Datentyp: boolesch

TypeName

Zweck
Der Typname des Symbols als Zeichenfolge, z. B. "Entität":

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

UpdateDate

Zweck
Das Datum, an dem das Symbol zuletzt geändert wurde.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

UpdateTime

Zweck
Die Uhrzeit, zu der das Symbol zuletzt geändert wurde.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

Xpos

Zweck
Die Horizontale (X-Koordinate) des Symbols in hundertstel Zoll.

Dies gilt für die Position in der unteren rechten Ecke des Symbols; Startpunkt ist die linke Seite des Diagramms.

Parameter
Datentyp: lang

Xsize

Zweck
Die Breite (X-Achse) des Symbols. Die Maßeinheit ist hunderstel Zoll.

Parameter
Datentyp: lang

Ypos

Zweck
Die Vertikale (Y-Koordinate) des Symbols in hunderstel Zoll.

Dies gilt für die Position in der unteren rechten Ecke des Symbols; Startpunkt ist die Oberkante des Diagramms.

Parameter
Datentyp: lang

Ysize

Zweck
Die Höhe (Y-Achse) des Symbols. Die Maßeinheit ist hunderstel Zoll.

Parameter
Datentyp: lang

Methoden

ConnectFrom

Zweck
Wird verwendet, um ein Symbol am Linienanfang zu verbinden.

Syntax
```
Symbol Object.ConnectFrom Line
```

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse, von der aus das Symbol verbunden wird.

`Line`

Verwendung: erforderlich

Datentyp: Symbol

Jedes instanziierte Liniensymbol

Beispiel
```
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

Zweck
Wird verwendet, um ein Symbol am Linienende zu verbinden.

Syntax
```
Symbol Object.ConnectTo Line
```

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse, mit der das Liniensymbol verbunden wird.

`Line`

Verwendung: erforderlich

Datentyp: Symbol

Jedes instanziierte Liniensymbol

Beispiel
```
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
```

Delete

Zweck
Löscht das Symbol aus dem Diagramm. Wenn ein Diagramm geschlossen wird, werden alle Symbole darin automatisch gelöscht, es sei denn, das Diagrammobjekt wird gespeichert.

GetChildDiagrams

Zweck
Durch diese Methode wird eine SAObjects-Objektgruppe von Diagrammen für ein bestimmtes Symbol an der Stelle abgerufen, an der untergeordnete Elemente angehängt sind.

Die SAObjects-Objektgruppe wird erst dann vollständig ausgefüllt, wenn das Flag "Complete" für die Objektgruppe "True" ist. "GetAllSymbols" sollte zusammen mit der ReadAll- oder der IsMoreThan-Methode verwendet werden.

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

GetField

Zweck
Dies sind die Merkmale des Symbols, wie z. B. "Font Type" (Schriftarttyp) oder "Font Height" (Schriftarthöhe), von denen einige als "Font Type" (Schriftarttyp) festgelegt werden können und andere, wie z. B. "AuditID", nicht.

Syntax
```
Symbol Object. GetField FieldID
```

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse

`FieldID`

Verwendung: erforderlich

Datentyp: SYMBOLFIELDS

Symbolfeld; eine vollständige Liste aller Symbolfelder finden Sie unten.

Get Metric

Zweck

Ruft bestimmte Listen, Berechnungen und Teile von interner Funktionalität, die Symbole betreffen, auf.

Syntax

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

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse

`Metric`

Verwendung: erforderlich

Data Type: SYMBOLMETRIC

Symbolmessdaten; eine vollständige Liste aller Symbolmessdaten finden Sie unten.

`FieldType`

Verwendung: optional

Datentyp: FLDTYPE

Feldtyp; eine vollständige Liste der Rational System Architect-Feldtypen finden Sie in Kapitel 17.

`NbrChars`

Verwendung: optional

Datentyp: lang

Wenn der Feldtyp eingegeben wurde, teilt dieser Parameter SA mit, wie viele Zeichen vor dem Dezimalzeichen zurückzugeben sind.

`NbrDec`

Verwendung: optional

Datentyp: lang

Wenn der Feldtyp eingegeben wurde, teilt dieser Parameter SA mit, wie viele Stellen nach dem Dezimalzeichen zurückzugeben sind.

GetProperty

Zweck

Dies gibt den Eigenschaftsinhalt für eine bestimmte Symboleigenschaft zurück. Eine vollständige Liste aller Eigenschaftsnamen finden Sie in den Dateien URSPROPS.TXT und SAPROPRS.TXT.

Parameter

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einer Junction eines IDEF3-Prozessfluss-/OV-6a-Diagramms ist die Eigenschaft "Logic" (Logik) eine Umbenennung - die Eigenschaft heißt eigentlich "Junction Logic" (Junctionlogik). Dies können Sie nur wissen, wenn Sie die Definition einer Junction in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Junction Logic" heißt, aber als "Junction Logic" bezeichnet wurde.

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

Syntax

`Symbol Object.`**`GetProperty`** `Name`

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

GetPropertyAsCollection

Zweck

Manche Eigenschaften definieren Beziehungen zu anderen Eigenschaften. Eine Entität kann z. B. über die zugehörige Eigenschaft "Modell" auf das zugehörige Modell verweisen. Durch diese Methode wird eine Objektgruppe von OneOf- und ListOf-Diagrammen oder -Definitionen zurückgegeben. Weitere Informationen zu OneOf- und ListOf-Eigenschaftstypen finden Sie in Kapitel 14.

Parameter

Datentyp: OfCollection

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einer Zusammenführung eines IDEF3-Prozessfluss-/OV-6a-Diagramms ist die Eigenschaft "Logic" (Logik) eine Umbenennung - die Eigenschaft heißt eigentlich "Junction Logic" (Junctionlogik). Dies können Sie nur wissen, wenn Sie die Definition einer Junction in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Junction Logic" heißt, aber als "Junction Logic" bezeichnet wurde.

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

Syntax

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

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse

`PropName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

Beispiel

```
Dim Models As OfCollection

   Set Models =
  SASym.GetPropertyAsCollection("Model")
```

GetRelatedObjects

Zweck

Durch diese Methode wird eine SAObjects-Objektgruppe von Objekten zurückgegeben, die zum aktuellen Symbolobjekt gehören.

Syntax

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

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse

`RelType`

Verwendung: erforderlich

Datentyp: RELATETYPE

SA-Beziehung; eine vollständige Liste aller Beziehungen finden Sie in Kapitel 16.

Beispiel

```
Dim oCollectionOfRelatedItems As SAObjects

Set oCollectionOfRelatedItems =
  oSymbol.GetRelatedObjects(RELCONNEND)

