IBM Rational

System Architect

Guide d'extensibilité de VBA

Version 11.3.1

Avant d'utiliser le présent document, lisez la section "Remarques" de l'annexe, à la page 20-1.

LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. Les informations qui y sont fournies sont susceptibles d'être modifiées avant que les produits décrits ne deviennent eux-mêmes disponibles. En outre, il peut contenir des informations ou des références concernant certains produits, logiciels ou services non annoncés dans ce pays. Cela ne signifie cependant pas qu'ils y seront annoncés.

Pour plus de détails, pour toute demande d'ordre technique, ou pour obtenir des exemplaires de documents IBM, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial.

Vous pouvez également consulter les serveurs Internet suivants :

http://www.fr.ibm.com (serveur IBM en France)
http://www.can.ibm.com (serveur IBM au Canada)
http://www.ibm.com (serveur IBM aux Etats-Unis)

Compagnie IBM France
Direction Qualité
17, avenue de l'Europe
92275 Bois-Colombes Cedex

Cette édition s'applique à IBM® Rational® System Architect®, version 11.3.1 et à toutes les éditions et modifications ultérieures jusqu'à mention contraire dans les nouvelles éditions.

Table des matières

Table des matières

Table des matières

1

Introduction

Introduction

Ce chapitre décrit les principes de base de l'utilisation de Rational System Architect et Visual Basic for Applications (VBA). Vous y apprendrez comment utiliser les macros, l'Editeur de VBA et l'Explorateur d'objets.

Automatisation dans Rational System Architect

Rational System Architect est conçu pour s'utiliser avec VBA, ce qui permet de contrôler son environnement à l'aide de programmes et de contrôler d'autres applications à l'aide des outils d'automatisation OLE.

Microsoft VBA et son environnement de développement sont installés en même temps que Rational System Architect. L'environnement de programmation, l'environnement de débogage, le langage et le système d'aide sont identiques à ceux des autres applications compatibles VBA, notamment les produits de la suite Microsoft Office.

Les outils d'automatisation vous permettent d'intégrer d'autres applications dans Rational System Architect de deux manières. Vous pouvez utiliser Rational System Architect :

- comme contrôleur d'automatisation et appeler un objet d'automatisation OLE à partir d'un script Rational System Architect,
- comme serveur d'automatisation et appeler un objet d'automatisation OLE à partir d'une autre application compatible OLE.

Vous pouvez utiliser les macros de Rational System Architect pour :

- optimiser la fonctionnalité de Rational System Architect en automatisant la vérification des présentations et des règles des diagrammes,
- créer des diagrammes, des symboles et des définitions à partir d'informations contenues dans d'autres applications,

capturer les événements qui se produisent dans Rational System Architect et les enregistrer dans un fichier ou une base de données.

Programmation dans Rational System Architect avec VBA

Rational System Architect enregistre l'état de l'environnement du projet VBA dans le fichier d'initialisation SA2001.INI, que vous pouvez trouver dans le dossier Windows ou WinNt. La section **Macros** contient des données semblables à ce qui suit :

```
[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
```

L'extrait ci-dessus indique que le fichier de projet permanent SAAUTO.MAC est chargé au démarrage de Rational System Architect. Ce fichier sera caché et en lecture seule. Le fichier de projet temporaire SAMPLE.MAC est actif et chargé en lecture seule. Si l'un des fichiers de projet temporaires contient un module AutoExec, les macros s'exécutent automatiquement. Par défaut, les nouveaux projets s'ouvrent dans le mode lecture seule.

Exécution d'une macro

Pour exécuter des macros directement à partir de la boîte de dialogue **Macros**, procédez comme suit :

1. Dans le menu **Outils**, sélectionnez le sous-menu **Macros**.
2. Sélectionnez l'option **Exécuter la macro**.

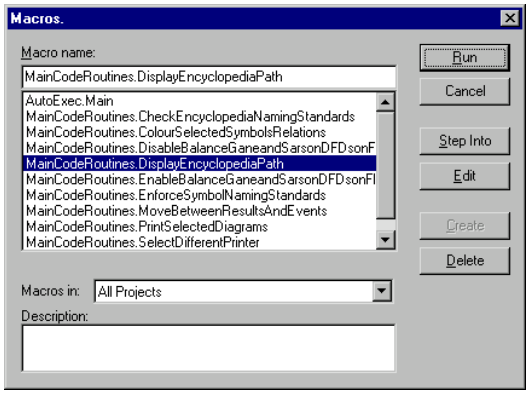


Figure 2-1 : Boîte de dialogue Macros

Vous pouvez également exécuter une macro à partir d'une barre d'outils.

Pour ajouter une macro dans une barre d'outils, procédez comme suit :

- Cliquez avec le bouton droit sur la barre de menus, ou sur une barre d'outils de votre choix, puis sélectionnez l'option **Personnaliser**.
- Sélectionnez l'onglet **Commandes** puis l'option **Macros**.
- Faites défiler les macros disponibles puis **faites glisser** la macro désirée sur une barre d'outils.

Projets de macro

Vous pouvez ajouter d'autres fichiers de projet temporaires à un modèle de projet.

Pour accéder à la boîte de dialogue **Projets de macro**, procédez comme suit :

1. Ouvrez le menu **Outils**.
2. Sélectionnez le sous-menu **Macros**.
3. Sélectionnez l'option **Projets de macro**.

Vous pouvez ajouter d'autres fichiers de projet dans la liste des projets disponibles ou supprimer des projets existants dans cette liste.

Vous pouvez désélectionner la case à cocher **Activer les macros automatiques** pour désactiver les macros de type AutoExec.

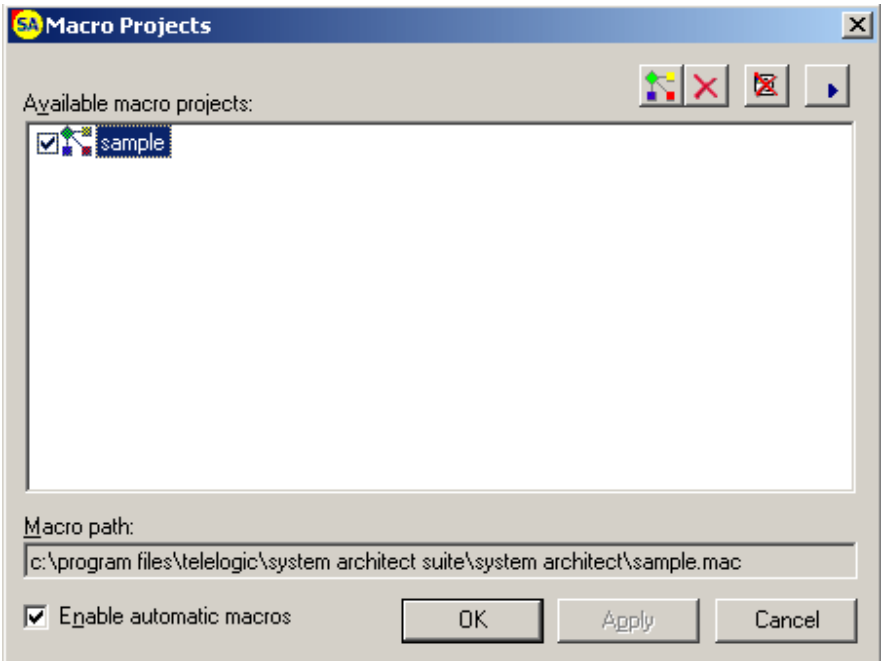


Figure 2-2 : Boîte de dialogue Projets de macro

Ajout d'un nouveau projet

Cliquez sur le bouton de commande **Ajouter** pour afficher la boîte de dialogue **Ouvrir le projet de macro**.

Les fichiers des projets Rational System Architect se reconnaissent par leur extension `.MAC`.

Vous pouvez sélectionner et ouvrir un projet existant, ou créer un nouveau projet, en saisissant un nom de fichier non présent dans la liste.

Si vous désirez modifier un projet, n'oubliez pas de désélectionner la case à cocher **Ouvrir en lecture seule** avant d'ouvrir le fichier de ce projet.

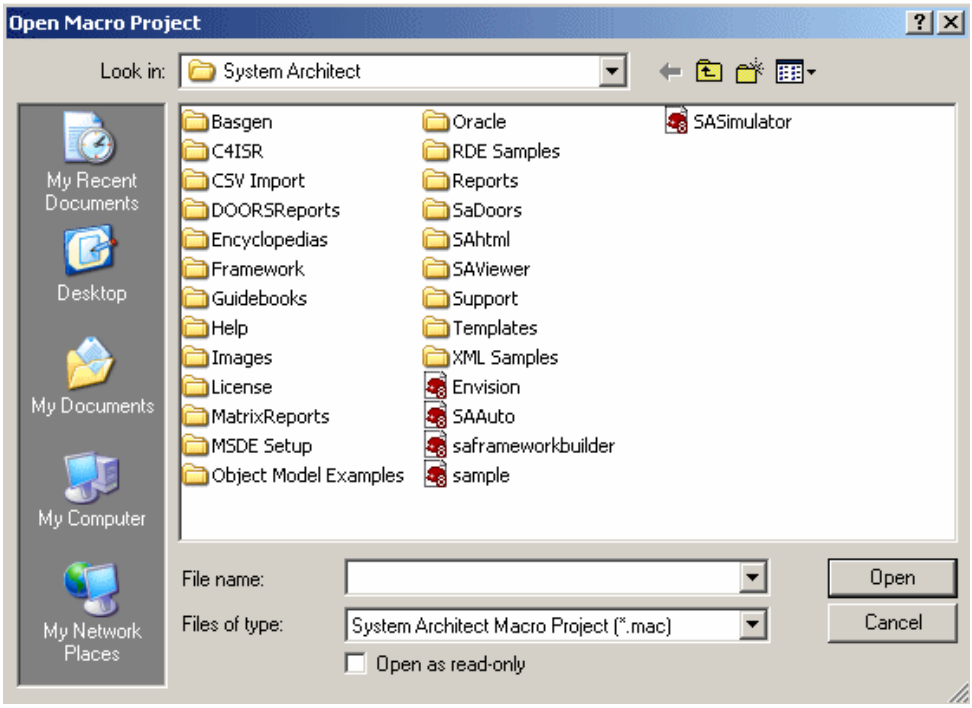


Figure 2-3 : Boîte de dialogue Ouvrir le projet de macro

Ouverture de l'Editeur de VBA

Pour créer une macro ou modifier une macro, vous devez préalablement ouvrir l'Editeur de VBA.

L'éditeur de VBA peut être ouvert de différentes façons :

- Dans le menu **Outils**, sélectionnez le sous-menu **Macros** puis l'option Editeur de VBA.
- Appuyez sur Alt+F11.
- Dans la boîte de dialogue **Macros**, cliquez sur le bouton de commande **Modifier**.
- Dans la boîte de dialogue **Projets de macro**, cliquez sur le bouton de commande **Appliquer** (s'il est actif) puis sur le bouton de commande **Exécuter la macro**. La boîte de dialogue **Macros** contenant le bouton de commande **Modifier** apparaît alors.

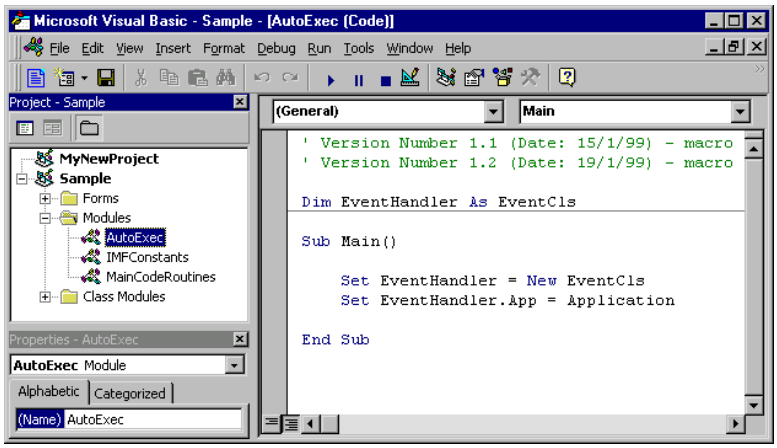


Figure 2-4 : Editeur de VBA

Explorateur de projets

L'Editeur de VBA comprend plusieurs fenêtres. La fenêtre Projet est habituellement ouverte lorsque vous accédez à l'Editeur de VBA pour la première fois.

Pour afficher la fenêtre **Projet**, procédez comme suit :

1. Dans le menu **Vue**, sélectionnez l'option **Explorateur de projets**.
2. Le projet modèle contient trois **groupes de dossiers** nommés **Formulaires**, **Modules** et **Classes**.
3. Vous pouvez ouvrir chaque dossier pour afficher son contenu. Pour cela, cliquez sur le signe "plus" situé en regard d'un dossier, ou double-cliquez sur le dossier lui-même ou sur son nom.

Le contenu de chaque dossier s'affiche par ordre alphabétique. Vous pouvez également afficher la liste alphabétique complète de tous les éléments d'un dossier.

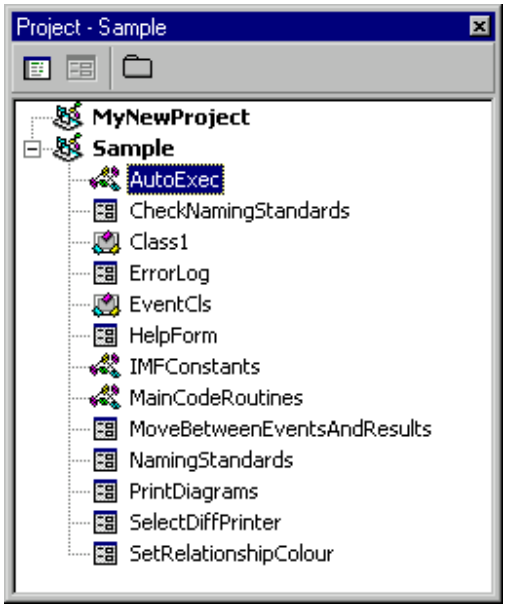


Figure 2-5 : Fenêtre Projet

Fenêtre Propriétés

Lorsque vous cliquez sur l'un des **Modules**, la fenêtre**Propriétés** affiche le nom du module activé. Vous pouvez alors modifier ce nom dans la fenêtre **Propriétés**.