oCollectionOfRelatedItems.ReadAll
```

Save

Zweck
Um das Symbol nach der Erstellung in einem Diagramm zu speichern, rufen Sie die **Save-**Methode auf.

Beispiel
```
oSymbol.Save
```

SetField

Zweck
Dadurch werden Feldwerte für ein Symbol festgelegt. Es sind zwei Argumente erforderlich, nämlich das Feld und der zugehörige Wert.

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

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse

`FieldID`

Verwendung: erforderlich

Datentyp: SYMBOLFIELD

Symbolfeld; eine vollständige Liste aller Symbolfelder finden Sie unten.

`Value`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Wert des Symbolfelds

SetProperty

Zweck

Wenn Sie den Eigenschaftsnamen und den zugehörigen Wert kennen, können Sie eine Eigenschaft eines Symbols festlegen. Eine vollständige Liste aller Eigenschaftsnamen finden Sie in den Dateien URSPROPS.TXT und SAPROPRS.TXT.

Parameter

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einer Zusammenführung eines IDEF3-Prozessfluss-/OV-6a-Diagramms ist die Eigenschaft "Logic" (Logik) eine Umbenennung - die Eigenschaft heißt eigentlich "Junction Logic" (Junctionlogik). Dies können Sie nur wissen, wenn Sie die Definition einer Junction in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Junction Logic" heißt, aber als "Junction Logic" bezeichnet wurde.

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

Syntax

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

`Symbol Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Symbolklasse

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

`Value`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Wert der Symboleigenschaft

Symbolfelder

Die Eigenschaft "Symbolfeld" kann eine Reihe von Eigenschaften für das Symbol enthalten. In der Regel enthält sie Informationen, die ein Benutzer nicht direkt eingeben kann, sondern die sich aus der normalen Verwendung ergeben. Beispiele hierfür sind "Update Time" (Aktualisierungszeit), "Audit" (Prüfung), "Position", "Size" (Größe) und "TypeName" (Typname).

Im Objektmodell gibt es einen Aufzählungstyp namens *SYMBOLFIELD*. Dieser wird als Parameter an die Operationen **GetField(FieldID as *SYMBOLFIELD*)** und **SetField(FieldID as *SYMBOLFIELD*, Value as String)** übergeben. Dies ermöglicht es dem VPA-Programmierer, Symbolfelder der unteren Ebene sowohl zu lesen als auch zu aktualisieren.

SYMBOLFIELD-Konstante	Beschreibung	Datentyp	
SYMFLD_AUDITID	Prüf-ID des Symbols	Zeichenfolge Schreibgeschützt	
SYMFLD_BARRANGMENT	Anordnung der Symbolverzeichnisstruktur	“0” = Untergeordnete Elemente horizontal anordnen “1” = Untergeordnete Elemente vertikal anordnen “2” = Untergeordnete Elemente als Block anordnen	
SYMFLD_BOTHERSYMBOLOGY	Zeigt eine alternative Form des Symbols (z. B. das Symbolstereotyp) an.	Boolescher Wert	
SYMFLD_CBGCOLOR	Hintergrundfarbe eines Symbols in einem Zeichenanzeigediagramm	“0” = Schwarz “1” = Blau “2” = Grün “3” = Zyan	“4” = Rot “5” = Magenta “6” = Braun/Gelb “7” = Weiß

SYMBOLFIELD-Konstante	Beschreibung	Datentyp	
SYMFLD_CFGCOLOR	Vordergrundfarbe eines Symbols in einem Zeichenanzeigediagramm	“0” = Schwarz “1” = Blau “2” = Grün “3” = Zyan	“4” = Rot “5” = Magenta “6” = Braun/Gelb “7” = Weiß
SYMFLD_COCCOFFSET	Bestimmt den Abstand zwischen den einzelnen Iterationen von Werten für einige Eingabefelder in einem Zeichenanzeigediagramm.	Numerisch	
SYMFLD_COCCURS	Ermöglicht mehrere Iterationen von Werten für einige Eingabefelder in einem Zeichenanzeigediagramm.	Numerisch	
SYMFLD_COMMENT	Legt die Eigenschaft "Grafikkommentar" des Symbols fest.	Zeichenfolge	
SYMFLD_CPROMPT	Cobalt-Eingabeaufforderungszeichen des Symbols in einem Zeichenanzeigediagramm	1-Bit-Zeichenfolge	
SYMFLD_CUNCLECOUNT	Gibt die Anzahl der Symbole, die direkt an das übergeordnete Symbol angehängt sind, nach Datenflusslinien zurück.	Hexadezimal Schreibgeschützt	
SYMFLD_DDCOMMENT	Datenverzeichnis-ID des Kommentarsymbols des Symbols	Numerisch Schreibgeschützt	

SYMBOLFIELD-Konstante	Beschreibung	Datentyp
SYMFLD_DDIDENTITY	Datenverzeichnis-ID des Symbols	Numerisch Schreibgeschützt
SYMFLD_DESCLOC	Position des Grafik-kommentars des Symbols	“XPos YPos” Numerisch
SYMFLD_DESCSIZE	Größe des Grafikkommentars des Symbols	“XSize YSize” Numerisch
SYMFLD_DWSTYLE	Spiegelt wider, welche Optionen der Benutzer für ein Grafikanzeigesymbol in einem Grafikanzeige-diagramm ausgewählt hat.	Hexadezimal
SYMFLD_ENDLOC	Position des Liniensymbolendes	“XPos YPos” Numerisch
SYMFLD_ERROR1	Erster von Rational System Architect erkannter Fehler	Fehlernummer
SYMFLD_ERROR2	Zweiter von Rational System Architect erkannter Fehler	Fehlernummer
SYMFLD_FONTFLAGS	Aktiviert Fettschrift, Kursivschrift, Unterstreichung oder Durchstreichung für die Symbolschriftart.	Hexadezimal “0x0002” = Fettschrift “0x0005” = Kursivschrift “0x000B” = Unterstreichung “0x0010” = Durchstreichung
SYMFLD_FONTHEIGHT	Symbolschriftartgröße	Hexadezimal
SYMFLD_FONTNAME	Symbolschriftartname (z. B. Arial, Times New Roman, usw.)	Zeichenfolge

SYMBOLFIELD-Konstante	Beschreibung	Datentyp
SYMFLD_FREXARCCHAR	Fügt einen ausschließlichen Bogen am Anfang eines Liniensymbols ein.	Boolescher Wert
SYMFLD_FROMCARDINALITY	Gibt den Namen für "FromCardinality" zurück (z B. "Nur ein", "Ein oder mehrere" usw.). Eine vollständige Liste der Werte für "FromCardinality" finden Sie in den Informationen zum Attribut "FromCardinality" im Abschnitt zu den Symbol-klassenattributen.	Zeichenfolge Schreibgeschützt
SYMFLD_FROMCARDNUMBER	Gibt die Konstantennummer für "FromCardinality" zurück. Eine vollständige Liste der Werte für "FromCardinality" finden Sie in den Informationen zum Attribut "FromCardinality" im Abschnitt zu den Symbol-klassenattributen.	Numerisch Schreibgeschützt
SYMFLD_FROMCONNECTCOMPASSPOINT	Gibt N,E,S oder W als Ausgangspunkt eines ICOM-Pfeils zurück, der am Linienanfang verbunden ist.	Zeichenfolge Schreibgeschützt
SYMFLD_HASFROMARROW	Gibt "True" zurück, wenn die Linie am Linienanfang einen Pfeil aufweist.	Boolescher Wert Schreibgeschützt
SYMFLD_HASFROMTUNNEL	Gibt "True" zurück, wenn der ICOM-Pfeil am Linienanfang "getunnelt" ist.	Boolescher Wert Schreibgeschützt
SYMFLD_HASTOARROW	Gibt "True" zurück, wenn die Linie am Linienende einen Pfeil aufweist.	Boolescher Wert Schreibgeschützt

SYMBOLFIELD-Konstante	Beschreibung	Datentyp
Gibt "True" zurück, wenn der ICOM-Pfeil am Linienende "getunnelt" ist.	Gibt "True" zurück, wenn der ICOM-Pfeil am Linienende "getunnelt" ist.	Boolescher Wert Schreibgeschützt
SYMFLD_LINESTYLE	Symbollinienstil	Hexadezimal, 4 Byte
SYMFLD_LOC	Position des Symbols im Diagramm	“XPos YPos” Numerisch
SYMFLD_NAME	Symbolname	Zeichenfolge
SYMFLD_NAMECRLF	Name des Symbols mit Zeilenumbruch. Im Namensfeld kann der Benutzer bis zu fünf Zeilen Text eingeben.	Zeichenfolge
SYMFLD_NAMECRLF1	Wenn der Name als Zeichenfolge fortlaufenden Texts angezeigt würde (z. B. "JimJaneTomLouRon") die Zeichenzahl, bei der der Text in der zweiten Zeile beginnt (z. B. 4).	Numerisch
SYMFLD_NAMECRLF2	Wenn der Name als Zeichenfolge fortlaufenden Texts angezeigt würde (z. B. "JimJaneTomLouRon") die Zeichenzahl, bei der der Text in der dritten Zeile beginnt (z. B. 8).	Numerisch
SYMFLD_NAMECRLF3	Wenn der Name als Zeichenfolge fortlaufenden Texts angezeigt würde (z. B. "JimJaneTomLouRon") die Zeichenzahl, bei der der Text in der vierten Zeile beginnt (z. B. 11).	Numerisch

SYMBOLFIELD-Konstante	Beschreibung	Datentyp
SYMFLD_NAMECRLF4	Wenn der Name als Zeichenfolge fortlaufenden Texts angezeigt würde (z. B. "JimJaneTomLouRon") die Zeichenzahl, bei der der Text in der fünften Zeile beginnt (z. B. 14).	Numerisch
SYMFLD_NAMELOC	Position des Symbolnamenfelds	“XPos YPos” Numerisch
SYMFLD_NAMESIZE	Größe des Symbolnamensfelds	“XSize YSize” Numerisch
SYMFLD_ORDER	Anordnung des Assoziationsendes	“0” = Ungeordnet “1” = Geordnet “2” = Sortiert
SYMFLD_PENSTYLE	Symbolumrissstil und -breite	Hexadezimal, 4 Byte
SYMFLD_ROTATION	Dreht das Flagsymbol im Strukturdiagramm.	Numerische Werte von 0 bis 31 drehen des Flagsymbol im Uhrzeigersinn. Beispiel: “0” = Süden “8” = Westen “16” = Norden “24” = Osten
SYMFLD_SAMAJORTYPE	Übergeordneter Typ (z. B. Symbol)	Zeichenfolge Schreibgeschützt
SYMFLD_SAMAJORTYPENUMBER	Nummer des übergeordneten Typs (z. B. 2)	Numerisch Schreibgeschützt

SYMBOLFIELD-Konstante	Beschreibung	Datentyp
SYMFLD_SEQNUM	Entitätsnummer in Entitätssymbolen	Numerisch
SYMFLD_SIZE	Symbolgröße	“XSize YSize” Numerisch
SYMFLD_STARTLOC	Position, an der das Liniensymbol beginnt	“XPos YPos” Numerisch
SYMFLD_STYLEFLAGS	Aktiviert Symbolfarben	Hexadezimal “0x0001” = Umrissfarbe “0x0002” = Füllfarbe “0x0004” = Schriftfarbe
SYMFLD_SUPERSUB	Legt den Wert für die Beziehung des Symbols (übergeordnet oder untergeordnet) fest.	“0” = Keines von beidem “1” = Übergeordnet “2” = Untergeordnet
SYMFLD_TEXTFLAGS	Texteigenschaften des Symbols	Hexadezimal
SYMFLD_TOCARDINALITY	Gibt den Namen für "ToCardinality" (z. B. "Genau ein", "Ein oder mehrere" usw.) zurück. Eine vollständige Liste der Werte für "ToCardinality" finden Sie in den Informationen zum Attribut "ToCardinality" im Abschnitt zu den Symbolklassenattributen.	Zeichenfolge Schreibgeschützt

SYMBOLFIELD-Konstante	Beschreibung	Datentyp
SYMFLD_TOCARDNUMBER	Gibt die Konstantennummer für "ToCardinality" zurück. Eine vollständige Liste der Werte für "ToCardinality" finden Sie in den Informationen zum Attribut "ToCardinality" im Abschnitt zu den Symbolklassen-attributen.	Numerisch Schreibgeschützt
SYMFLD_TOCONNECTCOMPASSPOINT	Gibt N,E,S oder W für den Ausgangspunkt eines ICOM-Pfeils, der am Linienende verbunden ist, zurück.	Zeichenfolge Schreibgeschützt
SYMFLD_TOEXARCCHAR	Fügt einen ausschließlichen Bogen am Ende eines Liniensymbols ein.	Boolescher Wert
SYMFLD_TYPE	SA-Symboltyp	Interne Konstantennummer von SAType
SYMFLD_TYPENAME	Name des SA-Symboltyps (z. B. "Entität", "ICOM-Pfeil" usw.)	Zeichenfolge Schreibgeschützt
SYMFLD_U_S1_WPICTYPE	Bildtyp (zum Diagramm hinzugefügte Grafikdatei)	Zeichenfolge Schreibgeschützt
SYMFLD_U_S1_ZPPICFILE	Pfadname der Datei, die zum Anzeigen des Bilds verwendet wird	Zeichenfolge Schreibgeschützt
SYMFLD_UPDATEDATE	Datum der letzten Aktualisierung	Datumsfeld Schreibgeschützt
SYMFLD_UPDATEDATEINTL	Datum der letzten Aktualisie-rung (internationales Format)	Datumsfeld Schreibgeschützt

SYMBOLFIELD-Konstante	Beschreibung	Datentyp
SYMFLD_UPDATETIMEINTL	Uhrzeit der letzten Aktualisierung (internationales Format)	Zeitfeld Schreibgeschützt
SYMFLD_XPENTITY	Interne Nummer für das Symbol	Numerisch Schreibgeschützt
SYMFLD_XPGROUP	Interne Nummer für das übergeordnete Symbol	Numerisch Schreibgeschützt
SYMFLD-XPLINK	Interne Nummer des Symbols, mit dem das ausgewählte Symbol verbunden ist (z. B. ein Verweis, der in einem IDEF3-Prozessflussdiagramm mit einer Verhaltenseinheit verbunden ist)	Numerisch Schreibgeschützt
SYMFLD_XPSIBLING	Interne Nummer des nächsten der fortlaufenden gleichgeordneten Elemente	Numerisch Schreibgeschützt
SYMFLD_XPSUBORDINATE	Interne Nummer für das erste untergeordnete Symbol	Numerisch Schreibgeschützt
SYMFLD_ZPDESC	Legt den Grafikkommentar des Symbols fest.	Zeichenfolge
SYMFLD-ZPSSADMSTR	Unbekannt	

Symbolmessdaten

In der Vergangenheit wurden Messwerte verwendet, um Listen zu erstellen, Regelprüfungen auszuführen und Berechnungen für verschiedene Rational System Architect-Berichte bereitzustellen. Nun kann der Benutzer individuelle Messdaten ausführen, indem er die GetMetric-Methode in der Symbolklasse aufruft. Ein Verzeichnis aller Symbolmessdaten finden Sie in der SYMBOLMETRIC-Aufzählungsliste im SA-Objektbrowser. Im Folgenden finden Sie eine Tabelle aller verfügbaren Symbolmessdaten und der zugehörigen Beschreibungen.

Symbolmessdaten	Beschreibung
SYMMETANNOTATION	Gibt "True" zurück, wenn es sich um ein Annotations-symbol handelt ["Doc Block" (Dokumentationsblock), "Text Box" (Textfeld), "Rectangle" (Rechteck), "Page Connector" (Seitenkonnektor)].
SYMMETBALANCE	Vergleicht die Eingabe- und Ausgabelinien des Symbols mit den Eingabe- und Ausgabelinien des zugehörigen untergeordneten Prozesses. Erstellt eine Liste nicht übereinstimmender Eingabe- und Ausgabelinien mit dem Namen und dem Symboltyp. (Siehe Hilfedatei zum Neuausrichten untergeordneter Symbole.) Wenn das Symbol ein Datenspeicher, ein AND-Konnector oder ein XOR-Konnector ist, dann werden hierdurch die definierten Elemente und Strukturen dieser Symbole verglichen. Erstellt eine Liste undefinierter Elemente im Symbol und gibt an, ob die ein- und abgehenden Datenflüsse definierte Elemente aufweisen. (Siehe Hilfedatei zur horizontalen Neuausrichtung.)
SYMMETBALANCEMSPEC	Richtet die Minispezifikation der Symboldefinition neu aus. Wird für Prozesse in Datenflussdiagrammen und Module in Strukturgrafikdiagrammen verwendet. Zusätzliche Informationen hierzu finden Sie in der Rational System Architect-Hilfedatei zum Thema "Minispezifikationen".

Symbolmessdaten	Beschreibung
SYMMETBOTTOM	"Bottom" ist ein abgeleitetes boolesches Feld (T/F). Der Wert ist "True" für ein Symbol, wenn es nicht zu einem Diagramm erweitert wird.
SYMMETCHARCOUNT	Gibt die Anzahl der Zeichen in der Eigenschaft "Beschreibung" des Symbols zurück.
SYMMETCONNECTOR	Gibt "True" zurück, wenn das Symbol ein Konnektor-symbol ist (AND- oder XOR-Konnektor oder ICOM-Pfeilverknüpfung).
SYMMETCURRENT	Boolesches Feld (T/F). Gibt "True" zurück, wenn das Symbol sich im derzeit angezeigten Diagramm befindet.
SYMMETELEMENTLIST	Erstellt eine Liste der Elemente der unteren Ebene für die Symboldefinition. Ein Element ist ein Element der unteren Ebene, wenn es keine erweiternden Beziehungen aufweist.
SYMMETEXPRESSION	Erstellt eine Liste der Syntax mit fehlerhaften Ausdrücken oder der undefinierten Datenelemente oder -strukturen, die vom Ausdruck der Symboldefinition verwendet werden.
SYMMETICOMDEST	Gibt die Zielrolle des ICOM-Pfeils [input (Eingabe), control (Steuerung), mechanism (Mechanismus) oder boundary (Grenze)] als Zeichenfolge zurück.
SYMMETICOMSOURCE	Gibt die Quellenrolle des ICOM-Pfeils [call (Aufruf), output (Ausgabe) oder boundary (Grenze)] als Zeichenfolge zurück.
SYMMETINPUTLIST	Erstellt eine Liste der Elemente der untergeordneten Ebene, die als Eingabe für die Symboldefinition verwendet werden. Ein Element ist ein Element der unteren Ebene, wenn es keine erweiternden Beziehungen aufweist.
SYMMETISFOREIGNKEY	Gibt "True" zurück, wenn die Symboldefinition ein Fremdschlüssel ist.

Symbolmessdaten	Beschreibung
SYMMETKEYCOMPNBR	Gibt die Komponentennummer des primären Schlüssels eines Symbols zurück. Ein Benutzer kann alternativ die Komponentennummer anzeigen, indem er die Attributliste der Symboldefinition im Browserdetail (z. B. @1, @2 usw.) erweitert.
SYMMETLEVELNUMBER	Gibt die Nummer, die die hierarchische Position des Symbols (z. B. 5.3, 5.3.1 usw.) beschreibt, als Zeichenfolge zurück.
SYMMETLEVELNUMBERSORT	Gibt die Nummer, die die hierarchische Position des Symbols beschreibt, als Zeichenfolge zurück. Jede Nummer enthält 3 Stellen (z. B. 003.005.002). So kann der Benutzer die Ergebnisse besser sortieren.
SYMMETLINECOUNT	Gibt die Anzahl der Linien in der Eigenschaft "Beschreibung" einer Symboldefinition zurück.
SYMMETNORMALIZE1	Führt eine Prüfung daraufhin durch, ob die Symboldefinition sich in der ersten Normalform befindet. Eine Entität befindet sich in der ersten Normalform, wenn sie keine sich wiederholenden Gruppen enthält.
SYMMETNORMALIZE23	Führt eine Prüfung daraufhin aus, ob die Symboldefinition sich in der zweiten und dritten Normalform befindet. Eine Entität befindet sich in der zweiten Normalform, wenn sie sich in der ersten Normalform befindet und jedes Attribut, bei dem es sich nicht um einen Schlüssel handelt, funktional vollständig vom Primärschlüssel abhängt. Eine Entität befindet sich in der dritten Normalform, wenn sie sich in der zweiten Normalform befindet und alle Attribute, bei denen es sich nicht um Schlüssel handelt, ausschließlich vom Primärschlüssel abhängen.
SYMMETOUTPUTLIST	Erstellt eine Liste der Elemente unterer Ebene, die als Ausgabe für die Symboldefinition verwendet wird. Ein Element ist ein Element der unteren Ebene, wenn es keine erweiternden Beziehungen aufweist.

Symbolmessdaten	Beschreibung
SYMMETPARENTSLASHDATA	Gibt Fremdschlüssel-Schrägstrichdaten der Symboldefinition zurück, die Informationen dazu enthalten, woher der Attributschlüssel stammt. Die Schrägstrichdaten eines Fremdschlüssels können auch angezeigt werden, indem Sie die Attributliste der Symboldefinition in den Browserdetails erweitern. Die Schrägstrichdaten erscheinen wie im folgenden Beispiel: Row_Number / FKFROM “Stock_Location.Row_Number(stores)” /
SYMMETREFERENCE	Gibt "True" zurück, wenn auf die Symboldefinition von einem anderen Objekt verwiesen wird.
SYMMETRULES	Führt eine Regelprüfung auf Nichteinhaltung von Standardmethodikregeln für das Symbol durch.
SYMMETSELECTED	Gibt "True" zurück, wenn das Symbol in einem derzeit geöffneten Diagramm hervorgehoben ist.
SYMMETSEQINPARENTSLIST	Überprüft die Eigenschaftengruppe des übergeordneten Objekts auf eine Liste untergeordneter Objekte hin. Gibt die Nummer zurück, unter der die Symboldefinition in der Liste erscheint.
SYMMETTOP	Gibt "False" zurück, wenn das Diagramm, in dem sich das ausgewählte Symbol befindet, ein untergeordnetes Element eines anderen Symbols ist. Gibt "True" zurück, wenn das Diagramm keine Erweiterung eines anderen Symbols ist.
SYMMETUNMARKEDLIST	Erstellt eine Liste von Elementen der unteren Ebene, die von der Definition aller Liniensymbole, die weder als Eingabe noch als Ausgabe gekennzeichnet sind, verwendet wird (keine Pfeilspitzen). Ein Element ist ein Element der unteren Ebene, wenn es keine erweiternden Beziehungen aufweist.
SYMMETUPDATEUSES	Aktualisiert die Beziehungstabelle des Datenverzeichnisses (RELATN.DBF). Weist keinen Rückgabewert auf.

Symbolmessdaten	Beschreibung
SYMMETWORDCOUNT	Gibt die Anzahl der Wörter in der Eigenschaft "Beschreibung" der Symboldefinition zurück.

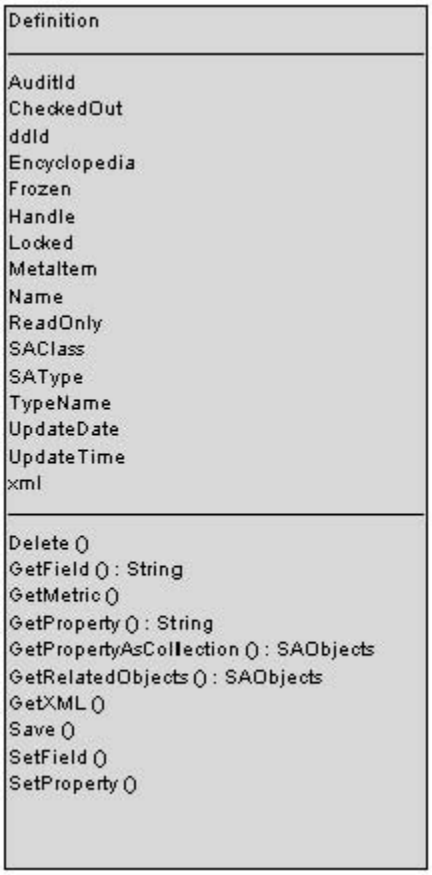
8

Die Definitionsklasse

In diesem Kapitel enthaltene Themen	Seite
Attribute	8-3
Methoden	8-8
Felder	8-16
Messdaten	8-18

Einführung

Hierbei handelt es sich um die Definitionsklasse mit den unten dargestellten Attributen und Methoden.



Attribute

AuditID

Zweck
Alle in Rational System Architect gespeicherten Elementdefinitionen sind mit der ID der Person gekennzeichnet, die die Definition erstellt oder zuletzt geändert hat. Die ID ist in der Definition als "AuditId" codiert.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

CheckedOut

Zweck
Wenn "True" festgelegt ist, wird die Definition für alle, außer für den Benutzer mit der AuditID, der sie ausgecheckt hat, schreibgeschützt.

Parameter
Datentyp: boolesch

ddID

Zweck
Alle in Rational System Architect gespeicherten Definitionen sind intern eindeutig durch die Verwendung einer Datenverzeichnis-ID gekennzeichnet.

Parameter
Datentyp: lang

Schreibgeschützt

Encyclopedia

Zweck
Ermöglicht den Zugriff auf die Attribute und Methoden der übergeordneten Enzyklopädieklasse.

Parameter
Schreibgeschützt

Frozen

Zweck
Der Benutzer muss über die Berechtigung zum Blockieren verfügen, um dieses Attribut festzulegen. Wenn "True" festgelegt ist, wird die Definition für jeden Benutzer schreibgeschützt, auch für den mit der AuditID, der die Definition blockiert hat.

Parameter
Datentyp: boolesch

Handle

Zweck
Hierbei handelt es sich um die Speicherkennung der Definition, die nur zur Laufzeit verfügbar ist. Diese Kennung ist nicht eindeutig und kann bei jedem Zugriff darauf unterschiedlich sein.

Parameter
Datentyp: lang

Schreibgeschützt

Beispiel

```
Dim Handle As Long

Handle = oDefinition.Handle
```

Locked

Zweck
Gibt "True" oder "False" zurück, um anzugeben, ob eine Definition gesperrt ist oder nicht, d. h., ob sie von einem Benutzer verwendet wird oder nicht. Hierfür ist ein gültiges Definitionsobjekt erforderlich.

Parameter
Datentyp: boolesch

Schreibgeschützt

MetaItem

Zweck
Ermöglicht den Zugriff auf die MetaItem-Attribute.

Parameter
Schreibgeschützt

Name

Zweck
Der Name des angegebenen Definitionsobjekts.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

ReadOnly

Zweck
Gibt an, ob die Definition schreibgeschützt ist.

Parameter
Datentyp: boolesch

Schreibgeschützt

SAClass

Zweck
Der Klassentyp der Definition. Wird auch als "Major Type Number" (Nummer des übergeordneten Typs) bezeichnet.

Parameter
Datentyp: lang

Schreibgeschützt

Anmerkung: Hierdurch wird für eine Definition immer "3" zurückgegeben.

SAType

Zweck
Die konstante Ganzzahl der Definition. Alle Definitionen in Rational System Architect weisen eine eindeutige numerische Konstanten-ID auf.

Parameter
Datentyp: lang

Schreibgeschützt

TypeName

Zweck
Der Typ der Definition als Zeichenfolge, z. B. "Prozess"

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

UpdateDate

Zweck
Das Datum, an dem die Definition zuletzt geändert wurde.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

UpdateTime

Zweck
Die Uhrzeit, zu der die Definition zuletzt geändert wurde.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

xml

Zweck
Die XML-Zeichenfolge der Definition. Bearbeitet durch die GetXML-Methode.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

Methoden

Delete

Zweck
Löscht eine durch ein Definitionsobjekt angegebene Definition.

Beispiel
```
Call oDefinition.Delete
```

GetField

Zweck
Hierbei handelt es sich um Merkmale der Definition, wie z. B. "Type Number" (Typennummer), "Undefined Flag" (Nicht definiertes Flag), "Major Type Name" (Name des übergeordneten Typs).

Syntax
```
Definition Object. GetField FieldID
```

`Definition Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Definitionsklasse

`FieldID`

Verwendung: erforderlich

Datentyp: DEFFLD

Definitionsfeld; eine vollständige Liste aller Definitionsfelder finden Sie unten.

Get Metric

Zweck

Ruft bestimmte Listen, Berechnungen und Teile von interner Funktionalität, die Diagramme betreffen, ab.

Syntax

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

`Definition Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Definitionsklasse

`Metric`

Verwendung: erforderlich

Datentyp: DEFINITIONMETRIC

Definitionsmessdaten; eine vollständige Liste aller Definitionsmessdaten finden Sie unten.

`FieldType`

Verwendung: optional

Datentyp: FLDTYPE

Feldtyp; eine vollständige Liste der Rational System Architect-Feldtypen finden Sie in Kapitel 17.

`NbrChars`

Verwendung: optional

Datentyp: lang

Wenn der Feldtyp eingegeben wurde, teilt dieser Parameter SA mit, wie viele Zeichen vor dem Dezimalzeichen zurückzugeben sind.

`NbrDec`

Verwendung: optional

Datentyp: lang

Wenn der Feldtyp eingegeben wurde, teilt dieser Parameter SA mit, wie viele Stellen nach dem Dezimalzeichen zurückzugeben sind.

GetProperty

Zweck

Dadurch wird der Inhalt der Eigenschaft für eine beliebige angegebene Definitions-eigenschaft zurückgegeben.

Parameter

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einem Servicezeitprofil ist z. B. der Name der Eigenschaft "Time Units" (Zeiteinheiten) eine Umbenennung - die Eigenschaft heißt eigentlich "Duration Time Units" (Dauerzeiteinheiten). Dies können Sie nur wissen, wenn Sie die Definition einer Junction in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Duration Time Units" heißt, aber als "Time Units" bezeichnet wurde.

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

Syntax

`Definition Object.`**`GetProperty`** `Name`

`Definition Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Definitionsklasse

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

GetPropertyAsCollection

Zweck

Manche Eigenschaften definieren Beziehungen zu anderen Eigenschaften. Eine Entitäten-definition verweist z. B. durch ihre Eigenschaft "Beschreibung" auf Attribute. Durch diese Methode wird eine Objektgruppe von OneOf- und ListOf-Diagrammen oder -Definitionen zurückgegeben. Weitere Informationen zu OneOf- und ListOf-Eigenschaftstypen finden Sie in Kapitel 14.

Parameter

Datentyp: OfCollection

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einem Servicezeitprofil ist z. B. der Name der Eigenschaft "Time Units" (Zeiteinheiten) eine Umbenennung - die Eigenschaft heißt eigentlich "Duration Time Units" (Dauerzeiteinheiten). Dies können Sie nur wissen, wenn Sie die Definition einer Junction in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Duration Time Units" heißt, aber als "Time Units" bezeichnet wurde.

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

Syntax

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

`Definition Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Definitionsklasse

`PropName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

Beispiel

```
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

Zweck

Durch diese Methode wird eine SAObjects-Objektgruppe zugehöriger Objekte für das aktuelle Definitionsobjekt zurückgegeben.

Syntax

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

`Definition Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Definitionsklasse

`RelType`

Verwendung: erforderlich

Datentyp: RELATETYPE

SA-Beziehung; eine vollständige Liste aller Beziehungen finden Sie in Kapitel 16.

GetXML

Zweck
Exportiert die XML-Zeichenfolge der Definition in einen gültigen XML-Dateinamen.

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

`Definition Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Definitionsklasse

`StrXMLTextOut`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Eine gültige XML-Datei, in die SA die XML-Zeichenfolge der Definition exportiert.

Save

Zweck
Wird zum Speichern einer Instanz einer Definition verwendet.

Beispiel
```
Call oDefinition.Save
```

SetField

Zweck
"SetField" ermöglicht das Festlegen eines Definitionsfelds mit einem angegebenen Wert.

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

`Definition Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Definitionsklasse

`FieldID`

Verwendung: erforderlich

Datentyp: DGMFLD

Definitionsfeld; eine vollständige Liste aller Definitionsfelder finden Sie unten.

`Value`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Wert des Definitionsfelds

SetProperty

Zweck

Die Einstellung eines Werts für eine Definitionseigenschaft erfordert, dass der Name der Eigenschaft als erstes Argument und der Wert als zweites Argument festgelegt wird. Die Eigenschaftsnamen finden Sie in den Dateien SAPROPS.CFG und USRPROPS.TXT.

Parameter

Oft ist der tatsächliche Name einer Eigenschaft nicht der Name, der in einem Definitionsdialogfenster angezeigt wird. In einem Servicezeitprofil ist z. B. der Name der Eigenschaft "Time Units" (Zeiteinheiten) eine Umbenennung - die Eigenschaft heißt eigentlich "Duration Time Units" (Dauerzeiteinheiten). Dies können Sie nur wissen, wenn Sie die Definition einer Junction in der Datei SAPROPS.CFG nachgesehen haben und gesehen haben, dass die Eigenschaft eigentlich "Duration Time Units" heißt, aber als "Time Units" bezeichnet wurde.

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

Syntax

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

`Definition Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte Definitionsklasse

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Name der Eigenschaft, wie in der Datei SAPROPS.CFG erfasst.

`Value`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Eigenschaft "Value" der Definition"

Definitionsfelder

Die Eigenschaft "Definition Field" kann eine Reihe von Eigenschaften zur Definition enthalten. Sie enthalten in der Regel Informationen, die ein Benutzer nicht direkt eingeben kann, sondern die sich aus der normalen Verwendung ableiten. Im Objektmodell gibt es einen Aufzählungstyp namens *DEFFLD*. Dieser wird als Parameter an die Operationen **GetField(FieldID as *DEFFLD*)** und **SetField(FieldID as *DEFFLD*, Value as String)** übergeben. Dadurch kann der VBA-Programmierer Definitionsfelder der unteren Ebene sowohl lesen als auch aktualisieren.

DEFFLD constant	Beschreibung	Datentyp
DEFNFLD_SAAUDITID	Prüf-ID der Definition	Schreibgeschützt
DEFNFLD_SAIDENTITY	Datenverzeichnis-ID der Definition	Schreibgeschützt
DEFNFLD_SAIDENTITY4	Datenverzeichnis-ID der Definition	Binärzahl (4 Byte) Schreibgeschützt
DEFNFLD_SAISUNDEFINED	Gibt "T" zurück, wenn die Definition nicht definiert ist, und "F", wenn die Definition definiert ist.	Schreibgeschützt
DEFNFLD_SALOCK	Sperrt die Definition.	“T” = Gesperrt “F” = Entsperrt
DEFNFLD_SAMAJORTYPE	Übergeordneter Typ (z. B. "Definition")	Schreibgeschützt
DEFNFLD_SAMAJORTYPENUMBER	Nummer des übergeordneten Typs (z. B. "3")	Schreibgeschützt
DEFNFLD_SANAME	Name der Definition	Zeichenfolge
DEFNFLD_SATYPE	SA-Typ der Definition (z. B. "UML-Klasse", "Entität" usw.)	Schreibgeschützt

DEFFLD constant	Beschreibung	Datentyp
DEFNFLD_SATYPENUMBER	Interne Konstantennummer des Definitionstyps. Eine vollständige Liste finden Sie in der Datei DEFNS.BAS im SA-Verzeichnis.	Schreibgeschützt
DEFNFLD_SAUPDATEDATE	Datum der letzten Aktualisierung.	Schreibgeschützt
DEFNFLD_SAUPDATETIME	Uhrzeit der letzten Aktualisierung.	Schreibgeschützt

Definitionsmessdaten

In der Vergangenheit wurden Messwerte verwendet, um Listen zu erstellen, Regelprüfungen auszuführen und Berechnungen für verschiedene Rational System Architect-Berichte bereitzustellen. Nun können Benutzer individuelle Messdaten ausführen, indem sie die GetMetric-Methode in der Definitionsklasse ausführen. Ein Verzeichnis aller Definitionsmessdaten finden Sie in der DEFINITIONMETRIC-Aufzählungsliste im SA-Objektbrowser. Im Folgenden finden Sie eine Tabelle aller verfügbaren Definitionsmessdaten und der zugehörigen Beschreibungen.