Si vous double-cliquez sur le nom d'un **Module**, son code s'affiche.

Si vous double-cliquez sur le nom d'un **Formulaire**, l'**objet Formulaire** correspondant apparaît et la fenêtre Propriétés affiche la liste alphabétique des propriétés du formulaire.

Lorsque vous cliquez avec le bouton droit sur le nom d'un formulaire, vous pouvez ensuite **afficher** le **code du formulaire**.

La fenêtre **Propriétés** peut également afficher les propriétés par catégories.

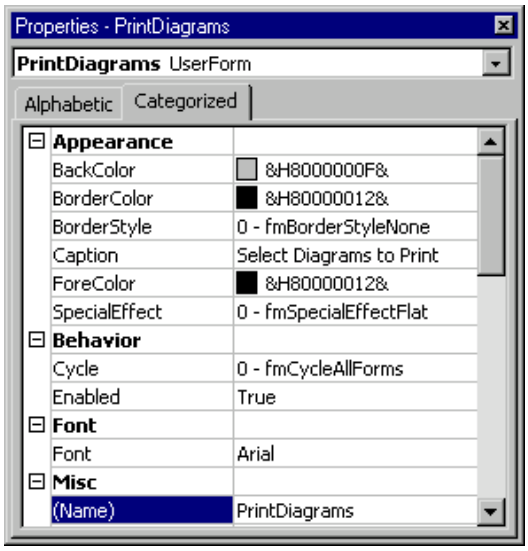


Figure 2-6 : Fenêtre Propriétés

Insertion de modules et de formulaires

Vous pouvez ajouter d'autres **modules** et **formulaires** à un projet à l'aide du menu **Insérer**.

Vous pouvez également **supprimer des modules** ou **formulaires sélectionnés**, et éventuellement les **exporter**, à l'aide de l'option **Supprimer** du menu **Fichier**.

Si vous cliquez avec le bouton droit sur la fenêtre **Projet**, vous accédez aux options **Insérer** et **Supprimer**.

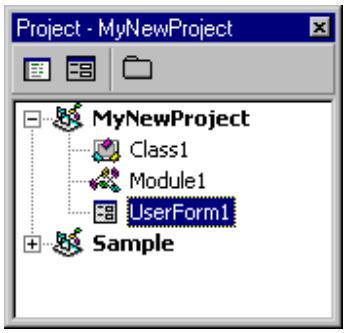


Figure 2-7 : Fenêtre Projet

Explorateur d'objets

L'Explorateur d'objets est une fonctionnalité de l'Editeur de VBA qui permet d'interroger différentes bibliothèques d'objets, bibliothèques de types et bibliothèques de liens dynamiques.

Référencement des fichiers de bibliothèque

Pour associer des bibliothèques supplémentaires à l'Explorateur d'objets, procédez comme suit :

- Dans le menu **Outils**, sélectionnez l'option **Références**.

En principe, la bibliothèque SA2001 doit apparaître dans la liste des références sélectionnées.

Vous pouvez faire défiler les éléments de la liste Références disponibles pour sélectionner d'autres références.

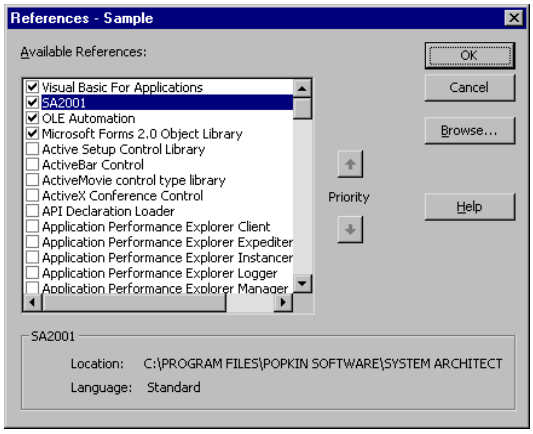


Figure 2-8 : Sélection des références

Pour afficher l'Explorateur d'objets, appuyez sur la touche de fonction F2, ou :

- Dans le menu **Vue**, sélectionnez l'option **Explorateur d'objets**.

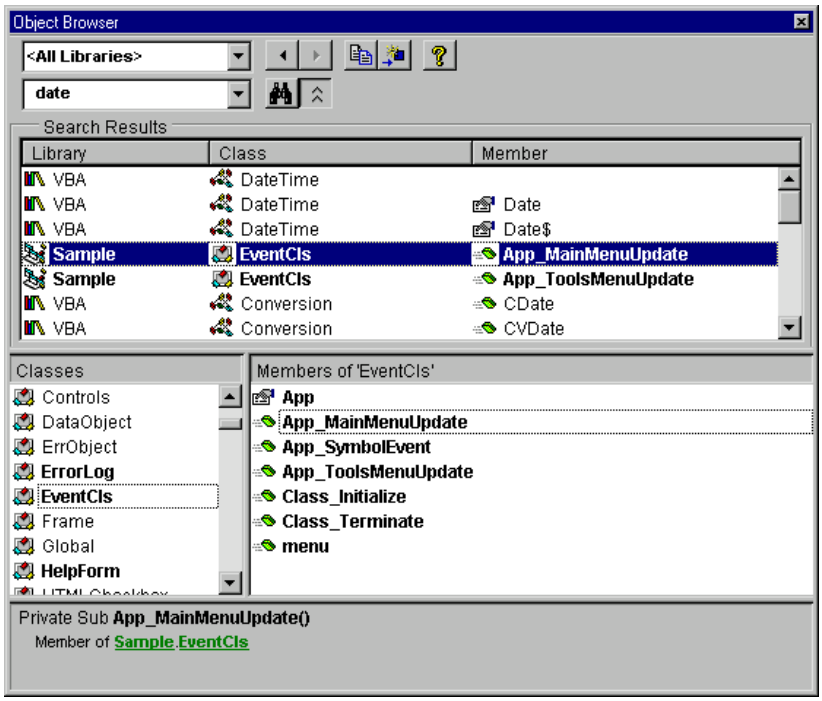


Figure 2-9 : Explorateur d'objets

Vous pouvez filtrer l'affichage de l'Explorateur d'objets pour ne voir que certaines bibliothèques, comme dans l'exemple ci-dessus, et rechercher des éléments dont le nom de classe ou de membre contient un texte spécifique.

Si vous sélectionnez un projet dans la fenêtre de l'Explorateur d'objets, ce dernier met en évidence tous les éléments qui font partie de ce projet.

En particulier, les éléments Classes, Constantes, Types énumératifs, Evénements, Variables globales, Méthodes, Modules, Propriétés et Types définis par l'utilisateur apparaîtront précédés d'une icône spécifique.

2

Automatisation dans Rational System Architect

Introduction

L'automatisation se définit par la capacité d'une application de déclarer et d'utiliser des objets créés par d'autres applications. Quand vous utilisez Visual Basic for Applications (VBA), vous pouvez créer un code afin d'accéder à des objets d'une ou plusieurs applications dans le même programme.

L'automatisation permet de créer une solution totalement intégrée basée sur les fonctionnalités de différents produits. VBA étant intégré à Rational System Architect, d'autres applications peuvent bénéficier de ses fonctionnalités. Les sections qui suivent décrivent cette fonctionnalité.

Ce chapitre décrit l'automatisation et explique de quelle façon elle permet de créer des solutions totalement personnalisables. Il détaille également comment configurer Rational System Architect pour apporter aux solutions développées toutes les fonctionnalités requises.

Automatisation

Grâce à l'automatisation, un programmeur peut incorporer des objets provenant d'autres applications dans son application afin de créer une solution intégrée. Par exemple, vous pouvez utiliser Rational System Architect pour créer un modèle de données développé par plusieurs utilisateurs et faire en sorte que tous les utilisateurs de l'entreprise puissent en voir les données avec un rapport MS Word ou une feuille de calcul Excel spécialement conçu à cet effet. Grâce à l'automatisation, vous pouvez exécuter un rapport Word ou Excel intégré directement à partir d'un menu personnalisé dans Rational System Architect. L'automatisation permet aussi aux utilisateurs d'utiliser des outils conçus pour exécuter des tâches également intégrées.

L'automatisation était autrefois appelée "automatisation OLE" ou encore "automatisation ActiveX".

Serveur et contrôleur d'automatisation

L'automatisation est le seul moyen de contrôler les objets d'une autre application. D'un point de vue technique, une première application contient le code VBA qui contrôle l'automatisation tandis qu'une deuxième application fournit les objets à utiliser. Le système qui pilote ce processus est appelé le contrôleur d'automatisation et le système qui fournit les objets est appelé le serveur d'automatisation.

Référencement de la bibliothèque de types

VBA possède un mécanisme qui lui permet de comprendre quel objet sera disponible dans un programme VBA. Si vous créez votre code dans Rational System Architect, VBA en déduira que sa bibliothèque d'objet d'automatisation est disponible mais il ne saura pas si d'autres types de bibliothèques le sont également. C'est pourquoi vous devez référencer les bibliothèques qui sont nécessaires.

La **bibliothèque de types** est une base de données qui contient des informations sur tous les objets disponibles pour l'automatisation dans toutes les applications. Ces informations comprennent des données sur les objets, les attributs, les événements et les méthodes disponibles dans l'application. Il s'agit généralement d'un fichier séparé qui est installé en même temps que l'application mais, dans certains cas, il peut être intégré dans le principal fichier exécutable.

Pour référencer les données de la bibliothèque de types, vous devez ouvrir l'Editeur de VBA. Vous pouvez ouvrir ce dernier à partir du menu Editeur VBA de l'application ou en appuyant simultanément sur les touches **Alt+F11**. Une fois l'Editeur de VBA ouvert, sélectionnez **Outils | Références**.

Dans cet exemple, les seuls objet référencés sont les objets standard de VBA et Rational System Architect. Pour inclure d'autres applications, faites défiler la liste et choisissez l'application désirée, par exemple la bibliothèque d'objets de Microsoft Excel 9.0. Ceci inclut tous les composants de la bibliothèque de types d'Excel dans l'application active.

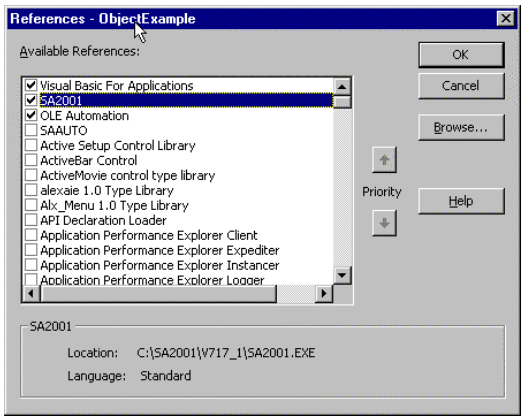


Figure 2-1 : Sélection d'une bibliothèque de types

Affichage des objets d'automatisation

Une fois la référence à une autre application définie, vous pouvez afficher la liste des objets, attributs et méthodes de toutes les bibliothèques de types actuellement référencées. L'Editeur de VBA possède son propre explorateur d'objets qui permet d'afficher toutes ces propriétés.

Pour accéder à l'Explorateur d'objets de VBA, appuyez sur la touche F2 ou sélectionnez l'option de menu **Vue | Explorateur d'objets** dans l'Editeur de VBA.

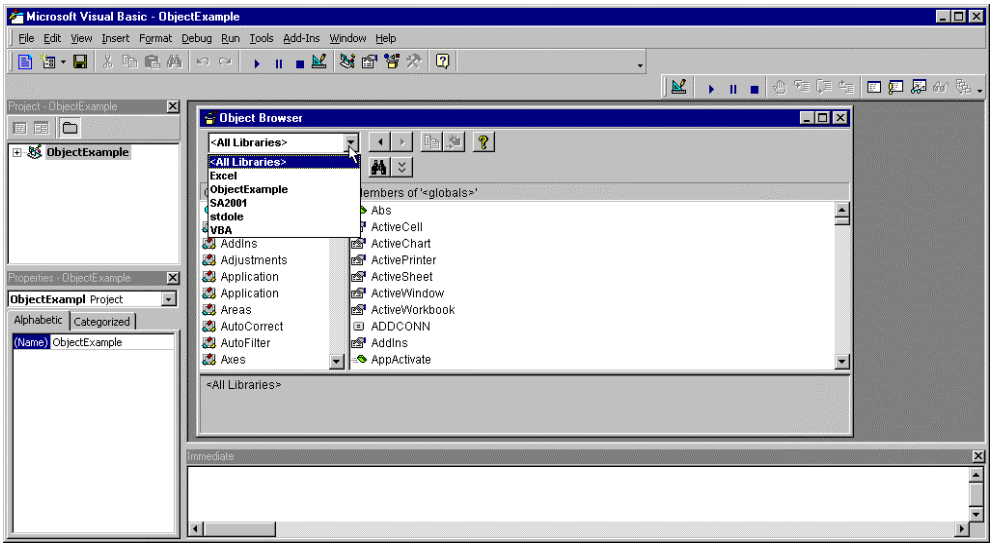


Figure 2-2 : Explorateur d'objets VBA

L'Explorateur d'objets répertorie toutes les bibliothèques sélectionnées dans les références de bibliothèque de types et vous pouvez aussi sélectionner individuellement un ensemble d'objets donné. Dans l'exemple ci-après, nous avons choisi d'afficher la bibliothèque de types SA2001, en particulier la classe Definition des objets d'automatisation.

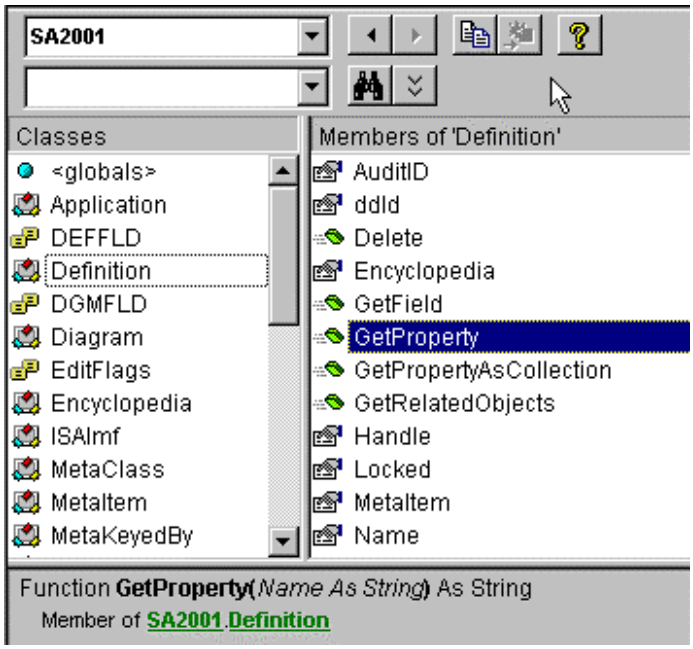


Figure 8-3 : Affichage de la bibliothèque de types SA2001

Les membres de la classe Definition sont répertoriés et la méthode **GetProperty** est sélectionnée. Les paramètres sont détaillés dans la fenêtre de navigation du bas, notamment leur type et leur type de valeur de retour. Une fois la référence à un objet d'automatisation déclarée, toutes les données internes de l'application deviennent disponibles pour le programmeur VBA qui peut ensuite développer une solution intégrée.

Création d'une instance de l'application

Bien que les applications requises soient référencées par VBA, elles ne sont réellement utilisables qu'à compter du moment où un code permet de contrôler les objets de ces applications. La section suivante de ce chapitre décrit comment définir une nouvelle instance de l'application afin d'accéder à ses objets automatisés et comment déclarer celle-ci.

Une instance est une session de l'application requise. Pour pouvoir utiliser les objets d'automatisation, l'application doit résider en mémoire (sans être forcément visible). Pour accéder aux objets, VBA utilise les instructions Dim et Set afin de déclarer l'instance d'un objet d'automatisation exactement comme une déclaration serait définie pour un type

intégré. La seule différence avec l'automatisation est qu'à l'emplacement où un objet doit être utilisé, vous devez créer une nouvelle instance de cet objet avec l'instruction Set.

Le code suivant déclare la variable *ExObj* comme une application Excel. Notez le nom complet server.class de la déclaration ; la classe d'application référencée est celle d'Excel. Il existe d'autres applications dont des classes peuvent s'appeler "application".

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

La ligne de code suivante crée une instance de l'application Excel et définit l'emplacement du code où cette instanciation a lieu.

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

La variable *ExObj* peut maintenant accéder aux objets d'automatisation d'Excel, par exemple pour afficher la boîte de dialogue standard d'Excel **Ouvrir** à l'aide de l'objet d'automatisation **GetOpenFilename()** qui possède un filtre de fichier texte. Le code utilisé est le suivant :

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

Tous les objets d'automatisation d'Excel sont à présent disponibles ainsi que ceux de Rational System Architect.

Libération de l'instance de l'application

Les instances des objets d'automatisation sont créées avec la commande Set, manipulées avec des appels de fonction d'automatisation puis sont fermées. La fermeture de la classe Automatisation permet de libérer les ressources de la mémoire.

Vous devez utiliser le code suivant quand un objet d'automatisation n'est plus utilisé. Ceci aura pour effet de fermer le serveur d'automatisation.

```
Set Exobj = Nothing
```

Récapitulatif

Dim Object = Server.class	Déclaration du serveur
Set Object = New Server.class	Instanciation du serveur
Code d'automatisation	Code VBA qui accède aux objets d'automatisation du serveur
Dim Object = Nothing	Libération du serveur d'automatisation dans la mémoire

Solutions personnalisables

La section précédente présentait les principes de l'automatisation et décrivait comment intégrer les objets d'une ou plusieurs applications dans d'autres applications. Tout produit qui prend en charge VBA et l'automatisation peut être regardé comme une solution personnalisable. Cela signifie que tout produit acheté dans le commerce et qui prend en charge VBA peut être modifié à volonté. Ceci ne se réduit toutefois pas à l'intégration avec d'autres applications. Une grande partie du travail de personnalisation d'une application demande également de modifier le mode de fonctionnement du produit.

Les différents utilisateurs d'un produit peuvent envisager la personnalisation de plusieurs manières, par exemple modifier des options de menu, intégrer le produit dans des applications MS Office, automatiser des tâches répétitives, mais toutes ces opérations demanderont d'utiliser VBA et l'automatisation. Le tableau suivant répertorie cinq raisons principales pour lesquelles la personnalisation d'un produit peut être envisagé. Toutes ces motivations peuvent être satisfaites avec VBA.

Catégorie	Signification	Exemple dans Rational System Architect
Modification du mode opératoire de l'application	Modification du mode de fonctionnement de l'application pour satisfaire les règles et les procédures de l'entreprise	Créez un symbole sur un diagramme. Ainsi les règles de dénomination de l'entreprise seront vérifiées et l'utilisateur sera repéré si ces règles sont enfreintes.
Automatisation des tâches répétitives	Combinez des ensembles de tâches manuelles ordinaires en séries d'actions qui pourront être exécutées sans limite.	Impression d'une série de diagrammes définie par l'utilisateur
Extension des fonctionnalités d'une application	Ajout de nouvelles fonctions dans une application existante	Création automatique d'un diagramme de carte de processus issu d'un organigramme des processus avec les rôles attribués et les unités organisationnelles.
Intégration à d'autres applications	Contrôler une autre application pour disposer de fonctionnalités	Créez une feuille de calcul Excel des données CRUD de corrélation entre les processus

	normalement indisponibles.	et les entités.
Accès aux données de l'entreprise	Echangez des données avec des bases de données distantes et des applications ne pouvant normalement pas accéder à des bases de données.	Importation automatique à partir de différentes sources d'informations ne pouvant normalement pas être importées.

La combinaison de Rational System Architect et de VBA fournit un environnement de développement intégré standard qui permet de créer des solutions personnalisables en incorporant tous les éléments de programmation fournis par Microsoft, en particulier les formulaires IntelliSense et Microsoft.

Planification d'une solution automatisée avec Rational System Architect

Les bénéfices de l'automatisation vont de la réduction des tâches répétitives à la création d'une application totalement intégrée. La section qui suit détaille quelques-unes des modifications que vous pouvez apporter à Rational System Architect.

Contrôle du mode opératoire

Rational System Architect réagit à certains événements qui peuvent survenir en cours d'opération. Ces événements peuvent déclencher l'exécution d'un code qui automatisera certaines tâches.

Les événements pris en charge par Rational System Architect comprennent le démarrage et la fermeture du produit, l'ouverture et la fermeture d'une encyclopédie, l'ouverture, la fermeture et la sauvegarde d'un diagramme, le changement d'un ID d'audit, et plusieurs événements liés aux symboles. Les événements liés aux symboles comprennent le placement d'un symbole sur un diagramme, l'attribution d'un nom, la connexion et la déconnexion (avec un symbole de ligne) et la suppression.

Cette fonctionnalité permet de modifier en temps réel le fonctionnement de Rational System Architect à l'aide de VBA.

Contrôle de l'apparence des objets

Rational System Architect 10.1 permet de personnaliser les menus de Rational System Architect et d'ajouter ou d'enlever des commandes dans la structure de menus existante. Par exemple, vous pouvez ajouter une commande afin d'exécuter une macro. Avant Rational System Architect version 10.1 (versions 10.0 et précédentes), la seule manière d'ajouter des commandes dans des options de menu était d'utiliser Rational System Architect VBA. Par exemple, vous pouviez mettre à jour un menu contenant des options spécifiques à une méthodologie pendant l'exécution d'un code. Ces options de menu étaient alors affichées uniquement pour un diagramme spécifique en fonction de la méthodologie sélectionnée.

Dans les deux cas, des options de menu et des menus instantanés peuvent être ajoutés aux menus et des images bitmap peuvent leur être associées. Cette technique est souvent utilisée pour ajouter une macro à une option de menu et lui associer une image bitmap. Cette macro peut aussi être ajoutée à une barre d'outils ainsi qu'à un menu.

Si vous souhaitez modifier des macros conçues pour être utilisées dans des options de menu afin de les réutiliser dans Rational System Architect version 10.0 ou antérieure, vous trouverez les instructions requises à cet effet dans le manuel de conversion de Rational System Architect (fichier conversion.pdf) disponible sur le site de support IBM.

Tâches automatisées

Vous pouvez créer une macro VBA pour générer des rapports à partir des informations du référentiel et exécuter des tests de cohérence. Tous les objets et les propriétés contenus dans le dictionnaire de données peuvent être créés, lus, mis à jour ou supprimés sur la base de règles définies dans le code. Les tâches répétitives de base, comme la synchronisation d'une valeur entre deux dictionnaires de données, peuvent être automatisées.

Exécution de contrôles

Vous pouvez appliquer une série prédéfinie de normes au modèle VBA en temps réel en utilisant les membres commandés par les événements du modèle VBA de Rational System Architect. Il peut s'agir par exemple d'imposer une norme de dénomination ou de renseigner un champ obligatoire.

Interface entre des applications externes

Vous pouvez utiliser VBA pour importer, exporter, lire, créer, modifier, mettre à jour ou supprimer des objets sur la base de valeurs contenues dans d'autres applications. Les fonctions concernées sont notamment créer des diagrammes, des symboles et des définitions, établir des relations entre des symboles ou modifier les propriétés d'un dictionnaire de données.