Definitionsmessdaten	Beschreibung
DEFMETBOTTOM	"Bottom" ist ein abgeleitetes boolesches Feld (T/F). Der Wert ist "True" für Definitionen, die keine Ausdrücke sind, sowie für Ausdrücke, die keine Datenelemente oder Datenstrukturen enthalten.
DEFMETCURRENT	Boolesches Feld (T/F). Gibt "True" zurück, wenn des Symbol der Definition sich in einem derzeit angezeigten Diagramm befindet.
DEFMETEXPRESSION	Erstellt eine Liste der fehlerhaften Ausdruckssyntax oder der undefinierten Datenelemente oder Datenstrukturen, die vom Ausdruck der Definition verwendet werden.
DEFMETISFOREIGNKEY	Gibt "True" zurück, wenn es sich bei der Definition um einen Fremdschlüssel handelt.
DEFMETKEYCOMPNBR	Gibt die Komponentennummer des Primärschlüssels einer Definition zurück. Ein Benutzer kann alternativ die Komponentennummer anzeigen, indem er die Attributliste der Definition im Browserdetail anzeigt (z. B. @1, @2 usw.).
DEFMETNORMALIZE1	Führt eine Prüfung daraufhin durch, ob sich die Definition in der ersten Normalform befindet. Eine Entität befindet sich in der ersten Normalform, wenn sie keine sich wiederholenden Gruppen enthält.

Definitionsmessdaten	Beschreibung
DEFMETNORMALIZE23	Führt eine Prüfung daraufhin durch, ob die Definition sich in der zweiten und dritten Normalform befindet. Eine Entität befindet sich in der zweiten Normalform, wenn sie sich in der ersten Normalform befindet und jedes Attribut, bei dem es sich nicht um einen Schlüssel handelt, funktional vollständig vom Primärschlüssel abhängt. Eine Entität befindet sich in der dritten Normalform, wenn sie sich in der zweiten Normalform befindet und alle Attribute, bei denen es sich nicht um Schlüssel handelt, ausschließlich vom Primärschlüssel abhängen.
DEFMETPARENTSLASHDATA	Gibt Fremdschlüssel-Schrägstrichdaten der Definition zurück, die Informationen dazu enthalten, woher der Attributschlüssel stammt. Die Schräg-strichdaten eines Fremdschlüssels können auch angezeigt werden, indem die Attributliste der Definition in den Browserdetails erweitert wird. Die Schrägstrichdaten erscheinen wie im folgenden Beispiel: Row_Number / FKFROM “Stock_Location.Row_Number(stores)” /
DEFMETREFERENCE	Gibt "True" zurück, wenn von einem anderen Objekt auf die Definition verwiesen wird.
DEFMETSELECTED	Gibt "True" zurück, wenn ein ausgewähltes Symbol vorhanden ist, das durch die Definition definiert wird.
DEFMETSEQINPARENTSLIST	Überprüft die Eigenschaftengruppe des über-geordneten Objekts auf eine Liste untergeordneter Objekte. Gibt die Nummer zurück, unter der die Definition in der Liste erscheint.
DEFMETSYNCHRONIZE	Wenn der Benutzer bereits angegeben hat, dass die Definition mit einem anderen Objekt synchronisiert werden sollte, führen diese Messdaten die Synchronisation aus. Die Synchronisation kann im SA2001.INI-Editor festgelegt werden.

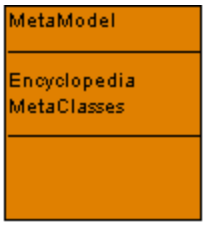
Definitionsmessdaten	Beschreibung
DEFMETUPDATEUSES	Aktualisiert die Beziehungstabelle des Daten-verzeichnisses (RELATN.DBF). Weist keinen Rückgabewert auf.
DEFNMETCHARCOUNT	Gibt die Anzahl der Zeichen in der Eigenschaft "Beschreibung" der Definition zurück.
DEFNMETLINECOUNT	Gibt die Anzahl von Linien in der Eigenschaft "Definition" einer Definition zurück.
DEFNMETWORDCOUNT	Gibt die Anzahl der Wörter in der Eigenschaft "Definition" der Definition zurück.

9

Die MetaModel-Klasse

Einführung

Das MetaModel-Objekt bietet Zugriff auf die Metaklassenobjekte für eine Enzyklopädie.



In diesem Kapitel enthaltene Themen	Seite
Attribute	9-2

Attribute

Encyclopedia

Zweck
Verweist auf das übergeordnete Enzyklopädieobjekt des Metamodellobjekts.

Parameter
Schreibgeschützt

MetaClasses

Zweck
Bietet Zugriff auf die Objektsammlung von MetaClass-Objekten für eine Enzyklopädie.

Parameter
Datentyp: SA Collection

Schreibgeschützt

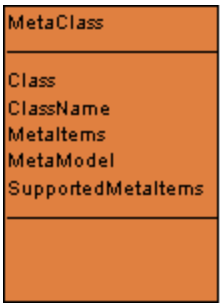
Beispiel

```
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

Die MetaClass-Klasse

Einführung
Das MetaClass-Objekt gibt Informationen zu Klassenstufen für Objekte im Repository zurück.



In diesem Kapitel enthaltene Themen	Seite
Attribute	10-2

Attribute

Klasse

Zweck
Die Klassennummer des Objekts im Repository. Die Rückgabewerte lauten "1" für ein Diagramm, "2" für ein Symbol und "3" für eine Definition. Dies wird auch als Haupttypen-nummer bezeichnet.

Parameter
Datentyp: lang

Schreibgeschützt

Beispiel
```
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

Zweck
Der Name der Klasse. Die gültigen Klassennamen in einer Enzyklopädie sind Diagramme, Symbole und Definitionen. Diese werden auch als Haupttyp bezeichnet.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

Beispiel
```
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

Zweck
Diese Eigenschaft ermöglicht den Zugriff auf die MetaItems-Klasse aller Metaelemente.

Parameter
Datentyp: SACollection

Schreibgeschützt

Beispiel

```
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

Zweck
Dies ermöglicht den Zugriff auf die übergeordnete MetaModel-Klasse.

Parameter
Schreibgeschützt

SupportedMetaItems

Zweck
Diese Eigenschaft ermöglicht den Zugriff auf lediglich diejenigen Metaelemente, die für diese Enzyklopädie aktiviert sind.

Parameter
Datentyp: SACollection

Schreibgeschützt

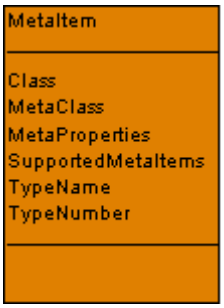
Beispiel
```
Dim sa As SA2001.Application

Set sa = New SA2001.Application

Debug.Print
  sa.Encyclopedia.MetaModel.MetaClasses.Item(1).SupportedMetaItems.Count

Set sa = Nothing
```

11

MetaItem-Klasse

Einführung

Dieses Objekt bietet Informationen zu einem einzelnen Objekttyp in der Enzyklopädie. Diese entsprechen den Informationen in den Dateien "saprops.cfg" und "usrprops.txt" für bestimmte Objekttypen.



In diesem Kapitel enthaltene Themen	Seite
Attribute	11-2

Attribute

Klasse

Zweck
Die Klasse des Objekts. Die Werte lauten "1" für ein Diagramm, "2" für ein Symbol und "3" für eine Definition. Dies wird auch als Haupttypennummer bezeichnet.

Parameter
Datentyp: lang

Schreibgeschützt

Beispiel
```
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

Zweck
Diese Eigenschaft ermöglicht den Zugriff auf das übergeordnete MetaClass-Objekt.

Parameter
Schreibgeschützt

MetaProperties

Zweck
Diese Eigenschaft ermöglicht den Zugriff auf alle Eigenschaften, die von diesem Objekttyp als Sammlung unterstützt werden.

Parameter
Datentyp: SACollection

Schreibgeschützt

Beispiel

```
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

Zweck
Diese Eigenschaft ermöglicht den Zugriff auf lediglich diejenigen Metaelemente, die für diese Enzyklopädie aktiviert sind.

Parameter
Datentyp: SACollection

Schreibgeschützt

TypeName

Zweck
Der Name des Objekttyps.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

Beispiel

```
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

Zweck

Die numerische Konstante, die diesem Objekttyp von Rational System Architect zugeordnet wurde.

Parameter

Datentyp: lang

Schreibgeschützt

Beispiel

```
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
```

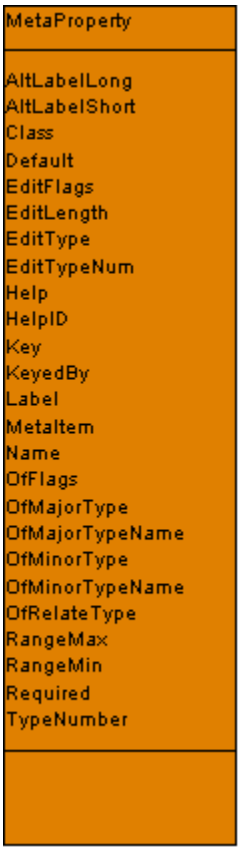
12

MetaProperty-Klasse

In diesem Kapitel enthaltene Themen	Seite
Attribute	12-3
Bearbeitungstypen	12-7

Einführung

Das MetaProperty-Objekt ermöglicht Ihnen den Zugriff auf Informationen zu den jeweiligen Eigenschaften eines bestimmten Objekts in der Enzyklopädie. Dieses Objekt entspricht dem Schlüsselwort "Eigenschaft" in den Dateien "saprops.cfg" und "usrprops.txt".



Attribute

Die folgenden Eigenschaften können alle mithilfe des folgenden Beispiels und durch Ersetzen des Namens durch eine der unten beschriebenen Eigenschaften abgerufen werden.

```
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
Die alternative Rational System Architect-Langbeschreibung für die Eigenschaft.

Datentyp: Zeichenfolge

Schreibgeschützt

AltLabelShort
Die alternative Rational System Architect-Kurzbeschreibung für die Eigenschaft.

Datentyp: Zeichenfolge

Schreibgeschützt

Class
Der übergeordnete Objektklassentyp der Eigenschaften. Dies wird auch als Haupttypennummer bezeichnet.

Datentyp: lang

Schreibgeschützt

Standard
Die Standardwertegruppe für die Eigenschaft bei der ersten Verwendung.

Datentyp: Zeichenfolge

Schreibgeschützt

EditFlags
Die Anzahl von Eigenschaften, die für eine Eigenschaft ausgefüllt werden müssen.

Datentyp: lang

Schreibgeschützt

EditLength
Die Bearbeitungslänge der Eigenschaft.

Datentyp: lang

Schreibgeschützt

EditType
Eine Liste von SAEditTypes finden Sie in der Tabelle unten.

Datentyp: Zeichenfolge

Schreibgeschützt

EditTypeNum
Eine Liste von numerischen SAEditType-Konstanten finden Sie in der Tabelle unten.

Datentyp: lang

Schreibgeschützt

Help
Der für diese Eigenschaft festgelegte Hilfetext.

Datentyp: Zeichenfolge

Schreibgeschützt

HelpID
Die ID des Hilfetexts.

Datentyp: lang

Schreibgeschützt

Key
Ob die Eigenschaft ein Schlüssel ist oder nicht.

Datentyp: boolesch

Schreibgeschützt

KeyedBy
Ob die Eigenschaft von einer anderen Eigenschaft verschlüsselt ist oder nicht.

Datentyp: SACollection

Schreibgeschützt

Label
Die Rational System Architect-Standardbeschreibung für die Eigenschaft.

Datentyp: Zeichenfolge

Schreibgeschützt

MetaItem
Ermöglicht Zugriff auf das übergeordnete MetaItem-Objekt.

Schreibgeschützt

Name
Der Name der Eigenschaft.

Datentyp: Zeichenfolge

Schreibgeschützt

OfFlags
Die Anzahl von Flags für die Eigenschaft.

Datentyp: lang

Schreibgeschützt

OfMajorType
Die numerische Konstante der übergeordneten Objektklasse.

Datentyp: lang

Schreibgeschützt

OfMajorTypeName
Der Typname der übergeordneten Objektklasse.

Datentyp: Zeichenfolge

Schreibgeschützt

OfMinorType
Der Typ der aktuell übergeordneten numerischen Konstante des Objekts.

Datentyp: lang

Schreibgeschützt

OfMinorTypeName
Der aktuelle Typname des übergeordneten Objekts.

Datentyp: Zeichenfolge

Schreibgeschützt

OfRelateType
Die Beziehungsnummer des übergeordneten Objekts. Eine vollständige Liste aller Beziehungsnummern erhalten Sie in Kapitel 15.

Datentyp: lang

Schreibgeschützt

RangeMax
Der maximale Bearbeitungsbereichswert.

Datentyp: lang

Schreibgeschützt

RangeMin
Der minimale Bearbeitungsbereichswert.

Datentyp: lang

Schreibgeschützt

Erforderlich
Die erforderliche Eigenschaft für das Objekt.

Datentyp: lang

Schreibgeschützt

TypeNumber
Die Typnummer des übergeordneten Objekts.

Datentyp: lang

Schreibgeschützt

EditType	Nummer	Beschreibung
Text	1	Bei der als Text definierten Eigenschaft kann es sich um eine Liste oder um beliebige alphanumerische Zeichen handeln, die vom Benutzer eingegeben werden.
Date	2	Die Länge der Eigenschaft muss 10 Zeichen betragen und basiert auf dem Datumsformat, das in Windows festgelegt ist.
Numeric	3	Gibt an, dass die Eigenschaft Werte enthalten muss, die Nummern sind.
Boolean	4	Die Eigenschaft verfügt über einen von zwei Werten: True (T) oder False (F); der Wert wird im Definitions-dialogfenster als Kontrollkästchen angezeigt.
Expression	5	Der Wert der Eigenschaft muss als Reihe von Zeichen-folgen eingegeben werden; die einzelnen Zeichen-folgen müssen mit einem Pluszeichen (+) getrennt werden. Dieses Wort wurde von EXPRESSIONOF ersetzt.
Minispec	6	Eine Eigenschaft, deren Wert die Verarbeitungslogik eines Prozesssymbols ausdrückt. Minispezifikationen werden mithilfe einer formalen Syntax geschrieben, die oft als "strukturiertes Englisch" bezeichnet wird.
Time	7	Die Eigenschaft enthält eine Zeitmarke in der Notation, die dem Zeitformat entspricht, das unter Windows festgelegt ist.
ListOf	8	Erstellt eine Eins-zu-eins- oder eine Eins-zu-viele-Beziehung zwischen dem aktuellen Definitionstyp und dem Definitionstyp, der nach der Einstellung "ListOf" für alle Elemente in der Listeneigenschaft benannt ist.