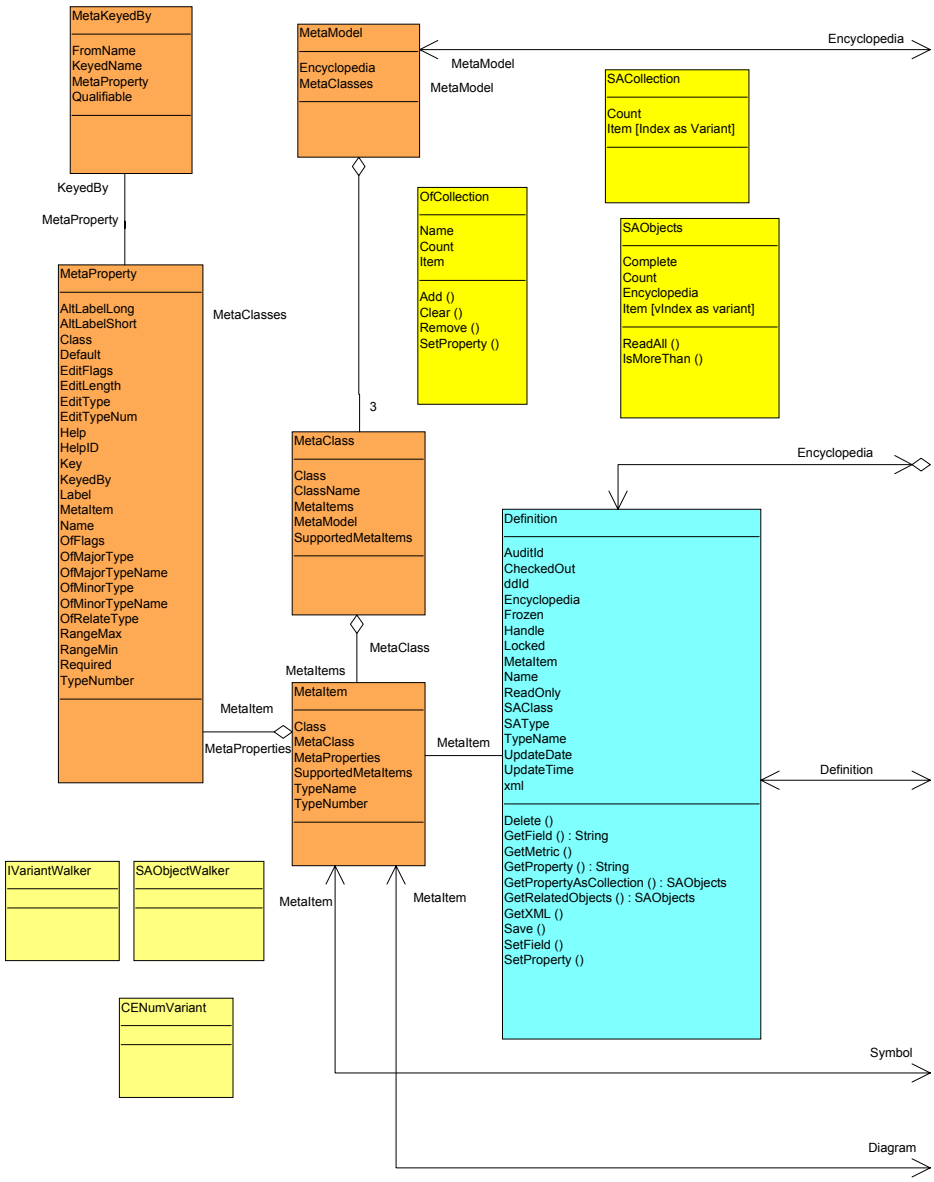
3

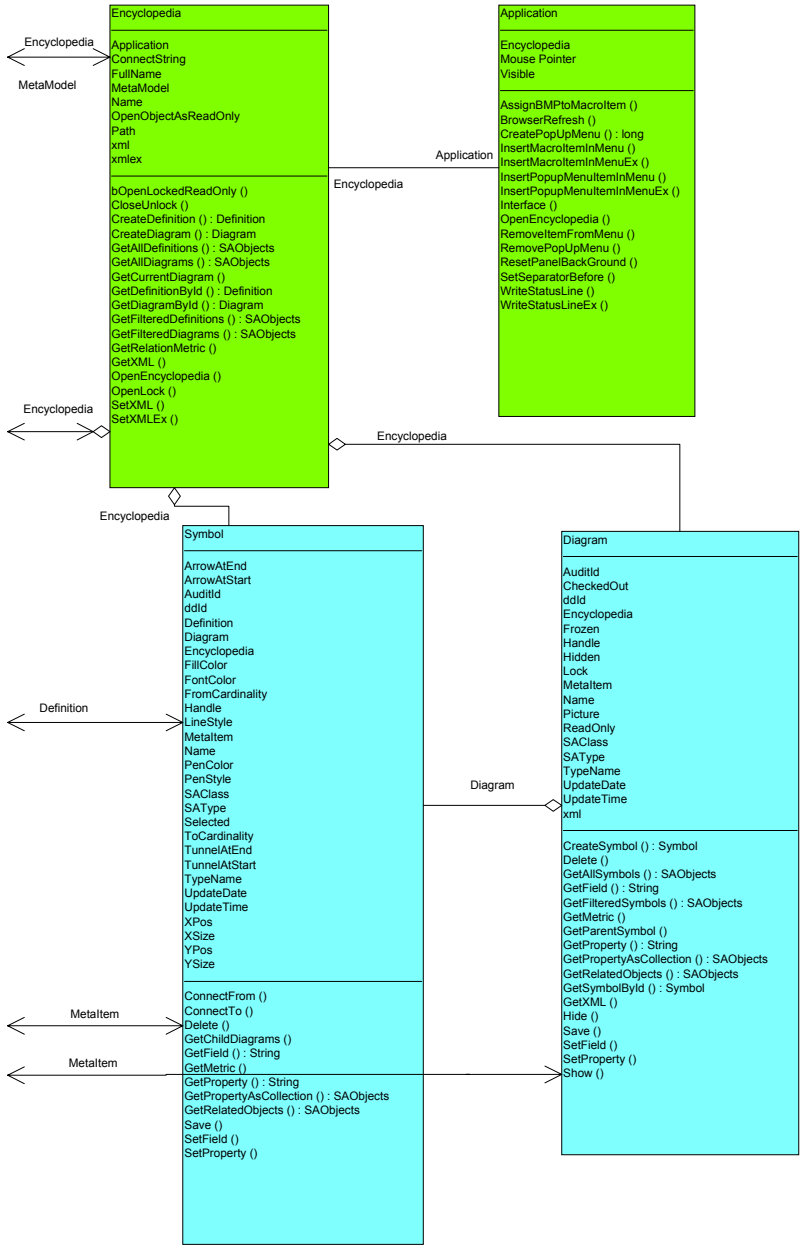
Modèle d'objet de Rational System Architect

Vous pouvez accéder aux objets de Rational System Architect qui sont utilisables par VBA au moyen de l'Explorateur d'objets de l'Editeur de VBA. Chaque type d'objet est défini sous la forme d'une classe qui contient une liste de propriétés et de méthodes prises en charge.

Pour visualiser l'ensemble du modèle d'objet, il est plus facile d'utiliser un diagramme des classes. L'illustration qui suit montre une partie du modèle d'objet de Rational System Architect représentée dans Rational System Architect à l'aide de la notation de diagramme de classes UML.

Modèle d'objet de Rational System Architect




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
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

Classes du modèle d'objet

Voici les classes du modèle d'objet de Rational System Architect avec des exemples d'utilisation pour chacun.

Classe	Exemples d'utilisation
Application	Evénements de Rational System Architect Manipulation de menu
Encyclopedia	Créer des objets de définition et de diagramme Extraire des objets de définition et de diagramme
Diagram	Créer et extraire des objets de symbole Manipulation des propriétés d'un diagramme
Symbol	Manipulation des propriétés d'un symbole Informations sur les connexions d'un symbole Extraire les objets de diagramme enfant
Definition	Manipulation des propriétés d'une définition Manipulation d'une définition apparentée
SACollection	Collection de propriétés Rational System Architect

SAObjects	Collection d'objets Rational System Architect (diagrammes, symboles et définitions)
OfCollection	Collection de définitions ou de diagrammes OneOf ou ListOf
MetaModel MetaClass MetaItem	Objets de méta-modèle de Rational System Architect (types de définitions, de symboles et de diagrammes pris en charge)
MetaKeyedBy	Méta-modèle de Rational System Architect saisis par des définitions (saisis par des définitions avec leur structure)
MetaProperty	Ensemble de propriétés de méta-modèle de Rational System Architect (propriétés prises en charge avec leur structure dans chaque type de définition)

4

Classe Application

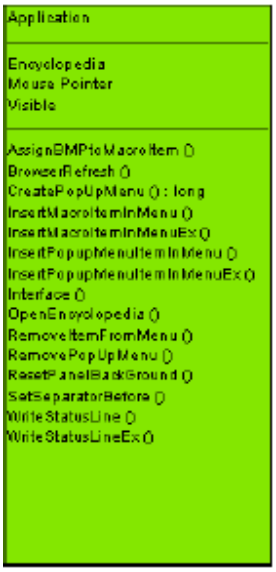
Rubriques contenues dans ce chapitre	Page
Attributs	4-3
Méthodes	4-5

Introduction

Cette rubrique décrit l'objet Application de Rational System Architect qui permet de contrôler l'interface utilisateur. Il s'agit du niveau le plus élevé dans le modèle d'objet.

Une instance de l'objet Application est créée comme suit :

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



Attributs

Encyclopedia

Finalité
Il s'agit de l'objet Encyclopédie, qui facilite l'accès aux méthodes et aux attributs de la classe d'encyclopédie.

Paramètres
Lecture seule

Exemple

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

MousePointer

Finalité
Cette attribut permet de contrôler le type de pointeur de la souris que l'utilisateur peut voir dans l'application.

Paramètres
Type de données : entier

Exemple
Vous pouvez obtenir la valeur courante de l'attribut Mousepointer comme suit :

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

Pour utiliser un pointeur de souris de type sablier, entrez la valeur 11.

```
oApplication.Mousepointer  = 11
```

Les fichiers d'aide de Visual Basic (VB) contiennent une liste complète des valeurs admises.

Visible

Finalité
Détermine si l'application est en cours de fonctionnement ou non. Si vous affectez la valeur "False" à cet attribut, l'application se ferme.

Paramètre
Type de données : booléen

Exemple
```
oApplication.Visible = False
```

Méthodes

AssignBMPtoMacroItem

Finalité
Associe un fichier bitmap, par exemple une icône de menu, à une macro VBA.

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`MacroName`

Utilisation : obligatoire

Type de données : chaîne

Tout projet de macro valide

Syntaxe : “Project Name, Module Name, Subroutine Name()”

`BMPFileName`

Utilisation : obligatoire

Type de données : chaîne

Chemin et nom de fichier de l'image BMP (par exemple C:\Windows\world.bmp)

BrowserRefresh

Finalité
Régénère l'explorateur de Rational System Architect. Tous les éléments ajoutés à l'encyclopédie après la dernière réactualisation sont affichés.

Syntaxe

```
Application Object.BrowserRefresh
```

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

CreatePopUpMenu

Finalité

Crée un menu instantané à insérer dans l'interface utilisateur. Vous pouvez associer le fichier bitmap d'une icône au menu instantané.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`PopUpName`

Utilisation : obligatoire

Type de données : chaîne

Nom du nouveau menu instantané

`BMPFileName`

Utilisation : facultatif

Type de données : chaîne

Chemin et nom de fichier de l'image BMP (par exemple C:\Windows\world.bmp)

InsertMacroItemInMenu

Finalité

Crée dans le menu Rational System Architect une option de menu qui appelle un sous-programme de macro existant.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`MacroName`

Utilisation : obligatoire

Type de données : chaîne

Tout projet de macro valide

Syntaxe : “Project Name, Module Name, Subroutine Name()”

`MacroItemCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom de la macro qui sera insérée dans un menu SA

`InMenuTitleCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom du menu instantané SA dans lequel la macro sera placée

`BeforeMenuItemCaption`

Utilisation : facultatif

Type de données : chaîne

Nom de l'option de menu instantané SA avant laquelle la macro sera placée

Remarque : Si vous n'utilisez pas cet attribut, la macro est placée à la fin du menu instantané.

InsertMacroItemInMenuEx

Finalité

Crée dans le menu Rational System Architect une option de menu qui appelle un sous-programme de macro existant. Cette méthode est une extension de la méthode InsertMacroItemInMenu.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`MacroName`

Utilisation : obligatoire

Type de données : chaîne

Tout projet de macro valide

Syntaxe : “Project Name, Module Name, Subroutine Name()”

`MacroItemCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom de la macro qui sera insérée dans un menu SA

`InMenuTitleCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom du menu instantané SA dans lequel la macro sera placée

`BeforeMenuItemCaption`

Utilisation : facultatif

Type de données : chaîne

Nom de l'option de menu instantané SA avant laquelle la macro sera placée

Remarque : Si vous n'utilisez pas cet attribut, la macro est placée à la fin du menu instantané.

`Balise`

Utilisation : facultatif

Type de données : chaîne

Affecter une balise spécifique à chaque option de menu permet à différentes options de menu d'avoir le même nom et d'appeler le même sous-programme. Quand le programme appelle une de ces options de menu, sa balise indique au sous-programme la partie de son code qu'il doit exécuter. Par exemple, un utilisateur peut créer une macro qui génère différents rapports dans MS Word. Au lieu d'écrire des sous-programmes distincts pour chaque type de rapport Word, l'utilisateur peut écrire tout le code dans un seul sous-programme et spécifier dans chaque option de menu différentes balises qui désigneront différentes fonctions à l'intérieur du code.

`bAfterSepartor`

Utilisation : facultatif

Type de données : booléen

Cet attribut s'utilise uniquement quand l'option de menu qui suit la macro placée par l'utilisateur est elle-même précédée d'un séparateur de ligne. Si vous affectez à cet attribut la valeur "true", la macro est placée après le séparateur de ligne. Si vous entrez "False" ou que vous n'utilisez pas cet attribut, la macro est automatiquement placée avant le séparateur de ligne.

InsertPopUpMenuItemInMenu

Finalité

Crée une option de menu instantané dans une option de menu existante de Rational System Architect.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`PopUpName`

Utilisation : obligatoire

Type de données : chaîne

Nom du nouveau menu instantané

`InMenuTitleCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom du menu instantané SA dans lequel le nouveau menu instantané sera placé

`BeforeMenuItemCaption`

Utilisation : facultatif

Type de données : chaîne

Nom de l'option de menu instantané SA avant laquelle le nouveau menu instantané sera placé

Remarque : Si vous n'utilisez pas cet attribut, le nouveau menu instantané est placé à la fin du menu instantané existant.

InsertPopUpMenuItemInMenuEx

Finalité

Crée une option de menu instantané dans une option de menu existante de Rational System Architect. Cette méthode est une extension de la méthode InsertPopUpMenuItemInMenu.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`PopUpName`

Utilisation : obligatoire

Type de données : chaîne

Nom du nouveau menu instantané

`InMenuTitleCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom du menu instantané SA dans lequel le nouveau menu instantané sera placé

`BeforeMenuItemCaption`

Utilisation : facultatif

Type de données : chaîne

Nom de l'option de menu instantané SA avant laquelle le nouveau menu instantané sera placé

Remarque : Si vous n'utilisez pas cet attribut, le nouveau menu instantané est placé à la fin du menu instantané existant.

`bAfterSepartor`

Utilisation : facultatif

Type de données : booléen

Cet attribut s'utilise uniquement quand l'option de menu qui suit la macro placée par l'utilisateur est elle-même précédée d'un séparateur de ligne. Si vous affectez à cet attribut la valeur "true", la macro est placée après le séparateur de ligne. Si vous entrez "False" ou que vous n'utilisez pas cet attribut, la macro est automatiquement placée avant le séparateur de ligne.

Interface

Finalité
Cette méthode est rarement employée. Elle permet d'appeler une instance d'une interface à l'aide d'une chaîne de texte au lieu d'une référence explicite.

Exemple

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

OpenEncyclopedia

Finalité
Ouvre une encyclopédie Rational System Architect.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`EncyclopediaPath`

Utilisation : obligatoire

Type de données : chaîne

Chemin de fichier de l'encyclopédie

La variable (EncyclopediaPath) correspond à un fichier udl. Les fichiers udl sont indirectement créés dans le chemin suivant : C:\Document and Settings\<nom_utilisateur>\Local Settings\Application Data\Telelogic\System Architect\Temp UDL files. Ces fichiers udl ont un nom du type SA_563.udl. Rational System Architect doit être ouvert pour que ce chemin soit visible.

OpenEncyclopediaUsingConnectionString

Finalité

Ouvre une encyclopédie Rational System Architect à l'aide d'une chaîne de connexion.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`strConnection`

Utilisation : obligatoire

Type de données : chaîne

`strConnection correspond à une chaîne contenue dans le fichier UDL.`

`Par exemple :`

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

OpenEncyclopediaUsingDisplayName

Finalité
Ouvre une encyclopédie Rational System Architect à l'aide de son nom affiché.

Syntaxe
Application Object.OpenEncyclopediaUsingDisplayName(strDisplayName)

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`strDisplayName`

Utilisation : obligatoire

Type de données : chaîne

strDisplayName correspond au nom affiché dans la barre de légende de SA, c'est-à-dire à connection-name(encyc-name).

Exemple :

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

RemoveItemFromMenu

Finalité
Cette méthode supprime une option de menu dans un menu instantané ou une option de menu de Rational System Architect.

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`ItemCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom de l'option de menu à supprimer du menu instantané

`FromMenuTitleCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom du menu instantané dans lequel l'option de menu doit être supprimée

RemovePopUpMenu

Finalité

Le menu instantané spécifié sera supprimé du système de menus de Rational System Architect.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`PopUpName`

Utilisation : obligatoire

Type de données : chaîne

Nom du menu instantané à supprimer

ResetPanelBackGround

Finalité
Réinitialise la couleur d'arrière-plan du panneau de la barre d'état

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`Panel`

Utilisation : obligatoire

Type de données : long

Les panneaux sont les volets ou les sections de la barre d'état. Ils vont de 1 pour le plus à gauche à 4 pour le plus à droite (les panneaux 2 et 3 apparaissent uniquement quand un symbole est sélectionné).

SetSeparatorBefore

Finalité
Place une barre de séparateur avant une option de menu dans un menu indiqué.

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`ItemCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom de l'option de menu avant laquelle la barre de séparateur sera placée.

`FromMenuTitleCaption`

Utilisation : obligatoire

Type de données : chaîne

Nom du menu instantané SA dans lequel le séparateur sera placé

`bHasSeparator`

Utilisation : obligatoire

Type de données : booléen

Cet attribut a la valeur "true" ou "false" selon que la barre de séparateur est présente ou non.

WriteStatusLine

Finalité

Permet d'afficher des courts messages dans la barre d'état de Rational System Architect (dans l'angle inférieur gauche) pour informer l'utilisateur qu'un code est en cours d'exécution.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`TextToShow`

Utilisation : obligatoire

Type de données : chaîne

Texte qui doit apparaître dans la barre d'état

WriteStatusLineEx

Finalité

Permet d'afficher des courts messages dans la barre d'état de Rational System Architect (dans l'angle inférieur gauche) pour informer l'utilisateur qu'un code est en cours d'exécution. Cette méthode est une extension de la méthode WriteStatusLine.

Syntaxe

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

`Application Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Application instanciée

`Panel`

Utilisation : obligatoire

Type de données : long

Les panneaux sont les volets ou les sections de la barre d'état. Ils vont de 1 pour le plus à gauche à 4 pour le plus à droite (les panneaux 2 et 3 apparaissent uniquement quand un symbole est sélectionné).

`TextToShow`

Utilisation : obligatoire

Type de données : chaîne

Texte qui doit apparaître dans la barre d'état

`BackColor`

Utilisation : obligatoire

Type de données : long

Couleur d'arrière-plan de la barre d'état

`ForeColor`

Utilisation : obligatoire

Type de données : long

Couleur d'avant-plan de la barre d'état

5

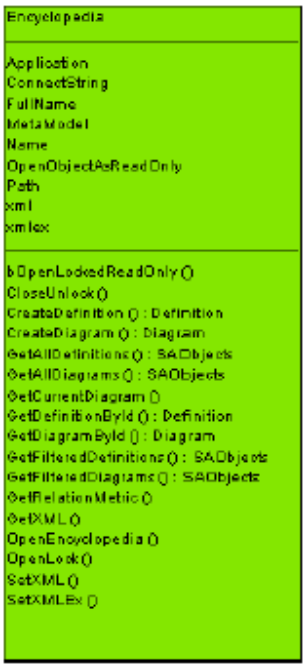
Classe Encyclopedia

Rubriques contenues dans ce chapitre	Page

Introduction

Ce chapitre décrit l'objet encyclopédie. Cet objet permet d'accéder aux méthodes et aux attributs d'encyclopédie détaillés ci-après.

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



Attributs

Application

Finalité
L'objet application renvoie l'objet application parent de l'objet encyclopédie actif.

Paramètres
Lecture seule

ConnectString

Finalité
Informations requises pour se connecter à une encyclopédie.

Paramètres :
Type de données : chaîne

Lecture seule

FullName

Finalité
Nom de l'encyclopédie active avec son chemin complet.

Paramètres
Lecture seule

Type de données : chaîne

MetaModel

Finalité
Classe MetaModel. Cette classe facilite l'accès à tous les attributs de métamodèle.

Paramètres
Lecture seule

Nom

Finalité
Renvoie le nom de l'encyclopédie active.

Paramètres
Lecture seule

Type de données : chaîne

OpenObjectsAsReadOnly

Finalité
Indique si les objets du modèle d'objet SA s'ouvrent en lecture seule.

Paramètres
Type de données : booléen

Chemin d'accès

Finalité
Chemin de l'encyclopédie active.

Paramètres
Type de données : chaîne

Lecture seule

Xml

Finalité
Chaîne XML de l'encyclopédie. Ce paramètre est utilisé par les méthodes GetXML et SetXML.

Paramètres
Type de données : chaîne

XmlEx

Finalité
Chaîne XML de l'encyclopédie. Ce paramètre est utilisé par la méthode SetXMLEx.

Paramètres
Type de données : chaîne

Méthodes

bOpenLockedReadOnly

Finalité
Cette méthode renvoie la valeur "True" si l'attribut OpenObjectsAsReadOnly a la valeur "True" ou que l'encyclopédie a été ouverte en lecture seule.

CloseUnlock

Voir OpenLock ci-après.

CreateDefinition

Finalité
Crée une instance de la classe Definition avec un type et un nom de définition spécifiés.

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom de la nouvelle définition

`SAType`

Utilisation : obligatoire

Type de données : long

Type de la définition Rational System Architect qui est créée (par exemple DFXPROCESS ou 3)

Remarque : Reportez-vous au fichier DEFNS.BAS contenu dans le répertoire de Rational System Architect pour obtenir la liste complète des définitions de SA ainsi que leurs numéros et noms constants internes.

Remarque : Pour créer et conserver une définition SA, vous devez appeler la méthode Save de la définition, sans quoi la nouvelle définition sera supprimée quand l'encyclopédie fermera.

CreateDiagram

Finalité

Crée une instance de la classe Diagram avec un type et un nom de diagramme spécifiés.

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom du nouveau diagramme

`SAType`

Utilisation : obligatoire

Type de données : long

Type du diagramme Rational System Architect qui est créé (par exemple GTCATPROCESSFLOW ou 89)

Remarque : Reportez-vous au fichier DIAGRAMS.BAS contenu dans le répertoire de Rational System Architect pour obtenir la liste complète des diagrammes de SA ainsi que leurs numéros et noms constants internes.

GetAllDefinitions

Finalité

Cette méthode renvoie toutes les définitions contenues dans l'encyclopédie sous la forme d'une collection de définitions.

Règles

Vous devez définir la taille de la variable SAObjects et l'identifier comme une collection de définitions.

Exemple

```
Dim oCollectionofDefinitions As SAObjects

Set oCollectionofDefinitions =
  oEncyclopedia.GetAllDefinitions

Call oCollectionofDefinitions.ReadAll
```

La collection SAObjects ne sera complétée que lorsque l'indicateur Complete de la collection aura la valeur "true". Utilisez la méthode GetAllDefinitions avec les méthodes ReadAll ou IsMoreThan.

GetAllDiagrams

Finalité

Cette méthode renvoie tous les diagrammes contenus dans l'encyclopédie sous la forme d'une collection de diagrammes.

Règles
Vous devez définir la taille de la variable SAObjects et l'identifier comme une collection de diagrammes.

Exemple

```
Dim oCollectionofDiagrams As SAObjects

Set oCollectionofDiagrams =
  oEncyclopedia.GetAllDiagrams

Call oCollectionofDiagrams.ReadAll
```

La collection SAObjects ne sera complétée que lorsque l'indicateur Complete de la collection aura la valeur "true". Utilisez la méthode GetAllDiagrams avec les méthodes ReadAll ou IsMoreThan.

GetCurrentDiagram

Finalité
Cette méthode renvoie le diagramme actuellement ouvert sous la forme d'un objet diagramme.

Règles
Vous devez définir la taille d'un objet diagramme et définir ce dernier comme étant le diagramme actuellement ouvert. Voir l'exemple ci-dessous.

Exemple

```
Dim OCurrentDiagram As Diagram

Set OCurrentDiagram = oEncyclopedia.GetCurrentDiagram
```

GetDefinitionById

Finalité
Cette méthode renvoie une définition sous la forme d'un objet définition à partir de son identité spécifiée.

Syntaxe

```
Encyclopedia Object.GetDefinitionById(Id)

Encyclopedia Object
```

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`Id`

Utilisation : obligatoire

Type de données : long

Toutes les définitions enregistrées dans Rational System Architect sont identifiées de manière unique et interne au moyen d'un identifiant de dictionnaire de données.

Exemple

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

GetDiagramById

Finalité

Tous les diagrammes enregistrés dans Rational System Architect sont identifiés de manière unique et interne au moyen d'un identifiant de dictionnaire de données. Cette méthode renvoie un diagramme sous la forme d'un objet diagramme à partir de son identité spécifiée.

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`Id`

Utilisation : obligatoire

Type de données : long

Tous les diagrammes enregistrés dans Rational System Architect sont identifiés de manière unique et interne au moyen d'un identifiant de dictionnaire de données.

Exemple

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

GetFilteredDefinitions

Finalité

Renvoie une collection filtrée de définitions contenues dans une encyclopédie.

Paramètres

Type de données : objets SA

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`WildCardName`

Utilisation : obligatoire

Type de données : chaîne

Critères de filtrage (par exemple "C*" = toutes les définitions débutant par "C")

Remarque : La recherche avec caractère générique distingue les majuscules des minuscules.

`SAType`

Utilisation : obligatoire

Type de données : long

Type de la définition Rational System Architect qui est récupérée (par exemple DFXPROCESS ou 3)

Remarque : Reportez-vous au fichier DEFNS.BAS contenu dans le répertoire de Rational System Architect pour obtenir la liste complète des définitions de SA ainsi que leurs numéros et noms constants internes.

Exemple
L'expression suivante renvoie toutes les définitions de processus commençant par "C".

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

La collection SAObjects ne sera complétée que lorsque l'indicateur Complete de la collection aura la valeur "true". Utilisez la méthode GetFilteredDefinitions avec les méthodes ReadAll ou IsMoreThan.

GetFilteredDiagrams

Finalité
Renvoie une collection filtrée de diagrammes contenus dans une encyclopédie.

Paramètres
Type de données : objets SA

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`WildCardName`

Utilisation : obligatoire

Type de données : chaîne

Critères de filtrage (par exemple "C*" = tous les diagrammes débutant par "C")

Remarque : La recherche avec caractère générique distingue les majuscules des minuscules.

`SAType`

Utilisation : obligatoire

Type de données : long

Type du diagramme Rational System Architect qui est récupéré (par exemple GTCATPROCESSFLOW ou 89)

Remarque : Reportez-vous au fichier DIAGRAMS.BAS contenu dans le répertoire de Rational System Architect pour obtenir la liste complète des diagrammes de SA ainsi que leurs numéros et noms constants internes.

Exemple

L'expression suivante renvoie tous les diagrammes de Gane & Sarson commençant par "Pr".

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

La collection SAObjects ne sera complétée que lorsque l'indicateur Complete de la collection aura la valeur "true". Utilisez la méthode GetFilteredDiagrams avec les méthodes ReadAll ou IsMoreThan.

GetRelationMetric

Finalité

Renvoie des informations sur le mode opératoire ou les relations existant entre deux objets Rational System Architect contenus dans l'encyclopédie.

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`SAObject1`

Utilisation : obligatoire

Type de données : objet

Indique l'un des deux objets Rational System Architect requis pour mesurer la relation.

`SAObject2`

Utilisation : obligatoire

Type de données : objet

Indique l'un des deux objets Rational System Architect requis pour mesurer la relation.

`Relation`

Utilisation : obligatoire

Type de données : RELATETYPE

Type de la relation existante entre les deux objets SA indiqués dans les paramètres ci-dessus. Voir le chapitre 16 pour consulter la liste complète des types de relations de Rational System Architect ainsi que leurs descriptions.

`Depth`

Utilisation : obligatoire

Type de données : long

Nombre de relations existantes entre les deux objets SA indiqués dans les paramètres ci-dessus. Par exemple, si Object 1 est une structure de données qui contient Object 2 et qu'Object 2 est une donnée élémentaire, la profondeur (depth) entre les deux objets est égale à 1.

`Metric`

Utilisation : obligatoire

Type de données : RELATIONMETRIC

Mesure de relation. Voir ci-après pour obtenir la liste complète des mesures de relation disponibles.

`FieldType`

Utilisation : obligatoire

Type de données : FLDTYPE

Type de zone. Voir le chapitre 17 pour obtenir la liste exhaustive des types de zones de Rational System Architect.

`NbrChars`

Utilisation : facultatif

Type de données : long

Nombre de caractères renvoyés par SA qui apparaîtront avant le séparateur décimal.

`NbrDec`

Utilisation : facultatif

Type de données : long

Nombre de caractères renvoyés par SA qui apparaîtront après le séparateur décimal.

GetXML

Finalité

Exporte la chaîne XML de l'encyclopédie dans un fichier .xml valide.

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`StrXML`

Utilisation : obligatoire

Type de données : chaîne

Si `bToFile` a la valeur "True", ce paramètre indique un nom de fichier XML valide dans lequel SA va exporter la chaîne XML de l'encyclopédie. Si `bToFile` a la valeur "False", strXML est traité comme la chaîne XML.

`bToFile`

Utilisation : obligatoire

Type de données : booléen

Si ce paramètre a la valeur "True", la méthode crée le fichier nommé dans le paramètre strXML. Si sa valeur est "False", la méthode affecte au paramètre strXML la chaîne XML de l'encyclopédie.

Méthodes OpenLock et CloseUnlock

Finalité

Les méthodes OpenLock et CloseUnlock permettent de contrôler l'état du verrouillage de l'encyclopédie Rational System Architect active. Cet état détermine si l'encyclopédie est verrouillée pour l'accès en lecture seule, en lecture/écriture ou pour la mise à jour pendant les opérations de VBA.

Si la méthode OpenLock est exécutée dans un mode particulier, la méthode CloseUnlock doit également être exécutée dans le même mode dans la suite du code.

Les méthodes OpenLock et CloseUnlock peuvent être exécutées plusieurs fois dans un code si différents niveaux de verrouillage sont nécessaires dans l'encyclopédie.

Si les méthodes OpenLock et CloseUnlock ne sont pas exécutées dans le code VBA, Rational System Architect exécute son propre verrouillage à chaque utilisation d'une méthode du modèle d'objet. Notez que cela peut réduire les performances de la macro.

Les deux méthodes renvoient une valeur booléenne qui indique si l'appel a abouti ou non.

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`LockMode`

Utilisation : obligatoire

Type de données : EncyLockMode

Etat de verrouillage de l'encyclopédie Rational System Architect active.

EncyLockMode	Signification
NETOPENREAD	Lecture seule
NETOPENREADWRITE	Lecture/écriture
NETOPENUPDATE	Droit de mise à jour pendant l'exécution des applications VBA.

Exemple

```
Dim sa As Application
Set sa = New Application
sa.Encyclopedia.OpenLock NETOPENREAD
    ' exécution du code SA à cet endroit
sa.Encyclopedia.CloseUnLock NETOPENREAD
Set sa = Nothing
```

SetXML

Finalité

Importe un fichier XML dans l'encyclopédie.