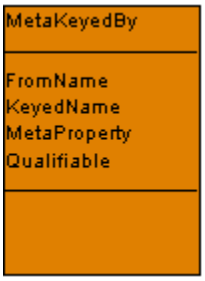
EditType	Nummer	Beschreibung
ExpressionOf	9	Erstellt eine Eins-zu-eins- oder eine Eins-zu-viele-Beziehung zwischen dem aktuellen Definitionstyp und dem Definitionstyp, der nach der Einstellung "ExpressionOf" für alle Elemente benannt ist, die in der Eigenschaft vorhanden sind.
OneOf	10	Erstellt eine Eins-zu-eins-Beziehung zwischen dem aktuellen Definitionstyp und dem Definitionstyp, der nach der Einstellung "OneOf" für das Element benannt ist, das in der Eigenschaft vorhanden ist.
TemplateOf	11	Syntax einer Triggervorlage.
ParmListOf	12	Erstellt eine Eins-zu-eins- oder eine Eins-zu-viele-Beziehung zwischen dem aktuellen Definitionstyp und dem Definitionstyp, der durch den Namen sowie durch die Parameter angegeben und nach der Einstellung "ParmListOf" für alle Elemente in der Listeneigenschaft benannt ist.
ParmOneOf	13	Erstellt eine Eins-zu-eins-Beziehung zwischen dem aktuellen Definitionstyp und dem Definitionstyp, der durch den Namen sowie durch die Parameter angegeben und nach der Einstellung "OneOf" für das Element benannt ist, das in der Eigenschaft vorhanden ist.

13

MetaKeyedBy-Klasse

Einführung

In der Abbildung unten sind die MetaKeyedBy-Klasse und die zugehörigen Attribute dargestellt.



In diesem Kapitel enthaltene Themen	Seite
Attribute	13-2

Attribute

FromName

Zweck
Die Eigenschaft "FromName" dient der Schlüsselqualifizierung einer MetaProperty.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

KeyedName

Zweck
Der Schlüsselname der MetaProperty.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

MetaProperty

Zweck
Ermöglicht Zugriff auf die übergeordnete MetaProperty-Klasse.

Parameter
Schreibgeschützt

Qualifiable

Zweck
Wert, ob die MetaProperty qualifizierbar ist, d. h. ob sie einen Verweis auf ihre Schlüsselstruktur enthält.

Parameter
Datentyp: boolesch

Schreibgeschützt

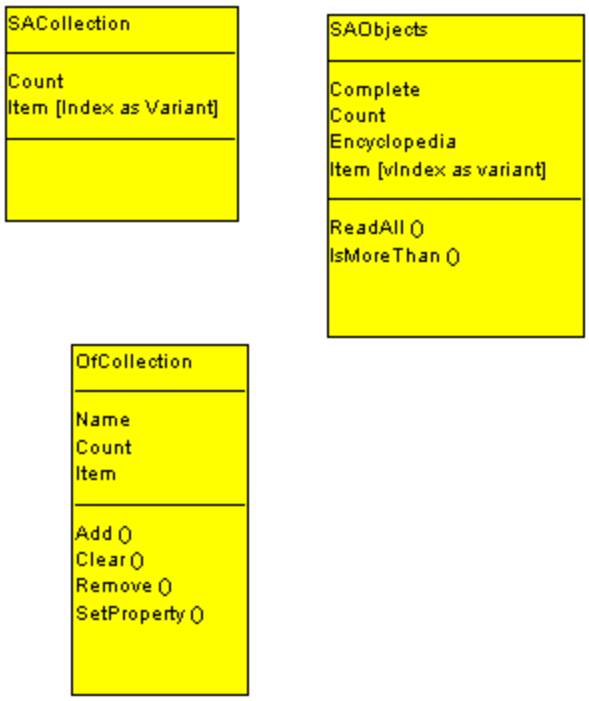
14

Rational System Architect-Objektgruppen

In diesem Kapitel enthaltene Themen	Seite
SAObjects-Klasse	14-3
SACollection-Klasse	14-6
OfCollection-Klasse	14-8

Einführung

In der folgenden Abbildung sind Rational System Architect-Objektgruppenklassen und ihre Attribute dargestellt.



SAObjects-Klasse

Diese Objektgruppenklasse wird zum Bearbeiten einer Gruppe von ähnlichen Rational System Architect-Objekten verwendet (Diagramme, Symbole oder Definitionen).

Complete

Zweck
Boolescher Wert, der "True" (Wahr) angibt, wenn alle möglichen Mitglieder in eine Objektgruppe eingelesen wurden.

Parameter
Datentyp: boolesch

Schreibgeschützt

Beispiel
```
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

Zweck
Wert, der die Anzahl von Objektgruppenmitgliedern angibt.

Parameter
Datentyp: lang

Schreibgeschützt

Beispiel

```
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

Zweck
Verweis zurück auf die Enzyklopädieklasse des aktuellen Objekts in der Objektgruppe.

Parameter
Schreibgeschützt

Item(Index)

Zweck
Die einzelnen Mitglieder der Objektgruppe verfügen über eine bestimmte Indexnummer in der Objektgruppe, die abgeleitet wird, wenn die Objektgruppe erstellt wird. Die Variable "item" ist ein Objekt, das auf dem Index der Objektgruppe basiert. Die Indexnummer oder der Indexname (sofern bekannt) kann auf dieses Objekt verweisen.

Parameter
Datentyp: boolesch

Schreibgeschützt

Beispiel

```
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)

Zweck
Gibt den Wert "True" (Wahr) zurück, wenn mehr Elemente in der Objektgruppe vorhanden sind, als der aktuelle Indexwert angibt – wird verwendet, wenn die Objektgruppen einzeln nacheinander gelesen werden.

Parameter
Datentyp: boolesch

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

ReadAll

Zweck
Liest jedes mögliche Vorkommen des Typs in der Objektgruppe.

Beispiel
```
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-Klasse

Diese Objektgruppenklasse wird zum Bearbeiten einer Gruppe von ähnlichen Rational System Architect-Eigenschaften verwendet.

Count

Zweck
Gibt einen numerischen Wert für die Anzahl von Objektgruppenmitgliedern zurück.

Parameter
Datentyp: lang

Schreibgeschützt

Beispiel
```
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)

Zweck
Die einzelnen Mitglieder der Objektgruppe verfügen über eine bestimmte Indexnummer in der Objektgruppe, die abgeleitet wird, wenn die Objektgruppe erstellt wird. Die Variable "item" ist ein Objekt, das auf dem Index der Objektgruppe basiert. Die Indexnummer oder der Indexname (sofern bekannt) kann auf dieses Objekt verweisen.

Parameter
Datentyp: boolesch

Schreibgeschützt

Beispiel

```
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-Klasse

Diese Objektgruppenklasse wird zum Bearbeiten einer Gruppe von Diagrammen oder Definitionen verwendet, die Komponenten eines besonderen Eigenschaftstyps sind, der in der unten dargestellten Tabelle aufgeführt ist.

Eigenschaftstyp	Beschreibung
ListOf	Erstellt eine Eins-zu-eins- oder eine Eins-zu-viele-Beziehung zwischen dem aktuellen Definitionstyp und dem Definitionstyp, der nach der Einstellung "ListOf" für alle Elemente in der Listeneigenschaft benannt ist.
OneOf	Erstellt eine Eins-zu-eins-Beziehung zwischen dem aktuellen Definitionstyp und dem Definitionstyp, der nach der Einstellung "OneOf" für das Element benannt ist, das in der Eigenschaft vorhanden ist.

Name

Zweck
Name der Eigenschaftenliste.

Parameter
Datentyp: Zeichenfolge

Schreibgeschützt

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

Count

Zweck
Ein numerischer Wert, der die Anzahl von Objektgruppenmitgliedern angibt.

Parameter
Datentyp: lang

Schreibgeschützt

Beispiel
```
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

Zweck
Die einzelnen Mitglieder der Objektgruppe verfügen über eine bestimmte Indexnummer in der Objektgruppe, die abgeleitet wird, wenn die Objektgruppe erstellt wird. Die Variable "item" ist ein Objekt, das auf dem Index der Objektgruppe basiert. Die Indexnummer oder der Indexname (sofern bekannt) kann auf dieses Objekt verweisen.

Parameter
Datentyp: boolesch

Schreibgeschützt

Beispiel
```
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

Zweck
Fügt ein Diagramm- oder ein Definitionsobjekt zur Eigenschaftenliste hinzu.

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

`OfCollection Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte OfCollection-Klasse

`Item`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Diagramm -oder Definitionsobjekt, das zur Eigenschaftenliste hinzugefügt werden soll.

`Before`

Verwendung: optional (kann nicht festgelegt werden, wenn der Parameter "After" bereits festgelegt ist)

Datentyp: lang

Nummer des Diagramm- oder Definitionsobjekts im Objektgruppenobjektzähler, nach dem das neue Diagramm- oder Definitionsobjekt hinzugefügt wird.

`After`

Verwendung: optional (kann nicht festgelegt werden, wenn der Parameter "Before" bereits festgelegt ist)

Datentyp: lang

Nummer des Diagramm- oder Definitionsobjekts im Objektgruppenobjektzähler, vor dem das neue Diagramm- oder Definitionsobjekt hinzugefügt wird.

Beispiel

```
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

Zweck

Löscht das OfCollection-Objekt aller Elemente in der Objektgruppe.

Beispiel

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

Remove

Zweck

Entfernt ein Diagramm- oder ein Definitionsobjekt aus der Objektgruppe.

Syntax

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

`OfCollection Object`

Verwendung: erforderlich

Datentyp: Objekt

Beliebige instanziierte OfCollection-Klasse

`Index`

Verwendung: erforderlich

Datentyp: lang

Die Nummer des Diagramm- oder Definitionselements im Objektgruppenzähler, das entfernt werden soll.

Beispiel

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

SetProperty

Zweck

Speichert die Eigenschaftenliste im OfCollection-Objekt. Es ist wichtig, dass das Definitionsobjekt ebenfalls gespeichert wird, da es die Eigenschaftenliste enthält, in der die Objektgruppe gespeichert ist.

Beispiel

```
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-Ereignisse

Einführung

In manchen Situationen müssen die Aktionen eines Benutzers sowie die Reaktion von Rational System Architect auf diese Aktionen gesteuert werden. Beispielsweise zum Verhindern bestimmter Vorgänge, zum Anzeigen von Warnungen, zum Erzwingen von Verhalten oder zum Ausführen zusätzlicher Vorgänge. In diesen Situationen ist es notwendig, die Aktion genau zur Zeit des Auftretens zu identifizieren und entsprechend zu reagieren.

Aktionen, die zu einem bestimmten Zeitpunkt auftreten, werden als “Ereignisse” bezeichnet. Diese können in Rational System Architect sofort beim Auftreten behandelt werden.

Rational System Architect erkennt eine Anzahl von Ereignissen, z. B. wenn ein Symbol in einem Diagramm platziert, ein Diagramm gespeichert oder eine Enzyklopädie geöffnet wird. In der folgenden Tabelle sind alle Ereignisse, die erkannt werden, zusammen mit Details zu Variablen, die übergeben werden, aufgeführt.

In diesem Kapitel enthaltene Themen	Seite
Anwendungsereignisse	15-2
Symbolereignisse	15-8

Anwendungsereignisse

AuditIDChanged

Zweck
Bei AuditID-Änderungen wird eine neue AuditId als NewAuditID übergeben. Die Prüf-ID kann im Menü **Datei, Prüf-ID…** geändert werden.

Syntax
```
AuditIDChanged(NewAuditID)
```

`NewAuditID`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Der Benutzername, in den die AuditID-Eigenschaft geändert wird.

Beispiel

DiagramClose

Zweck
Das Diagramm wird geschlossen.

Syntax
```
DiagramClose(hdgm)
```

`hdgm`

Verwendung: erforderlich

Datentyp: lang

Die Diagrammkennung wird als "hDgm" übergeben.

Beispiel

DiagramOpen

Zweck
Das Diagramm wird geöffnet.

Syntax
`DiagramOpen``(hdgm)`

`hDgm`

Verwendung: erforderlich

Datentyp: lang

Die Diagrammkennung wird als "hDgm" übergeben.

Beispiel

DiagramSave

Zweck
Das Diagramm wird gespeichert.

Beispiel

EncyClose

Zweck
Die Enzyklopädie wird geschlossen.

Beispiel

EncyOpen

Zweck
Die Enzyklopädie wird geöffnet.

Beispiel

MainMenuUpdate

Zweck
Das Hauptmenü wird aktualisiert. Dabei handelt es sich um das Menü "Datei" und um weitere zugehörige Menüoptionen.

Beispiel

MethodMenuUpdated

Zweck
Das Menü "Zeichnen" wird aktualisiert. Dies geschieht, wenn ein neuer Methodentyp eine erneute Menüzeichnung und das Hinzufügen neuer Menüoptionen erzwingt.

Beispiel

ReportsMenuUpdate

Zweck
Das Menü "Berichte" wird aktualisiert. Dies geschieht, wenn das Menü "Berichte" von Rational System Architect so geändert wird, dass es methodenspezifische Berichte oder Regelprüfungen anzeigt.

Beispiel

ToolsMenuUpdate

Zweck
Das Menü "Tools" wird aktualisiert. Dies geschieht, wenn das Menü "Tools" so aktualisiert wird, dass es Tooloptionen zu bestimmten verwendeten Methoden anzeigt.

Beispiel

ShowNode

Zweck
Ausgelöstes Ereignis, wenn der VBA-Filter des Browsers aktiv ist. Legen Sie den RetCode entsprechend fest, um dieses Element ein-/auszublenden.

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

`TabIndex`

Verwendung: erforderlich

Datentyp: Ganzzahl

Name der Registerkarte	Indexnummer der Registerkarte
Alle Methoden	1
Datenmodellierung	2
Geschäftsprozess	3
OO - traditionelles Produkt	4
Anwendung	5
Strukturiert	6
Organisation	7
Technologie	8
Position	9
Geschäftszielgebung	10
UML	11
XML	12

`TabName`

Verwendung: erforderlich

Datentyp: Zeichenfolge

Siehe Tabelle oben

`ddId`

Verwendung: erforderlich

Datentyp: lang

`Major`

Verwendung: erforderlich

Datentyp: lang

`Minor`

Verwendung: erforderlich

Datentyp: lang

`Name`

Verwendung: erforderlich

Datentyp: Zeichenfolge

`Memo`

Verwendung: erforderlich

Datentyp: Zeichenfolge

`LockFlags`

Verwendung: erforderlich

Datentyp: lang

`RetCode`

Verwendung: erforderlich

Datentyp: NodeStatus

Knotenstatus	Nummer
StatusDONTSHOW	0
StatusSHOW	1
StatusDONTCARE	2

Beispiel

ShutDown

Zweck
Für Rational System Architect wird ein Systemabschluss ausgeführt.

Beispiel

StartUp

Zweck
Für Rational System Architect wird ein Systemstart ausgeführt.

Beispiel

SymbolEvent

Zweck
Das Symbol wird in einem Diagramm platziert.

Syntax
`SymbolEvent``(hDgm, hSym, hSymOther, SymEvent, lData)`

`hDgm`

Verwendung: erforderlich

Datentyp: lang

Kennung des Diagramms

`hSym`

Verwendung: erforderlich

Datentyp: lang

Kennung des Symbols

`hSymOther`

Verwendung: erforderlich

Datentyp: lang

Kennung des Liniensymbols (mit dem eine Verbindung vom Knotensymbol hergestellt/getrennt wird).

`SymEvent`

Verwendung: erforderlich

Datentyp: SYMEVENTS

SYMEVENTS	Nummer	Beschreibung
ADDCONN	64	Das Knotensymbol wird mit einem Liniensymbol verbunden.
BREAKCONN	65	Die Verbindung des Knotensymbols zu einem Liniensymbol wird gelöscht.
SYM_DESELECTED	257	Das Symbol wird abgewählt.
SYM_SELECTED	256	Das Symbol wird ausgewählt.
TRANSFORMED	66	Das Symbol wird in ein anderes Symbol umgewandelt. Dieser Vorgang kann bei bestimmten Symbolen ausgeführt werden, indem Sie mit der rechten Maustaste auf das ausgewählte Symbol klicken und "Symbol umwandeln" auswählen.

`Ldata`

Verwendung: erforderlich

Datentyp: lang

Richtlinien für das programmgestützte Hinzufügen von Makroelementen zu Menüs

Ab V10.1 wurde bei Rational System Architect das bisherige Konzept mit einem festen Format von Menüs und Symbolleisten beinahe vollständig aufgegeben. Das heißt, dass die Benutzer jetzt Menüs dauerhaft anpassen können. Ausnahmen hierfür sind nur das Menü "Zeichnen" sowie die zugehörige Symbolleiste, bei denen eine Anpassung nur nach der Standardliste der Zeichentoolbefehle oder -schaltflächen möglich ist.

Das Ziel dieser Änderung ist das Zulassen angepasster Menü- und Symbolleisten-konfigurationen für bestimmte Benutzergruppen über SA Catalog Manager.

Bei V10.0 und früheren Versionen von Rational System Architect wurde in der Regel beim Schreiben des Makromenücodes berücksichtigt, dass Rational System Architect bestimmte Menüs löscht und deren erneute Zeichnung erzwingt. Dies bedeutete, dass die Benutzer, für die es erforderlich war, dass die Menüoptionen während einer Sitzung wieder ausgeblendet werden, die jeweiligen Elemente nicht programmgestützt aus einem Menü entfernen mussten. Ein typisches Beispiel hierfür war das Schließen eines Diagramms – das Menü "Tools" wurde gelöscht und neu erstellt. Auf diese Weise musste der Benutzer nur die Menüoption erneut einfügen, wenn es erforderlich war. Da Rational System Architect die Menüs jetzt nicht mehr löscht, bleiben die Menüoptionen eingeblendet, sofern der Makrocode nicht so geändert wird, dass sie entfernt (ausgeblendet) werden. Anweisungen zum Ändern solcher in Rational System Architect ab Version 10.0 integrierten Makros finden Sie im Konvertierungs-handbuch zu Rational System Architect, Conversion.pdf, auf der IBM Unterstützungswebsite.

Richtlinien zum Hinzufügen von Makros zu Menüs in Rational System Architect ab V10.1 finden Sie weiter unten in diesem Abschnitt.

Wie füge ich ein Makro zu einem Rational System Architect-Standardmenü hinzu?

Die SA2001.Application-Hauptmethode zum Unterstützen des Hinzufügens von Makro-elementen lautet wie folgt:

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

Diese Funktion fügt den MacroName (der sich aus <Projekt>.<Modul>.<Subroutine>" zusam-mensetzt) mithilfe der Beschriftung MacroItemCaption zum SA-Benutzermenü mit dem Titel InMenuTitleCaption hinzu, vor der Menüoption mit dem Titel BeforeMenuItemCaption. Wenn kein Before-Element angegeben ist, wird das Element zum Ende des Menüs hinzugefügt. Wenn das Before-Element #TOP# lautet, dann wird das Element oben ins Menü eingefügt.

Die Funktion gibt bei erfolgreicher Ausführung "0" zurück.

Das bedeutet Folgendes:

```
Set App = New SA2001.Application

    x = App.InsertMacroItemInMenu("MyProject.MyModule.MySub", "&Test Item",
"&Tools","Ma&cros")
```

Hierdurch wird das Element "Test Item" (Testelement) in das Menü "Tools" vor dem Kontextelement "Macros" eingefügt. Wenn Sie auf das Element klicken, wird die Subroutine "MySub" im Modul "MyModule" im Projekt "MyProject" ausgeführt.

Damit dies erfolgreich ist, MUSS die angegebene Subroutine bereits vorhanden sein. Außerdem MÜSSEN die Beschriftungen genau so angegeben werden wie im Menü.

Beispiel: Tools wird als Zeichenfolge &Tools

Ein Et-Zeichen steht für ein Unterstreichungszeichen nach dem darauf folgenden Buchstaben. Bei der Eingabe muss die Groß-/Kleinschreibung beachtet werden.

Auf den Subroutinennamen sollten KEINE "()"-Zeichen folgen.

Das Hinzufügen von BMPs zu Menüoptionen wird von der Methode AssignBMPtoMacroItem ausgeführt.

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

Diese Funktion gibt bei erfolgreicher Ausführung eine "0" zurück.

Dieser Schritt sollte ausgeführt werden, bevor Sie das Makroelement zum Menü hinzufügen - es muss jedoch nur einmal getan werden.

Beispiel:

```
Set App = New SA2001.Application

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

x = App.InsertMacroItemInMenu("MyProject.MyModule.MySub", "&Test Item",
"&Tools","Ma&cros")
```

Hierdurch wird "piccy.bmp" zum Makro "MySub" hinzugefügt. Fügen Sie es anschließend wie im vorherigen Beispiel zu den Menüs hinzu. Beachten Sie, dass die Option zum Anpassen von Menüs (klicken Sie hierfür mit der rechten Maustaste auf das Menü) nun "Piccy.BMP" sowie die Beschriftung "&Test Item" enthält, wenn Sie die Menüoption an einer anderen Position in Ihren Menüs ablegen möchten.

Ereignisse verwenden

Die folgenden SA2001.Application-Ereignisse können den Makrocode bei der Entscheidung unterstützen, wann Änderungen zum Ein-/Ausblenden des Menüs vorgenommen werden sollen.

- MainMenuUpdate: Wenn das gesamte Menü betroffen ist
- MethodMenuUpdate: Wenn Verzeichnistyp-Menüoptionen aktualisiert werden
- ToolsMenuUpdate: Wenn Tooltyp-Menüoptionen aktualisiert werden
- ReportsMenuUpdate: Wenn Berichtstyp-Menüoptionen aktualisiert werden
- App_ShutDown: Wenn für SA ein Systemabschluss durchgeführt wird

Ereignisse müssen NICHT MEHR über Benutzerelemente in den Menüs verfügen. Wenn Sie jedoch Elemente aus einem Menü entfernen möchten, stellen die folgenden Codebereiche die Gliederung für eine solche Ereignisverarbeitung zur Verfügung. (Hinweis: Das Projekt lauet "MyProject".)

MODULE - AutoExec

```
' Hauptmodul, das die VBA-basierte Ereignisverarbeitung verwaltet
Dim EventHandler As EventCls
' Hauptsubroutine, die die Ereignisverarbeitung startet
Sub Main()
  Set EventHandler = New EventCls     ' Erstellen Sie eine Ereignisverarbeitung
  Set EventHandler.App = Application  ' Stellen Sie eine Verbindung von der
Ereignisverarbeitung zu SA her
  ' Führen Sie zusätzliche Anwendungsstarts an dieser Stelle aus
  ' Hier sollten Sie Ihre BMPs hinzufügen und Ihre Kontextmenüs erstellen
  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
  ' Hinweis: Wenn ein Dialogfenster erstellt ist, bleibt es so lange vorhanden, bis es von
"RemovePopupMenu" entfernt wird. Es kann in Menüs beliebig hinzugefügt/entfernt werden.

  ' Erstellen Sie das Kontextmenü mit der zugehörigen Bitmap
  x = Application.CreatePopUpMenu("Sample Macros","SAWORD.BMP")
  ' Fügen Sie das Element zum Dialogfenster hinzu
  x = Application.InsertMacroItemInMenu("MyProject.MyModule.PRINTDIAGRAMS", "&Print a
Diagram", "Sample Macros")
End Sub
```

CLASS - EventCls

```
Public WithEvents App As Application  ' Die Anwendung ruft Ereignisse auf

Private Sub Class_Initialize()
  ' Es muss keine Aktion ausgeführt werden
End Sub

Private Sub Class_Terminate()
  ' Es muss keine Aktion ausgeführt werden
```

```
End Sub

Private Sub App_ReportsMenuUpdate()
  ' Das Menü "Reports" (Berichte) wurde erneut gezeichnet
  Dim x As Long
  x = App.SetSeparatorBefore("&Report Generator...", "&Reports", True)

  ' Fügen Sie eine Menüoption mit dem Namen 'Print all Diagrams' (Alle Diagramme drucken),
das PrintDiagrams2 aufruft, in das Menü "Reports" (Berichte) vor dem Element "Report
Generator..." (Berichtsgenerator) ein
  x = App.InsertMacroItemInMenu("MyProject.MyModule.PRINTDIAGRAMS2", "Print all
&Diagrams", "&Reports", "&Report Generator...")

  ' Fügen Sie Ihr Kontextmenü "Sample Macros" (Beispielmakros) in das Menü "Reports"
(Berichte) vor dem Element "Print all Diagrams" (Alle Diagramme drucken) ein.
  ' Hinweis: Sie sollten keine Dialogfenster erstellen und löschen, wenn Sie das nicht
unbedingt müssen - tun Sie dies einmal bei der Initialisierung.
  x = App.InsertPopupMenuItemInMenu("Sample Macros", "&Reports", "Print all &Diagrams")
End Sub
```

MODULE - MyModule

```
Public Sub PrintDiagrams()
        ' Auszuführender Benutzercode
        MsgBox "Print Diagrams"
End Sub
Public Sub PrintDiagrams2()
        ' Auszuführender Benutzercode
        MsgBox "Print Diagrams2"
End Sub
```

Wie füge ich ein Dialogfenster hinzu?

Verwenden Sie die CreatePopUpMenu-Methode mit dem Namen, der für das Dialogfenster verwendet werden soll. Verwenden Sie ggf. auch eine BMP. Das Dialogfenster wird zur Stammobjektgruppe von Dialogfenstern in SA hinzugefügt.

Wenn Sie SA verlassen, muss Ihre Ereignisverarbeitungsklasse hinzugefügte Dialogfenster mithilfe von RemovePopUpMenu nicht mehr entfernen.

Wie füge ich Makroelemente zu einem Dialogfenster hinzu?

Fügen Sie sie so hinzu wie beim Hinzufügen zu SA-Menüs.

Wie füge ich ein Dialogfenster zu einem SA-Menü hinzu?

Sie können ein Dialogfenster zu einem Rational System Architect-Menü auf dieselbe Weise hinzufügen wie ein Makroelement zu einem Menü; verwenden Sie hierbei jedoch InsertPopupMenuItemInMenu und geben Sie keinen Makronamen an.

Beispiel:

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

Wie füge ich ein Trennzeichen zu einer Menüoption hinzu?

Verwenden Sie SetSeparatorBefore, geben Sie den Menü- und den Elementnamen an und legen Sie "True" (Wahr) für "Trennzeichen" und "False" (Falsch) für "keine Trennzeichen" fest.

Wie entferne ich ein Element aus einem Menü?

Sie können Dialogfenster oder Menüoptionen mithilfe derselben Methode entfernen: Verwenden Sie RemoveItemFromMenu und geben Sie dabei den Namen des Menüs und der Menüoption an.

Hinweis: Sie können nur Elemente entfernen, die Sie hinzugefügt haben. Beispiel: Rational System Architect-Menüoptionen können nicht entfernt werden. Die Optionen werden nicht entfernt, sie werden nur ausgeblendet. Das ermöglicht es dem Benutzer, die Optionen so anzupassen, dass sie an einer anderen Position im Menüsystem angezeigt werden. Beachten Sie, dass Sie “” für den Menünamen angeben können, da der Menüname nun ignoriert wird. Die Option wird überall im Menüsystem ausgeblendet.

Da alle Dialogfenster und Tools, die zum Menüsystem hinzugefügt werden, auch nach einem Neustart von SA vorhanden bleiben, ist es möglicherweise notwendig, ein Kontextmenü zu entfernen, das nicht mehr benötigt wird.
Die Methode App.RemovePopUpMenu(<PopupName>) entfernt das Dialogfenster vollständig aus dem Menüsystem.