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`StrXML`

Utilisation : obligatoire

Type de données : chaîne

Si la valeur de bFromFile est "True", ce paramètre indique un nom de fichier XML valide depuis lequel SA va importer le code XML. Si cette valeur est "False", il s'agit de la chaîne XML de l'encyclopédie.

`bFromFile`

Utilisation : obligatoire

Type de données : booléen

Si ce paramètre a la valeur "True", la méthode importe le fichier nommé dans le paramètre strXML. Si sa valeur est "False", la méthode affecte au paramètre strXML la chaîne XML de l'encyclopédie.

`bValidate`

Utilisation : obligatoire

Type de données : booléen

Si ce paramètre a la valeur "True", la chaîne XML est validée par l'analyseur syntaxique.

SetXMLEx

Finalité

Syntaxe

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

`Encyclopedia Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Encyclopedia instanciée

`StrXML`

Utilisation : obligatoire

Type de données : chaîne

Si la valeur de bFromFile est "True", ce paramètre indique un nom de fichier XML valide depuis lequel SA va importer le code XML. Si cette valeur est "False", il s'agit de la chaîne XML de l'encyclopédie.

`ICollision`

Utilisation : obligatoire

Type de données : long

Options de collision	Description
0	Ne jamais remplacer une définition ou un diagramme existant
1	S'il existe une définition, extraire ses propriétés, la supprimer, la recréer puis redéfinir ses propriétés.
2	Mettre à jour les zones uniques quand des données sont fournies
3	Mettre à jour les zones uniques, effacer la zone en l'absence de données
256	Toujours remplacer le diagramme existant

`bFromFile`

Utilisation : obligatoire

Type de données : booléen

Si ce paramètre a la valeur "True", la méthode importe le fichier nommé dans le paramètre strXML. Si sa valeur est "False", la méthode affecte au paramètre strXML la chaîne XML de l'encyclopédie.

`bValidate`

Utilisation : obligatoire

Type de données : booléen

Si ce paramètre a la valeur "True", la chaîne XML est validée par l'analyseur syntaxique.

Méthodes de l'espace de travail

L'ajout d'espaces de travail dans Rational System Architect version 11.3 n'a pas d'effet sur les macros existantes car aucun objet d'espace de travail n'a été introduit dans le modèle d'objet. Rational System Architect permet d'accéder à un seul espace de travail à la fois. C'est pourquoi l'objet espace de travail se mappe presque complètement à l'objet encyclopédie. Les extensions suivantes ont été apportées au modèle d'objet pour faciliter la manipulation des espaces de travail.

IsEncyWorkspaceEnabled

Finalité
Renvoie "True" si l'encyclopédie active est compatible avec un espace de travail.

GetWorkspaceID

Finalité
Renvoie l'ID de l'espace de travail actif.

SetWorkspaceID

Finalité
Remplace l'espace de travail actif par un autre.

GetWorkspaceTree

Finalité
Renvoie une arborescence XML qui contient le nom des espaces de travail, leur ID et leur état de base.

GetWorkspaceName

Finalité
Renvoie le nom de l'espace de travail actif.

IsWorkspaceReadOnly

Finalité
Renvoie "True" si l'espace de travail actif est en lecture seule.

Appels de la classe Application de l'espace de travail

Les appels de la classe Application pour les espaces de travail sont répertoriés ci-après.

OpenEncyclopediaUsingConnectionStringAndWorkspace

Finalité
Version d'OpenEncyclopediaUsingConnectionString pour les espaces de travail.

OpenEncyclopediaUsingDisplayNameAndWorkspace

Finalité
Version d'OpenEncyclopediaUsingDisplayName pour les espaces de travail.

Evénements de la classe Application de l'espace de travail

Les événements de la classe Application pour les espaces de travail sont répertoriés ci-après.

WorkspaceOpen

Finalité
Evénement déclenché quand l'espace de travail change.

WorkspaceBeforeOpen

Finalité
Evénement déclenché avant le changement d'espace de travail pour permettre l'annulation. Notez que les références du modèle d'objet à des définitions ou des diagrammes sont invalidées comme pendant un changement d'encyclopédie.

Remarque : Ceci concerne exclusivement Rational System Architect version 11.3.0.2 et les versions ultérieures.

Mesures de relation

Une mesure de relation diffère d'une mesure de diagramme, de symbole ou de définition par le fait que des fonctions internes récupèrent des informations sur la relation existant entre deux objets Rational System Architect contenus dans l'encyclopédie. Pour chaque mesure de relation, vous devez déclarer les deux objets à examiner et la relation qui existe entre eux. Selon la mesure de relation utilisée, seuls certains objets spécifiques de Rational System Architect unis par des relations également spécifiques sont valides.

Pour accéder à ces mesures de relation vous devez appeler la méthode GetRelationMetric dans la classe Encyclopedia. L'Explorateur d'objets SA contient une liste énumérative nommée RELATIONMETRIC qui contient un répertoire de toutes les mesures de relations disponibles. Le tableau suivant répertorie toutes les mesures de relation disponibles avec leur description et leurs paramètres.

Mesure de relation	Description	Paramètres
RELMETCRUD	Renvoie la combinaison de caractères CRUD (Create, Read, Update, Delete) qui apparaît à côté du nom de processus dans le magasin de données sous la forme d'une chaîne.	SAObjects: Data Store, Process
RELMETDEPTH	Renvoie une valeur numérique qui indique le nombre de relations de type "utilise" existant entre deux objets.	SAObjects : doit posséder au moins une relation "utilise" ou "est utilisé par".
RELMETICOMROLE	Renvoie le rôle de relation (entrée, commande sortie, mécanisme ou limite) existant entre une flèche ICOM et le symbole de fonction/activité connecté.	SAObjects: ICOM Arrow, Function/Activity RelTypes: RELCONNSTART, RELSTARTAT, RELCONNEND, RELENDAT

RELMETINPUT	Vérifie si un symbole de flux "entre" dans un autre symbole ou diagramme. Renvoie une zone booléenne.	SAObjects : Symbole de flux, diagramme ou symbole de noeud RelType: RELCONNSTART, RELSTARTAT, RELCONNEND, RELENDAT, RELDIAGRAMCON, RELCONDIAGRAM
RELMETOUTPUT	Vérifie si un symbole de flux "sort" d'un autre symbole ou diagramme. Renvoie une zone booléenne.	SAObjects : Symbole de flux, diagramme ou symbole de noeud RelType: RELCONNSTART, RELSTARTAT, RELCONNEND, RELENDAT, RELDIAGRAMCON, RELCONDIAGRAM
RELMETSTATETABLE	Vérifie si le nom de l'événement attaché à une ligne de transition de sortie, connectée à un état, est mentionné dans la boîte de dialogue de définition de l'état. Si la valeur est "True", l'expression renvoie le nom de l'état.	SAObjects: Shlaer State, Shlaer Transition line RelType: RELCONNEND, RELCONNSTART

6

Classe Diagram

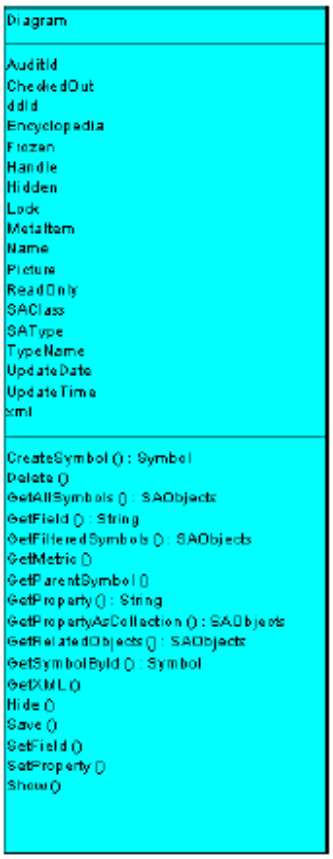
Rubriques contenues dans ce chapitre	Page

Introduction

Instance d'un diagramme contenu dans l'encyclopédie.

Pour renvoyer le diagramme actif sous la forme d'un objet, utilisez la syntaxe suivante :

```
Dim oApplication As
  SA2001.Application

Dim oDiagram As Diagram

Set oApplication = New
  Application

Set oDiagram =
  oApplication.oEncyclopedia.
  GetCurrentDiagram
```



Attributs

AuditID

Finalité
Tous les diagrammes enregistrés dans Rational System Architect sont assortis de l'identité de la personne qui les a créés ou les a modifiés en dernier. Cette identité est stockée dans chaque diagramme en tant qu'ID d'audit (AuditID).

Paramètres
Type de données : chaîne

Lecture seule

CheckedOut

Finalité
Si ce paramètre a la valeur True, le diagramme passe en lecture seule pour tous les utilisateurs sauf pour l'ID d'audit qui l'a réservé.

Paramètres
Type de données : booléen

ddID

Finalité
Tous les diagrammes enregistrés dans Rational System Architect sont identifiés de manière unique et interne au moyen d'un identifiant de dictionnaire de données. Cette méthode renvoie l'identité associée à un diagramme.

Paramètres
Type de données : long

Encyclopedia

Finalité
Facilite l'accès aux méthodes et aux attributs de l'encyclopédie parente.

Paramètres
Lecture seule

Frozen

Finalité
L'utilisateur doit disposer du droit de figer les objets pour définir cet attribut. Si ce paramètre a la valeur True, le diagramme passe en lecture seule pour tous les utilisateurs, y compris pour l'ID d'audit qui l'a figé.

Paramètre
Type de données : booléen

Descripteur

Finalité
Il s'agit du descripteur de mémoire du diagramme (disponible uniquement au moment de l'exécution). Ce descripteur n'est pas unique et est rarement le même.

Paramètres
Type de données : long

Lecture seule

Exemple

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

Hidden

Finalité
Renvoie la valeur “True” ou “False” pour indiquer respectivement si le diagramme est fermé ou ouvert.

Paramètres
Type de données : booléen

Lecture seule

Locked

Finalité
Renvoie la valeur “True” ou “False” pour indiquer respectivement si le diagramme est verrouillé ou non verrouillé (en cours d'utilisation).

Paramètres
Type de données : booléen

Lecture seule

MetaItem

Finalité
Facilite l'accès à la classe Méta-élément et à ses attributs.

Paramètres
Lecture seule

Nom

Finalité
Indique le nom de l'objet diagramme spécifié.

Paramètres
Type de données : chaîne

Lecture seule

Image

Finalité
Lorsqu'un diagramme est enregistré, Rational System Architect crée un métafichier Windows (.wmf) de ce diagramme et l'enregistre dans le répertoire de l'encyclopédie. Cet attribut permet d'accéder aux méthodes et aux attributs de stdPicture, un objet d'automatisation OLE qui contient des informations sur le contenu de l'image.

Paramètres
Type de données : stdPicture

Lecture seule

ReadOnly

Finalité
Renvoie True si l'objet a été ouvert en mode lecture seule.

Paramètres
Type de données : booléen

Lecture seule

SAClass

Finalité
Type de classe du diagramme. Cette propriété est également appelée le "numéro de type principal".

Paramètres
Type de données : long

Lecture seule

Remarque : La valeur renvoyée est toujours “1” pour un diagramme.

SAType

Finalité
Entier constant du diagramme. Tous les diagrammes enregistrés dans Rational System Architect possèdent un identifiant composé d'une valeur numérique constante et unique.

Paramètres
Type de données : long

Lecture seule

TypeName

Finalité
Type du diagramme exprimé sous la forme d'une chaîne, par exemple "relation d'entité".

Paramètres
Type de données : chaîne

Lecture seule

UpdateDate

Finalité
Date de la dernière modification du diagramme.

Paramètres
Type de données : chaîne

Lecture seule

UpdateTime

Finalité
Heure de la dernière modification du diagramme.

Paramètres
Type de données : chaîne

Lecture seule

xml

Finalité
Chaîne XML du diagramme. Ce paramètre est utilisé par la méthode GetXML.

Paramètres
Type de données : chaîne

Lecture seule

Méthodes

CreateSymbol

Finalité
Crée une instance de la classe Symbol avec un type et un nom spécifiques.

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée à laquelle le symbole sera ajouté.

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom du nouveau symbole

`SAType`

Utilisation : obligatoire

Type de données : long

Type du symbole Rational System Architect qui est créé (par exemple ETCATELEMBUSPROC ou 445)

Remarque : Reportez-vous au fichier SYMBOLS.BAS contenu dans le répertoire de Rational System Architect pour obtenir la liste complète des symboles de SA ainsi que leurs numéros et noms constants internes.

Remarque : Pour créer et conserver un symbole SA sur un diagramme, vous devez appeler la méthode Save de la classe Diagram, sans quoi le nouveau symbole sera supprimé du diagramme quand l'encyclopédie fermera.

Delete

Finalité
Supprime le diagramme désigné par l'objet diagramme spécifié.

GetAllSymbols

Finalité
Cette méthode renvoie tous les symboles contenus dans l'objet diagramme spécifié sous la forme d'une collection SAObjects. La collection SAObjects ne sera complétée que lorsque l'indicateur Complete de la collection aura la valeur "true". Utilisez la méthode GetAllSymbols avec les méthodes ReadAll ou IsMoreThan.

Règles
Vous devez définir la taille de la variable SAObjects et l'identifier comme une collection de symboles. Voir l'exemple ci-dessous.

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

GetField

Finalité
Caractéristiques du diagramme, par exemple la taille de la grille de symboles, la taille de la grille de lignes, le numéro de niveau, etc. Vous pouvez définir certaines de ces caractéristiques, par exemple le nom du diagramme, et d'autres non, par exemple le type de diagramme.

Syntaxe
```
Diagram Object. GetField FieldID
Diagram Object
```

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`FieldID`

Utilisation : obligatoire

Type de données : DGMFLD

Zone de diagramme. Voir ci-après pour obtenir la liste complète des zones de diagramme disponibles.