Wenn diese Methode in einem Makro, beispielsweise beim SA-Systemabschluss, verwendet wird, werden das Dialogfenster und die gesamte Anpassung immer dann entfernt, wenn für SA ein Systemabschluss durchgeführt wird. Das führt dazu, dass die Anpassung dieses Dialogfensters zwischen verschiedenen SA-Sitzungen nicht beibehalten wird.

16

Rational System Architect-Beziehungen

Einführung

In diesem Kapitel finden Sie Beschreibungen aller Beziehungen zwischen Objekten im Rational System Architect-Repository. Im SA-Objektbrowser finden Sie den Aufzählungstyp RELATETYPE. Er enthält alle möglichen Beziehungen, die Sie in Ihrem VBA-Makro verwenden können.

Die Diagramm-, Symbol- und Definitionsklassen verfügen jeweils über die Methode **GetRelatedObjects**. Um diese Methode auszuführen, muss der Benutzer angeben, welche Beziehung (von den in der Tabelle aufgeführten) vorhanden ist. Außerdem enthält die Enzyklopädieklasse die Methode **GetRelationMetric**. Der Benutzer muss einen der Beziehungstypen als erforderlichen Parameter angeben, um die Beziehungsmessdaten auszuführen.

In diesem Kapitel enthaltene Themen	Seite
Beziehungstypen	16-2

Beziehungstypen

RELATETYPE	Beschreibung	Nummer
RELNULL	Null	0
RELNULL2	Null	1
RELDIAGRAMCON	Diagramm enthält Symbole	2
RELCONDIAGRAM	Symbol in Diagrammen enthalten	3
RELSHOWTO	Symbol wird in einem (untergeordneten) Diagramm erweitert	4
RELSHOWFROM	Ein Diagramm wird in einem (übergeordneten) Symbol erweitert	5
RELCONNSTART	Ein Knotensymbol wird mit dem Beginn eines Liniensymbols verbunden	6
RELSTARTAT	Der Beginn einer Linie wird mit einem Knotensymbol verbunden	7
RELCONNEND	Ein Knotensymbol wird mit dem Ende eines Liniensymbols verbunden	8
RELENDAT	Das Ende einer Linie wird mit einem Knotensymbol verbunden	9
RELFLAGSENDS	Ein Modul wird mit einem Flagsymbol verbunden und sendet darüber Daten	10
RELFLAGSTR	Ein Flagsymbol wird mit einem Modul verbunden und empfängt Daten von diesem	11
RELFLAGRECVS	Ein Modul wird einem Flagsymbol verbunden und empfängt Daten von diesem	12
RELFLAGEND	Ein Flagsymbol wird mit einem Modul verbunden und stellt diesem Daten Verfügung	13

RELATETYPE	Beschreibung	Nummer
RELDFEELEMENT	Ein Ausdruck verwendet Daten (Elemente oder Strukturen)	14
RELELEMENTDFE	Daten (Elemente oder Strukturen) werden von einem Ausdruck verwendet	15
RELEXPLAINEDBY	Ein Symbol wird durch einen Kommentar erläutert	16
RELEXPLAINS	Ein Kommentar erläutert ein Symbol	17
RELPARTFULFILLS	Ein Symbol gilt für eine Anforderung, einen Testplan usw.	18
RELPARTFULFILLEDBY	Eine Anforderung, ein Testplan usw. gilt für ein Symbol	19
RELCOMPFULFILLS	Ein Symbol wird durch eine Definition definiert	20
RELDEFINEDBY	Ein Symbol wird durch eine Definition definiert	20
RELCOMPFULFILLEDBY	Eine Definition definiert ein Symbol	21
RELDEFINES	Eine Definition definiert ein Symbol	21
RELISQUALIFIEDBY	Ein Linien- oder ein Knotensymbol wird von einem Flagsymbol "qualifiziert"	22
RELQUALIFIES	Ein Flagsymbol "qualifiziert" ein Liniensymbol (oder ein Knotensymbol)	23
RELISA	Eine Definition "ist eine Instanz" einer Definition	24
RELINSTBY	Eine Definition wird von einer Definition "instanziiert"	25
RELIDENTIFIES	Eine Definition "identifiziert" eine andere Definition	26

RELATETYPE	Beschreibung	Nummer
RELKEYEDBY	Eine Definition wird von einer Definition identifiziert	27
RELEMBEDS	Ein Knotensymbol bettet ein Symbol vollständig ein	28
RELISEMBEDDEDBY	Ein Knotensymbol wird von einem Knotensymbol vollständig eingebettet	29
RELISPARENTIN	Eine Definition ist ein übergeordnetes Element einer Definition in einer Beziehung zwischen einem über- und einem untergeordneten Element	30
RELHASPARENTOF	Eine Definition verfügt über eine übergeordnete Definition in einer Beziehung zwischen einem über- und einem untergeordneten Element	31
RELISCHILDIN	Eine Definition ist ein untergeordnetes Element einer Definition in einer Beziehung zwischen einem über- und einem untergeordneten Element	32
RELHASCHILDOF	Eine Definition verfügt über eine untergeordnete Definition in einer Beziehung zwischen einem über- und einem untergeordneten Element	33
RELCOMPRISES	Definition umfasst eine Definition	34
RELISPARTOF	Eine Definition ist Teil einer Definition	35
RELISCATZNOF	Kategorisierungsdefinition kategorisiert eine generische Entitätendefinition	36
RELHASCATZN	Generische Entitätendefinition verfügt über eine Kategorisierung einer Kategorisierungsdefinition	37

RELATETYPE	Beschreibung	Nummer
RELISCATGRYIN	Entitätendefinition ist eine Kategorie in einer Kategorisierung	38
RELHASCATGRY0F	Kategorisierung verfügt über eine Kategorie einer Entitätendefinition	39
RELISCHILDOF	Symbol ist das untergeordnete Element eines anderen Symbols (nur in einem hierarchischen Diagramm)	40
RELISPARENTOF	Symbol ist das übergeordnete Element eines anderen Symbols (nur in einem hierarchischen Diagramm)	41
RELISFIRSTCHILDOF	Das Symbol ist das erste untergeordnete Element eines anderen Symbols (z. B. das Symbol ganz links) (nur in einem hierarchischen Diagramm)	42
RELHASFIRSTCHILD	Das Symbol verfügt über ein anderes Symbol als erstes untergeordnetes Element (nur in einem hierarchischen Diagramm)	43
RELISNEXTSIBLING	Das Symbol ist das nächste gleich-geordnete Element eines anderen gleichgeordneten Elements (nur in einem hierarchischen Diagramm)	44
RELISPRIORSIBLING	Das Symbol ist das vorherige gleichgeord-nete Element eines anderen Symbols (nur in einem hierarchischen Diagramm)	45
RELISINDEXOF	Datenmodell - Zugriffspfad oder Index einer Entität oder Tabelle	46
RELISINDEXEDBY	Eine Definition wird von einer anderen Definition indexiert	47

RELATETYPE	Beschreibung	Nummer
RELORIGINATESFROM	Eine Definition wird von einer anderen Definition eingeleitet (z. B. einem Grafik-anzeigediagramm)	48
RELISORIGINOF	Eine Definition ist der Ursprung einer Definition	49
RELISBASEDON	Eine Definition basiert auf einer Definition (in der Regel auf einem Datenelement)	50
RELISBASISFOR	Eine Definition ist die Basis für eine abgeleitete Definition	51
RELISLINKEDTO	Ein Symbol wird mit einem anderen Symbol verlinkt	52
RELISLINKEDWITH	Ein Symbol wird mit einem anderen Symbol verlinkt	53
RELUSER AND RELUSERCOMPLEMENT	Benutzer und Benutzerkomplement	54 bis 83
RELPOPKIN AND RELPOPKINCOMPLEMENT	IBM und IBM Komplement	84 bis 111
RELREPRESENTS	Explorer-Diagrammsymbol steht für ein Objekt	112
RELISPRESENTEDBY	Objekt wird von einem Explorer-Diagrammsymbol dargestellt	113
RELPOPKIN AND RELPOPKINCOMPLEMENT	IBM und IBM Komplement	114 bis 125
RELUSER AND RELUSERCOMPLEMENT	Benutzer und Benutzerkomplement	126 bis 135
RELLINKS TO	Objekt wird mit einem Objekt in DOORS verlinkt	136
RELISLINKEDFROM	Objekt ist von einem Objekt in DOORS verlinkt	137

RELATETYPE	Beschreibung	Nummer
RELISTOBESENTTO	Objekt soll an DOORS-Modul gesendet werden	138
RELISTORECEIVE	DOORS-Modul soll ein Objekt empfangen	139
RELHASBEENSENTTO	Objekt wurde an DOORS-Modul gesendet	140
RELHASRECEIVED	DOORS-Modul hat Objekt empfangen	141

17

Rational System Architect-Feldtypen

Einführung

Dieses Kapitel umfasst alle für den Benutzer verfügbaren Feldtypen, die im Rational System Architect-Objektbrowser unter dem Aufzählungstyp FLDTYPE aufgeführt sind. Diese Konstanten bestimmen das Format, in dem Daten zurückgegeben werden. In den meisten Fällen wurde bei Rational System Architect der Feldtyp intern festgelegt. Dieser Aufzählungstyp wird als erforderlicher Parameter für die GetRelationMetric-Methode der Enzyklopädieklasse verwendet. Er ist außerdem ein optionaler Parameter den GetMetric-Methoden der Diagramm-, Symbol- und Definitionsklassen.

Die Tabelle unten enthält eine vollständige Liste aller Feldtypen und der zugehörigen Beschreibungen.

In diesem Kapitel enthaltene Themen	Seite
Feldtypen	17-2

Feldtypen

FLDTYPE	Nummer	Beschreibung
FLDTYPAUTO	65	Hierbei handelt es sich um das Standardfeld, das intern von Rational System Architect ausgewählt wird.
FLDTYPCHARACTER	67	Enthält eine Zeichenfolge mit einer Länge von bis zu 256 Zeichen
FLDTYPDATE	68	Hierbei handelt es sich um ein Datumsfeld (z. B TT/MM/JJJJ)
FLDTYPLOGICAL	76	Boolesches Feld
FLDTYPMEMO	77	Felder für Kurzinfo speichern große Textblöcke mit bis zu 4.095 Zeichen
FLDTYPNUMERIC	78	Gibt an, dass das Feld Werte enthalten muss, die Nummern sind.
FLDTYPTIME	84	Hierbei handelt es sich um ein Zeitfeld (z. B. Stunden:Minuten:Sekunden)

18

Rational System Architect-Fehler

Einführung

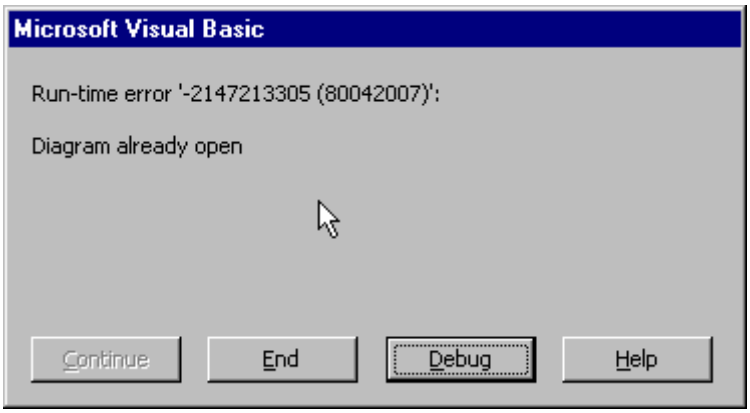
Dieses Kapitel umfasst die Fehler, die von Rational System Architect generiert werden und die während der Codeausführung abgefangen werden können. Das Kapitel befasst sich nicht mit Fehlern im geschriebenen Code (ein Beispiel hierfür ist der Visual Basic-Fehler 91 “Object Variable or With block variable not set” [Objektvariable oder Variable mit Block nicht festgelegt], bei dem eine Instanz der Variable nicht erstellt wurde).

In diesem Kapitel enthaltene Themen	Seite
SA-Fehler	18-3

Fehler behandeln

Die Eigenschaften und Methoden im Rational System Architect-Objektmodell führen den zugehörigen Code auf konsistente Weise aus, aber leider sind die Basisdaten nicht immer im erwarteten Zustand und Fehler können während der normalen Ausführung eines VBA-Makros auftreten. Während der normalen Ausführung können mehrere Probleme auftreten; diese müssen abgefangen und separat behandelt werden, sonst wird das Makro abrupt beendet, ohne die Codeausführung zu beenden.

Wenn während der Codeausführung ein Fehler auftritt, wird das folgende Dialogfenster angezeigt. Die Methode **Diagram.Show** wurde zum Öffnen der aktuellen Diagrammgrafik ausgeführt. Das Diagramm jedoch wurde bereits an einem anderen Punkt im Code geöffnet, deshalb trat ein Fehler auf.



Der Benutzer kann auf "End" (Beenden), um das Programm am derzeitigen Punkt zu beenden, oder auf "Debug" (Fehler beheben) klicken, wodurch die Schnittstelle des VBA-Editors angezeigt und die Position markiert wird, an der der Fehler im Code auftrat. Der Benutzer erhält weitere Erläuterungen, wenn er auf die Schaltfläche "Help" (Hilfe) klickt.

Rational System Architect-Fehler

Rational System Architect weist eine Reihe von Fehlern auf, die identifiziert werden können. VBA kann diese Fehler verfolgen, wenn sie in einem Makro auftreten, und eine Nachricht an die VBA-Fehlerbehandlungsroutine senden, die dann den Fehler dem Benutzer anzeigt. Die verschiedenen Fehler verursachen unterschiedliche Probleme. Wenn eine Methode ausgeführt wird, gibt sie einen Wert an VBA zurück, der den Status des Aufrufs angibt; wenn eine Ausnahme auftritt, dokumentiert VBA den Fehler. Der Aufzählungstyp SA2001Errors enthält eine vollständige Liste aller Rational System Architect-Fehler.

Fehler	Nummer	Beschreibung
SAERR_BADDDID	8195	
SAERR_DIAGRAMNOTOPEN	8198	Diagramm ist nicht geöffnet. Entweder wurde das Diagramm nicht geöffnet, oder die **.Hide**-Methode wurde zuvor aufgerufen.
SAERR_DIAGRAMNOTSAME	8201	
SAERR_DIAGRAMOPEN	8199	Diagramm ist bereits geöffnet. Entweder wurde das Diagramm zuvor geöffnet oder die **.Show**-Methode wurde zuvor aufgerufen.
SAERR_ENUMVARIANTERROR	8193	
SAERR_INVALIDCLASS	8202	
SAERR_INVALIDOBJECT	8194	
SAERR_INVALIDPROPERTY	8204	
SAERR_INVALIDTYPE	8207	
SAERR_NOTIMPLEMENTED	8196	
SAERR_OBJECTDOESNOTEXIST	8197	Entweder wurde das Objekt nicht instanziiert oder es wurde zuvor gelöscht.

Fehler	Nummer	Beschreibung
SAERR_OBJECTISLOCKED	8205	Entweder wurde zuvor die OpenLock-Methode aufgerufen oder das Objekt wurde zuvor ausgecheckt oder von einem anderen Benutzer blockiert.
SAERR_OBJECTNOTFOUND	1025	
SAERR_OPENEDASREADONLY	8206	Das Objekt wurde schreibgeschützt geöffnet.
SAERR_REQUIREDPROPERTYABSENT	8192	Die Verweiseigenschaft ist entweder leer oder nicht vorhanden.
SAERR_SA20001_IMF_ERROR	4096	Ein SAIMF-Fehler ist aufgetreten.
SAERR_SANOTRUNNING	1024	Rational System Architect wurde entweder nicht gestartet oder wurde zuvor herunter-gefahren.
SAERR_SYMBOLHASNODIAGRAM	8200	Auf das Symbol wird von keinem Diagramm verwiesen.
SAERR_TOOMANYOBJECTS	8203	

19

IBM Unterstützung

Einführung

Es sind einige Informationsquellen und Tools zur Selbsthilfe verfügbar, um Sie bei der Fehlerbehebung zu unterstützen. Sie können wie folgt vorgehen, wenn ein Fehler an Ihrem Produkt vorliegt:

In den Releaseinformationen zu Ihrem Produkt erhalten Sie Informationen zu bekannten Fehlern, Fehlerlösungsstrategien sowie Fehlerbehebungsinformationen.

Überprüfen Sie, ob ein Download oder ein Fix zur Behebung Ihres Fehlers verfügbar ist.

Durchsuchen Sie die verfügbaren Wissensdatenbanken, um festzustellen, ob die Lösung zu Ihrem Problem bereits dokumentiert ist.

Wenn Sie weiterhin Hilfe benötigen, wenden Sie sich an IBM® Software Support und melden Sie das Problem.

Themen in diesem Kapitel	Seite
IBM Rational-Softwareunterstützung kontaktieren	19-2

IBM Rational-Software-unterstützung kontaktieren

Wenn Sie mithilfe der Informationsquellen zur Selbsthilfe einen Fehler nicht beheben können, wenden Sie sich an die IBM® Rational®-Softwareunterstützung.

Hinweis: Wenn Sie Kunde von Telelogic sind, finden Sie eine einzelne Referenzsite für alle Unterstützungsinformationsquellen unter http://www.ibm.com/software/rational/support/telelogic/

Voraussetzungen

Um einen Fehler an die IBM Rational-Softwareunterstützung senden zu können, müssen Sie über einen aktiven Softwarewartungsvertrag für Passport Advantage® verfügen. Passport Advantage ist das umfassende IBM Angebot zur Softwarelizenzierung und -wartung (Produktupgrades und technische Unterstützung). Sie können sich online für Passport Advantage unter der folgenden Adresse registrieren: http://www.ibm.com/software/lotus/passportadvantage/howtoenroll.html

- Weitere Informationen zu Passport Advantage erhalten Sie in den Passport Advantage FAQs unter http://www.ibm.com/software/lotus/passportadvantage/brochures_faqs_quickguides.html.
- Weitere Unterstützung erhalten Sie bei Ihrem IBM Ansprechpartner.

Gehen Sie wie folgt vor, um einen Fehler online (von der IBM Website) an die IBM Rational-Softwareunterstützung zu senden:

- Melden Sie sich als Benutzer auf der Unterstützungswebsite für IBM Rational Software an. Ausführliche Informationen zum Registrieren finden Sie unter http://www.ibm.com/software/support/.
- Sie müssen im Serviceanforderungstool als autorisierter Anrufer aufgelistet sein.

Weitere Informationen

Rational-Softwareproduktneuigkeiten, -ereignisse und weitere Informationen erhalten Sie auf der Website "IBM Rational Software" unter http://www.ibm.com/software/rational/.

Fehler senden

Gehen Sie wie folgt vor, um einen Fehler an die IBM Rational-Softwareunterstützung zu senden:

1. Bestimmen Sie die Auswirkungen des Fehlers auf die Geschäftsabläufe. Wenn Sie einen Fehlerbericht an IBM senden, werden Sie darum gebeten, einen Schweregrad anzugeben. Deshalb müssen Sie die Auswirkungen des Fehlers auf die Geschäftsabläufe verstehen und bewerten können.

Verwenden Sie zum Bestimmen des Schweregrads die folgende Tabelle.

Schweregrad	Beschreibung
1	Der Fehler ist geschäftskritisch: Sie können das Programm nicht verwenden, was sich auf die Abläufe kritisch auswirkt. Dieser Zustand erfordert eine sofortige Lösung.
2	Der Fehler hat signifikante Auswirkungen auf die Geschäftsabläufe: Das Programm kann verwendet werden, ist jedoch stark eingeschränkt.
3	Der Fehler hat zu einem bestimmten Teil Auswirkungen auf die Geschäftsabläufe: Das Programm kann verwendet werden, aber weniger signifikante (keine vorgangs-kritischen) Funktionen sind nicht verfügbar.
4	Der Fehler hat minimale Auswirkungen auf die Geschäftsabläufe: Der Fehler hat geringe Auswirkungen auf Vorgänge oder es wurde eine vernünftige Umgehung des Fehlers implementiert.

2. Beschreiben Sie den Fehler und erfassen Sie Hintergrundinformationen. Seien Sie so genau wie möglich, wenn Sie Fehlerbeschreibungen an IBM senden. Fügen Sie alle relevanten Hintergrundinformationen hinzu, damit die Fachleute bei der IBM Rational-Softwareunterstützung Sie effizient bei der Fehlerbehebung unterstützen können. Halten Sie aus Zeitgründen die Antworten auf die folgenden Fragen bereit:

    - Welche Softwareversionen haben Sie ausgeführt, als der Fehler auftrat?
    - Um den entsprechenden Produktnamen und die zugehörige Version zu bestimmen, verwenden Sie die jeweils geeignete Option:
        - Starten Sie den IBM Installation Manager und klicken Sie auf **File** (Datei) > **View Installed Packages** (Installierte Pakete anzeigen). Erweitern Sie eine Paketgruppe und wählen Sie ein Paket aus, um den Paketnamen und die Versionsnummer anzuzeigen.
        - Starten Sie Ihr Produkt und klicken Sie auf **Help** (Hilfe) > **About** (Produktinfo), um den Angebotsnamen und die Versionsnummer anzuzeigen.
    - Was ist Ihr Betriebssystem und was ist die zugehörige Versionsnummer (inklusive Service-Packs oder Patches)?
    - Verfügen Sie über Protokolle, Traces und Nachrichten zu den Fehlersymptomen?
    - Können Sie den Fehler erneut verursachen? Wenn ja, welche Schritte führen Sie aus, um den Fehler erneut zu verursachen?
    - Haben Sie am System irgendwelche Änderungen vorgenommen? Haben Sie beispielsweise Änderungen an der Hardware, am Betriebssystem, an der Netzsoftware oder an anderen Systemkomponenten vorgenommen?

3. Verwenden Sie derzeit eine Lösungsstrategie für den Fehler? Wenn ja, beschreiben Sie diese Strategie, wenn Sie den Fehlerbericht erstellen.

4. Senden Sie den Fehlerbericht auf eine der folgenden Weisen an die IBM Rational-Softwareunterstützung:

    - Online: Rufen Sie die Unterstützungswebsite für IBM Rational Software unter https://www.ibm.com/software/rational/support/ auf. Klicken Sie im Tasknavigator für die Rational-Unterstützung auf **Open Service Request** (Serviceanforderung öffnen). Wählen Sie das elektronische Fehlerberichtstool aus und öffnen Sie ein PMR (Problem Management Record), um den Fehler zu beschreiben.

    - Weitere Informationen zum Öffnen einer Serviceanforderung finden Sie unter http://www.ibm.com/software/support/help.html.

    - Sie können eine Serviceanforderung auch online öffnen, indem Sie den IBM Support Assistant verwenden. Weitere Informationen finden Sie unter http://www.ibm.com/software/support/isa/faq.html.
    - Über das Telefon: Die entsprechende Telefonnummer für Ihr Land oder Ihre Region finden Sie im weltweiten Verzeichnis von IBM Ansprechpartnern unter http://www.ibm.com/planetwide/. Klicken Sie auf den Namen Ihres Landes oder Ihrer Region.

    - Über Ihren IBM Ansprechpartner: Wenn Sie auf die IBM Rational-Softwareunterstützung online oder telefonisch nicht zugreifen können, wenden Sie sich an Ihren IBM Ansprechpartner. Ihr IBM Ansprechpartner kann dann ggf. eine Serviceanforderung für Sie öffnen. Die vollständigen Informationen zu Ansprechpartnern für Ihr Land finden Sie unter http://www.ibm.com/planetwide/.

20

Anhang:

Einführung

Dieses Kapitel enthält Informationen zur gesetzlichen Verwendung und zu Marken von IBM® Rational® System Architect®.

Themen in diesem Kapitel	Seite
Bemerkungen	20-2
Marken	20-5

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern oder Regionen nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim zuständigen IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Services von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Services können auch andere, ihnen äquivalente Produkte, Programme oder Services verwendet werden, solange diese keine gewerblichen oder anderen Schutzrechte von IBM verletzen. Die Verantwortung für den Betrieb von Produkten, Programmen und Services anderer Anbieter liegt beim Kunden.

Für die in diesem Handbuch beschriebenen Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an folgende Adresse zu richten (Anfragen an diese Adresse müssen auf Englisch formuliert werden):

IBM Director of Licensing
IBM Europe, Middle East & Africa
Tour Descartes 2, avenue Gambetta
92066 Paris La Defense

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die hier enthaltenen Informationen werden in regelmäßigen Zeitabständen aktualisiert und als Neuausgabe veröffentlicht. IBM kann ohne weitere Mitteilung jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter werden lediglich als Service für den Kunden bereitgestellt und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Werden an IBM Informationen eingesandt, können diese beliebig verwendet werden, ohne dass eine Verpflichtung gegenüber dem Einsender entsteht.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängig voneinander erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

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

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt auf der Basis der IBM Rahmenvereinbarung bzw. der Allgemeinen Geschäftsbedingungen von IBM, der IBM Internationalen Nutzungsbedingungen für Programmpakete, der IBM Lizenzvereinbarung für Maschinencode oder einer äquivalenten Vereinbarung.

Alle in diesem Dokument enthaltenen Leistungsdaten stammen aus einer kontrollierten Umgebung. Die Ergebnisse, die in anderen Betriebsumgebungen erzielt werden, können daher erheblich von den hier erzielten Ergebnissen abweichen. Einige Daten stammen möglicherweise von Systemen, deren Entwicklung noch nicht abgeschlossen ist. Eine Gewährleistung, dass diese Daten auch in allgemein verfügbaren Systemen erzielt werden, kann nicht gegeben werden. Darüber hinaus wurden einige Daten unter Umständen durch Extrapolation berechnet. Die tatsächlichen Ergebnisse können davon abweichen. Benutzer dieses Dokuments sollten die entsprechenden Daten in ihrer spezifischen Umgebung prüfen.

Alle Informationen zu Produkten anderer Anbieter stammen von den Anbietern der aufgeführten Produkte, deren veröffentlichen Ankündigungen oder anderen allgemein verfügbaren Quellen. IBM hat diese Produkte nicht getestet und kann daher keine Aussagen zu

Leistung, Kompatibilität oder anderen Merkmalen machen. Fragen zu den Leistungsmerkmalen von Produkten anderer Anbieter sind an den jeweiligen Anbieter zu richten.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogramms illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

Copyrightlizenz

Diese Veröffentlichung enthält Musteranwendungsprogramme, die in Quellensprache geschrieben sind und Programmiertechniken in verschiedenen Betriebsumgebungen veranschaulichen. Sie dürfen diese Musterprogramme kostenlos kopieren, ändern und verteilen, wenn dies zu dem Zweck geschieht, Anwendungsprogramme zu entwickeln, zu verwenden, zu vermarkten oder zu verteilen, die mit der Anwendungsprogrammierschnittstelle für die Betriebsumgebung konform sind, für die diese Musterprogramme geschrieben werden. Diese Beispiele wurden nicht unter allen denkbaren Bedingungen getestet. Daher kann IBM die Zuverlässigkeit, Wartungsfreundlichkeit oder Funktion dieser Programme weder zusagen noch gewährleisten.

Kopien oder Teile der Musterprogramme bzw. daraus abgeleiteter Code müssen folgenden Copyrightvermerk beinhalten:

© (Name des Unternehmens) (Jahr). Teile des vorliegenden Codes wurden aus Musterprogrammen der IBM Corp. abgeleitet.

Wird dieses Buch als Softcopy (Book) angezeigt, erscheinen keine Fotografien oder Farbabbildungen.

Marken

IBM, das IBM Logo und ibm.com sind Marken oder eingetragene Marken der International Business Machines Corp., die in vielen Rechtsordnungen weltweit registriert sind. Andere Produkt- und Servicenamen können Marken von IBM oder anderen Unternehmen sein. Eine aktuelle Liste der IBM Marken steht im World Wide Web auf der Seite "Copyright and trademark information" unter www.ibm.com/legal/copytrade.html

Microsoft und Windows sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.

Weitere Unternehmens-, Produkt- oder Servicenamen können Marken anderer Hersteller sein.