GetFilteredSymbols

Finalité

Pour filtrer les symboles contenus dans un diagramme, entrez vos critères de filtrage avec un caractère générique pour le premier argument puis spécifiez le type de symbole recherché comme deuxième argument. Vous pouvez aussi entrer la chaîne vide “”. L'expression renvoie une collection SAObjects. La collection SAObjects ne sera complétée que lorsque l'indicateur Complete de la collection aura la valeur "true". Utilisez la méthode GetFilteredSymbols avec les méthodes ReadAll ou IsMoreThan.

Paramètres

Type de données : objets SA

Syntaxe

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`WildCardName`

Utilisation : obligatoire

Type de données : chaîne

Critères de filtrage (par exemple "C*" = tous les symboles débutant par "C")

Remarque : La recherche avec caractère générique distingue les majuscules des minuscules.

`SAType`

Utilisation : obligatoire

Type de données : long

Type du symbole Rational System Architect qui est créé (par exemple ETCATELEMBUSPROC ou 445)

Remarque : Reportez-vous au fichier SYMBOLS.BAS contenu dans le répertoire de Rational System Architect pour obtenir la liste complète des symboles de SA ainsi que leurs numéros et noms constants internes.

Exemple

Cet exemple de syntaxe renvoie tous les symboles de type entité commençant par la lettre "P" qui sont présents dans le diagramme.

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

Get Metric

Finalité

Appelle des listes, des calculs et des fonctions internes associés aux diagrammes.

Syntaxe

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`Mesure`

Utilisation : obligatoire

Type de données : DIAGRAMMETRIC

Mesure de diagramme. Voir ci-après pour obtenir la liste complète des mesures de diagramme disponibles.

`FieldType`

Utilisation : facultatif

Type de données : FLDTYPE

Type de zone. Voir le chapitre 17 pour obtenir la liste exhaustive des types de zones de Rational System Architect.

`NbrChars`

Utilisation : facultatif

Type de données : long

Si vous avez saisi un type de zone, ce paramètre indique à SA le nombre de caractères qui doivent être renvoyés avant le séparateur décimal.

`NbrDec`

Utilisation : facultatif

Type de données : long

Si vous avez saisi un type de zone, ce paramètre indique à SA le nombre de caractères qui doivent être renvoyés après le séparateur décimal.

GetParentSymbol

Finalité

Un diagramme peut être l'enfant d'un symbole parent, par exemple dans un diagramme de flux de données. Cette méthode renvoie l'objet du symbole parent de l'objet diagramme spécifié.

GetProperty

Finalité

Renvoie le contenu d'une propriété donnée pour un diagramme indiqué.

Paramètres

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans un diagramme intitulé Elementary Business Process of a Process Chart,la propriété "Locations" a été renommée et son véritable nom est "Location Types". Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Location Types” alors que son libellé est “Locations” :

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

Syntaxe

```
Diagram Object.GetProperty Name
```

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

GetPropertyAsCollection

Finalité

Certaines propriétés définissent des relations avec d'autres propriétés. Par exemple, un diagramme intitulé Process Chart fait référence à un objet Process Thread via sa propriété “Process Thread”. Cette méthode renvoie une collection de définitions ou de diagrammes de type OneOf et ListOf. Voir le chapitre 14 pour plus d'informations sur les types de propriétés OneOf et ListOf.

Paramètres

Type de données : OfCollection

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans un diagramme intitulé Elementary Business Process of a Process Chart,la propriété "Locations" a été renommée et son véritable nom est "Location Types". Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Location Types” alors que que son libellé est “Locations” :

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

Syntaxe

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`PropName`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

Exemple

```
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")
Boucle
```

GetRelatedObjects

Finalité
Cette méthode renvoie une collection SAObjects d'objets apparentés à l'objet diagramme actif.

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`RelType`

Utilisation : obligatoire

Type de données : RELATETYPE

Relation SA. Pour obtenir la liste exhaustive des relations disponibles, voir le chapitre 16.

GetSymbolById

Finalité
Tous les diagrammes enregistrés dans Rational System Architect sont identifiés de manière unique et interne au moyen d'un identifiant de dictionnaire de données. Cette méthode renvoie un symbole sous la forme d'un objet à partir de son identité spécifiée.

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`Id`

Utilisation : obligatoire

Type de données : long

Tous les symboles enregistrés dans Rational System Architect sont identifiés de manière unique et interne au moyen d'un identifiant de dictionnaire de données.

Exemple

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

GetXML

Finalité

Exporte la chaîne XML du diagramme dans un fichier .xml valide.

Syntaxe

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`StrXMLTextOut`

Utilisation : obligatoire

Type de données : chaîne

Fichier XML valide dans lequel SA doit exporter la chaîne XML du diagramme.

Masquer

Finalité

Ferme une instance d'un diagramme actuellement ouvert.

Syntaxe

```
Call oDiagram.Hide
```

Save

Finalité

Sauvegarde une instance d'un diagramme.

Syntaxe

```
Call oDiagram.Save
```

SetField

Finalité

Permet de définir une zone de diagramme avec une valeur spécifiée.

Syntaxe

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`FieldID`

Utilisation : obligatoire

Type de données : DGMFLD

Zone de diagramme. Voir ci-après pour obtenir la liste complète des zones de diagramme disponibles.

`Valeur`

Utilisation : obligatoire

Type de données : chaîne

Valeur de la zone de diagramme

SetProperty

Finalité

Pour définir la valeur d'une propriété de diagramme, indiquez le nom de la propriété comme premier argument et sa valeur comme deuxième argument. Les noms des propriétés figurent dans les fichiers saprops.cfg et usrprops.txt.

Paramètres

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans un diagramme intitulé Elementary Business Process of a Process Chart,la propriété "Locations" a été renommée et son véritable nom est "Location Types". Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Location Types” alors que que son libellé est “Locations” :

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

Syntaxe

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

`Diagram Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Diagram instanciée

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

`Valeur`

Utilisation : obligatoire

Type de données : chaîne

Valeur de la propriété de diagramme

Show

Finalité
Cette méthode ouvre le diagramme dans un écran de Rational System Architect.

Syntaxe
```
Call oDiagram.Show
```

Diagram Fields

La propriété de la zone de diagramme peut contenir plusieurs propriétés relatives au diagramme. Ces propriétés contiennent généralement des données que l'utilisateur ne peut pas saisir directement mais qui sont automatiquement renseignées pendant l'utilisation habituelle. Le modèle d'objet contient un type énumératif appelé *DGMFLD*. Ce type est transmis comme paramètre dans les expressions **GetField(FieldID as *DGMFLD*)** et **SetField(FieldID as *DGMFLD*, Value as String)**. Ceci permet au programmeur VBA de lire et de mettre à jour des zones de bas niveau du diagramme.

DGMFLD constant	Description	Type de données
DIAGFLD_BBORDER	Active/désactive la case à cocher de bordure de rapport dans la fenêtre Mise en page. Cette fonction permet de placer un cadre autour du rapport.	“O” = désélectionné “1” = sélectionné
DIAGFLD_BDGMBORDER	Sélectionne une bordure dans la liste déroulante des types de cadres du diagramme, dans la fenêtre Mise en Page.	“0” = pas de mise en forme, pas de cadre “1” = cadre simple
DIAGFLD_BDGMPDEFAULT	Définit les paramètres de la fenêtre Mise en page comme paramètres par défaut.	“0” = Non “1” = Oui
DIAGFLD_BORDEROFFSET	Définit la valeur de retrait du cadre dans la fenêtre Mise en page. Cette zone définit la distance entre le cadre et le texte du rapport selon la valeur spécifiée.	Chaîne Valeur numérique exprimée en centièmes de pouce.
DIAGFLD_BPRESENTATIONMENU	Active/désactive l'ajout automatique du menu Présentation dans la boîte à outils de dessin. Les autres symboles inclus dans le	“0” = Non “1” = Oui

DGMFLD constant	Description	Type de données
	menu Présentation comprennent des représentations d'un ordinateur, d'un téléphone, d'une personne, d'un disque et d'une imprimante. Ces symboles s'insèrent dans le diagramme de la même manière que les autres symboles et peuvent être nommés de manière appropriée.	
DIAGFLD_BREADONLY	Applique à l'objet diagramme le mode lecture seule.	“0” = Non “1” = Oui
DIAGFLD_BSHOWGRID	Active/désactive l'affichage automatique de la grille sous-jacente.	“0” = Non “1” = Oui
DIAGFLD_BSHOWLINESHADOW	Active/désactive l'insertion automatique d'une ombre autour de tous les symboles de ligne.	“0” = Non “1” = Oui
DIAGFLD_BSHOWNODESHADOW	Active/désactive l'insertion automatique d'une ombre autour de tous les symboles de noeud.	“0” = Non “1” = Oui
DIAGFLD_BSHOWPAGES	Active/désactive la case à cocher Pages dans la fenêtre Options d'affichage du diagramme. Si vous cochez cette case, les zones d'impression s'afficheront sous la forme de traits pointillés pour que vous puissiez prévisualiser les limites de la page si vous imprimez le diagramme dans	“O” = désélectionné “1” = sélectionné

DGMFLD constant	Description	Type de données
	le mode Taille réelle.	
DIAGFLD_BSHOWRULER	Active/désactive l'affichage des repères de l'axe des x et de l'axe des y de la règle mesurés en centimètres ou en pouces selon les paramètres locaux de l'ordinateur. Les axes des x et des y sont sauvegardés avec le dessin quand cette option est activée.	“0” = Non “1” = Oui
DIAGFLD_BSHOWSCROLL	Active/désactive l'affichage automatique des barres de défilement qui permettent à l'utilisateur de se déplacer dans un diagramme s'il dépasse les dimensions d'un seul écran dans le mode d'affichage actif. Cette option est sélectionnée par défaut.	“O” = désélectionné “1” = sélectionné
DIAGFLD_BSHOWTEXTSHADOW	Active/désactive l'insertion automatique d'une ombre autour de tous les symboles de texte.	“0” = Non “1” = Oui
DIAGFLD_BSNAPGRIDENT	Accroche les symboles de noeud à la grille la plus proche (normalement invisible) du diagramme une fois que vous avez modifié le paramètre de grille afin de définir une grille plus épaisse ou plus fine.	“0” = Non “1” = Oui
DIAGFLD_BSNAPGRIDLIN	Accroche les symboles de ligne à la grille la plus proche (normalement invisible) du diagramme une fois que vous	“0” = Non “1” = Oui

DGMFLD constant	Description	Type de données
	avez modifié le paramètre de grille afin de définir une grille plus épaisse ou plus fine.	
DIAGFLD_CGRAPHNAME	Nom du diagramme.	Chaîne
DIAGFLD_CLEVELNUMBER	Numéro de niveau du diagramme	Lecture seule Chaîne
DIAGFLD_DDDIAGRAM_DDIDENTITY	Numéro d'ID du dictionnaire de données du diagramme	Lecture seule Numérique
DIAGFLD_IDGMFORM	Sélectionne la forme de bordure dans la liste déroulante des types de cadres du diagramme, dans la fenêtre Mise en Page.	“0” = pas de mise en forme, pas de cadre “1” = travail IDEF0 “2” = publication IDEF0 “3” = IDEF3 “4” = édition IDEF3 “5” = formulaire SSADM
DIAGFLD_IGRAPHTYPE	Type de diagramme dans SA	Chaîne Numéro constant interne du diagramme. La liste complète de ces numéros se trouve dans le fichier diagrm.bas dans le répertoire de SA.
DIAGFLD_PGRIDNUMENT	Nombre de points de grille du symbole de noeud par pouce	“[Vertical] [Horizontal]” **Remarque :** Cette méthode s'utilise avec la méthode DIAGFLD_PGRIDSIZEENT
DIAGFLD_PGRIDNUMLIN	Nombre de points de grille du	“[Vertical] [Horizontal]”

DGMFLD constant	Description	Type de données
	symbole de ligne par pouce	**Remarque :** Cette méthode s'utilise avec la méthode DIAGFLD_PGRIDSIZELIN
DIAGFLD_PGRIDSIZEENT	Nombre de pouces par point de grille pour le symbole de noeud	“[Vertical] [Horizontal]” en centièmes de pouce **Remarque :** Cette méthode s'utilise avec la méthode DIAGFLD_PGRIDNUMENT
DIAGFLD_PGRIDSIZELIN	Nombre de pouces par point de grille pour le symbole de ligne	“[Vertical] [Horizontal]” en centièmes de pouce **Remarque :** Cette méthode s'utilise avec la méthode DIAGFLD_PGRIDNUMLIN
DIAGFLD_PGRIDUNIT100	Définit la distance sur laquelle vous pouvez faire glisser un objet dans la grille. La valeur par défaut est “100 100”.	“[Vertical] [Horizontal]” en centièmes de pouce
DIAGFLD_PSHADOWDELTA	Définit la distance à laquelle l'ombre est placée par rapport au symbole. La valeur par défaut est “20 10”	“[Vertical] [Horizontal]” en centièmes de pouce
DIAGFLD_RGBSHADOWCOLOR	Définit la couleur de l'ombre.	“[couleur RVB]”
DIAGFLD_RMARGIN	Définit les marges dans la fenêtre Mise en page.	“[Left] [Top] [Right] [Bottom]” en centièmes de pouce
DIAGFLD_SAAUDITID	ID d'audit du diagramme	Lecture seule Chaîne
DIAGFLD_SAIDENTITY	Numéro d'ID du dictionnaire de données du diagramme	Lecture seule Numérique

DGMFLD constant	Description	Type de données
DIAGFLD_SALOCK	Verrouille le diagramme.	“0” = déverrouillé “1” = verrouillé
DIAGFLD_SAMAJORTYPE	Type principal (diagramme)	Lecture seule Chaîne
DIAGFLD_SAMAJORTYPENUMBER	Numéro de type principal (1)	Lecture seule Numérique
DIAGFLD_SANAME	Nom du diagramme	Lecture seule Chaîne
DIAGFLD_SANUMBER	Numéro de niveau du diagramme (IDEF0 uniquement)	Lecture seule Numérique
DIAGFLD_SATYPE	Type de diagramme (par exemple graphique des processus, relation entre entités, etc.).	Lecture seule Chaîne
DIAGFLD_SATYPENUMBER	Numéro constant interne du diagramme.	Lecture seule Numérique
DIAGFLD_SAUPDATEDATE	Date de la dernière mise à jour	Lecture seule Zone de date
DIAGFLD_SAUPDATETIME	Heure de la dernière mise à jour	Lecture seule Zone horaire
DIAGFLD_USEDENTCOUNT	Nombre de symboles contenus dans le diagramme	Long (hexadécimal Lecture seule
DIAGFLD_WBORDERPENSTYLE	Style de trait du cadre défini dans la fenêtre Mise en page.	“[numéro SymPenStyle]” Pour obtenir la liste

DGMFLD constant	Description	Type de données
		exhaustive des styles de trait disponibles dans SA, voir le chapitre 7.
DIAGFLD_WORIENTATION	Orientation définie pour l'impression du diagramme dans la fenêtre Mise en page.	“0” = Valeur par défaut de l'imprimante “1” = Portrait “2” = Paysage “3” = Ajustement automatique

Mesures de diagramme

Auparavant, les mesures servaient à créer des listes, à exécuter des vérifications de règle et à fournir des calculs pour divers rapports de Rational System Architect. A présent, vous pouvez exécuter des mesures individuellement au moyen de la méthode GetMetric dans la classe Diagram. L'Explorateur d'objets de SA contient une liste énumérative nommée DIAGRAMMETRIC qui contient un répertoire de toutes les mesures de diagramme disponibles. Le tableau suivant répertorie toutes les mesures de diagramme disponibles avec leur description et leurs paramètres.

Mesure de diagramme	Description
DIAGMETBALANCE	Compare les lignes d'entrée et de sortie du diagramme avec les lignes d'entrée et de sortie de son processus parent. Crée une liste des lignes d'entrée et de sortie non concordantes en indiquant le type et le nom des symboles concernés. (Voir le fichier d'aide de la fonction Equilibrer le parent)
DIAGMETCHARCOUNT	Renvoie le nombre de caractère contenus dans la propriété de description du diagramme.
DIAGMETCURRENT	Zone booléenne (qui contient True ou False). Renvoie "True" si le diagramme est actuellement affiché.
DIAGMETELELMENTLIST	Crée une liste des éléments de bas niveau pour toutes les définitions de symbole du diagramme. Un élément est de bas niveau s'il n'est pas étendu par des relations.
DIAGMETINPUTLIST	Crée une liste des éléments de bas niveau utilisés comme entrée pour toutes les définitions de symbole du diagramme. Un élément est de bas niveau s'il n'est pas étendu par des relations.
DIAGMETLEVELNUMBER	Renvoie un nombre qui reflète la position hiérarchique du diagramme (par exemple 5.3, 5.3.1, etc.) sous la forme d'une chaîne.
DIAGMETLEVELNUMBERSORT	Renvoie un nombre qui reflète la position

Mesure de diagramme	Description
	hiérarchique du diagramme sous la forme d'une chaîne. Chaque nombre contient trois chiffres (par exemple 003.005.002). Ceci permet un meilleur tri des résultats.
DIAGMETLINECOUNT	Renvoie le nombre de lignes contenues dans la propriété de description du diagramme.
DIAGMETOUTPUTLIST	Crée une liste des éléments de bas niveau utilisés comme sortie pour toutes les définitions de symbole du diagramme. Un élément est de bas niveau s'il n'est pas étendu par des relations.
DIAGMETREFERENCE	Renvoie True si le diagramme est référencé par un autre objet.
DIAGMETRULES	Exécute des vérifications de règle qui recherchent les violations des règles méthodologiques standard dans le diagramme.
DIAGMETSELECTED	Renvoie True si le diagramme est actuellement ouvert et que des symboles y sont sélectionnés.
DIAGMETTOP	Renvoie False si le diagramme est un enfant d'un symbole. Renvoie True si le diagramme ne provient pas d'un symbole.
DIAGMETUNMARKEDLIST	Crée une liste des éléments de bas niveau utilisés par les définitions de tous les symboles de ligne du diagramme qui ne sont marqués ni comme entrée ni comme sortie (pas de pointe de flèche). Un élément est de bas niveau s'il n'est pas étendu par des relations.
DIAGMETWORDCOUNT	Renvoie le nombre de mots contenus dans la propriété de description du diagramme.

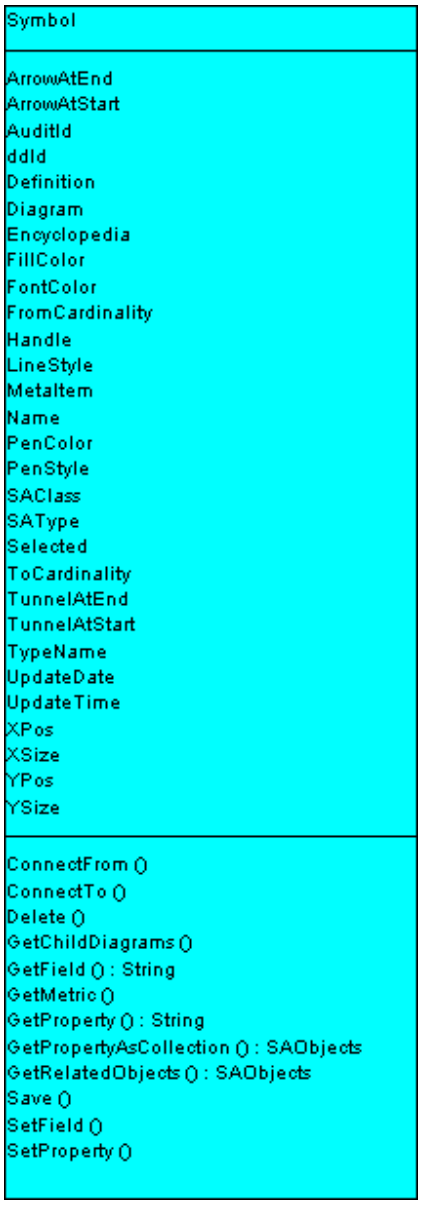
7

Classe Symbol

Rubriques contenues dans ce chapitre	Page

Introduction

Ce chapitre décrit la classe Symbol avec ses attributs et ses méthodes (représentés à droite).



Attributs

ArrowAtEnd

Finalité
Cet attribut définit les propriétés associatives d'un symbole de ligne. Il crée une pointe de flèche à la fin d'une ligne.

Paramètres
Type de données : booléen

Exemple

```
oSymbol.ArrowAtEnd = True
```

ArrowAtStart

Finalité
Cet attribut définit les propriétés associatives d'un symbole de ligne. Il crée une pointe de flèche au début d'une ligne.

Paramètres
Type de données : booléen

Exemple

```
oSymbol.ArrowAtStart = True
```

AuditId

Finalité
Tous les symboles enregistrés dans Rational System Architect sont assortis de l'identité de la personne qui les a créés ou les a modifiés en dernier. Cette identité est stockée dans chaque symbole en tant qu'ID d'audit (AuditID).

Paramètres
Type de données : chaîne

Lecture seule

ddId

Finalité
Tous les symboles enregistrés dans Rational System Architect sont identifiés de manière unique et interne au moyen d'un identifiant de dictionnaire de données. Cette méthode renvoie l'identité associée à un symbole.

Paramètres
Type de données : long

Lecture seule

Definition

Finalité
Facilite l'accès à la classe Definition du symbole.

Paramètres
Lecture seule

Diagram

Finalité
Facilite l'accès à la classe Diagram dans laquelle le symbole a été créé.

Paramètres
Lecture seule

Encyclopedia

Finalité
Facilite l'accès à la classe Encyclopedia à laquelle le symbole appartient.

Paramètres
Lecture seule

FillColor

Finalité
Cette propriété renvoie la couleur de remplissage du symbole. La valeur de la couleur est une valeur OLE_COLOR.

Paramètres
Une valeur OLE_COLOR est une valeur RVB (rouge, vert, bleu). Pour définir une valeur RVB, entrez les valeurs du bleu, du vert et du rouge avec un nombre compris entre 0 et 255 en utilisant la formule suivante :

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

FontColor

Finalité
Cette propriété renvoie la couleur de remplissage de la police du symbole. La valeur de la couleur est une valeur OLE_COLOR.

Paramètres
Une valeur OLE_COLOR est une valeur RVB (rouge, vert, bleu). Pour définir une valeur RVB, entrez les valeurs du bleu, du vert et du rouge avec un nombre compris entre 0 et 255 en utilisant la formule suivante :

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

FromCardinality

Finalité
Cette propriété sert à définir la cardinalité au début des lignes de relation d'un diagramme de relation entre entités ou de contraintes d'un modèle de données physique.

Les constantes définies ou renvoyées sont les suivantes :

Constante	Numéro	Signification
CARDINALITYZERO	0	Zéro
CARDINALITYONLYONE	1	Une seule
CARDINALITYZEROONE	2	Zéro ou plus
CARDINALITYONEMULT	3	Une ou plusieurs
CARDINALITYZEROONEMULT	4	Zéro, une ou plusieurs
CARDINALITYMULT	5	Plusieurs
CARDINALITYUNKNOWN	6	Non marquée
CARDINALITYNOTUSED	7	Aucune cardinalité

Descripteur

Finalité
Il s'agit du descripteur de mémoire du symbole (disponible uniquement au moment de l'exécution). Ce descripteur n'est pas unique et est rarement le même.

Paramètres
Type de données : long

Lecture seule

Exemple

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

LineStyle

Finalité

Vous pouvez choisir le style des lignes tracées sur un diagramme parmi plusieurs possibilités.

Les constantes habituellement définies ou renvoyées sont les suivantes :

Constante	Numéro	Signification
LSARC	4	Arc elliptique
LSAUTOSTROR	19	Droite orthogonale automatique
LSTRAA	1	Ligne droite, orientation quelconque
LSTROR	3	Droite orthogonale, non automatique

MetaItem

Finalité

Facilite l'accès aux attributs du méta-élément.

Paramètres

Lecture seule

Nom

Finalité
Nom du symbole

Paramètres
Type de données : chaîne

Lecture seule

PenColor

Finalité
Cette propriété définit ou renvoie la couleur du stylo du symbole. La valeur de la couleur est une valeur OLE_COLOR.

Paramètres
Une valeur OLE_COLOR est une valeur RVB (rouge, vert, bleu). Pour définir une valeur RVB, entrez les valeurs du bleu, du vert et du rouge avec un nombre compris entre 0 et 255 en utilisant la formule suivante :

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

PenStyle

Finalité
Cette propriété définit ou renvoie le style du stylo du symbole.

Il existe plusieurs constantes ayant le préfixe PEN. Elles correspondent à l'option Format… Symbole Style… Stylo dans Rational System Architect.

Constante	Numéro	Signification
PENDASH	1	
PENDASH2DOT	4	

PENDASHDOT	3	
PENDOT	2	
PENNULL	5	Aucun style de stylo.
PENSOLID1	16	
PENSOLID2	48	
PENSOLID3	64	
PENSOLID4	128	
PENSOLID4A	384	

SAClass

Finalité
Type de classe du symbole. Cette propriété est également appelée le "numéro de type principal".

Paramètres
Type de données : long

Lecture seule

Remarque : La valeur renvoyée est toujours “2” pour un symbole.

SAType

Finalité
Constante numérique du symbole.

Paramètres
Type de données : long

Lecture seule

Selected

Finalité
Indique si le symbole est ou non mis en évidence dans le diagramme.

Paramètres
Type de données : booléen

ToCardinality

Finalité
Cette propriété sert à définir la cardinalité à la fin des lignes de relation d'un diagramme de relation entre entités ou de contraintes d'un modèle de données physique.

Les constantes définies ou renvoyées sont les suivantes :

Constante	Numéro	Signification
CARDINALITYMULT	0	Plusieurs
CARDINALITYNOTUSED	1	Aucune cardinalité
CARDINALITYONEMULT	2	Une ou plusieurs
CARDINALITYONLYONE	3	Une seule
CARDINALITYUNKNOWN	4	Non marquée
CARDINALITYZERO	5	Zéro
CARDINALITYZEROONE	6	Zéro ou plus
CARDINALITYZEROONEMULT	7	Zéro, une ou plusieurs

TunnelAtEnd

Finalité
Cette propriété sert à définir les tunnels à la fin des flèches des symboles de ligne des diagrammes de fonction IDF0.

Paramètres
Type de données : booléen

TunnelAtStart

Finalité
Cette propriété sert à définir les tunnels au début des flèches des symboles de ligne des diagrammes de fonction IDF0.

Paramètres
Type de données : booléen

TypeName

Finalité
Type du symbole exprimé sous la forme d'une chaîne, par exemple "Entité".

Paramètres
Type de données : chaîne

Lecture seule

UpdateDate

Finalité
Date de la dernière modification du symbole.

Paramètres
Type de données : chaîne

Lecture seule

UpdateTime

Finalité
Heure de la dernière modification du symbole.

Paramètres
Type de données : chaîne

Lecture seule

Xpos

Finalité
Coordonnée horizontale (X) du symbole exprimée en centièmes de pouce.

Il s'agit de l'emplacement situé à l'angle inférieur droit du symbole calculé à partir du côté gauche du diagramme.

Paramètres
Type de données : long

Xsize

Finalité
Largeur (axe des X) du symbole. Valeur numérique exprimée en centièmes de pouce.

Paramètres
Type de données : long

Ypos

Finalité
Coordonnée verticale (Y) du symbole exprimée en centièmes de pouce.

Il s'agit de l'emplacement situé à l'angle inférieur droit du symbole calculé à partir du côté supérieur du diagramme.

Paramètres
Type de données : long

Ysize

Finalité
Hauteur (axe des Y) du symbole. Valeur numérique exprimée en centièmes de pouce.

Paramètres
Type de données : long

Méthodes

ConnectFrom

Finalité
Permet de connecter un symbole au début de la ligne.

Syntaxe
```
Symbol Object.ConnectFrom Line
```

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée à partir de laquelle le symbole de ligne sera connecté.

`Ligne`

Utilisation : obligatoire

Type de données : Symbole

Tout symbole de ligne instancié

Exemple
```
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

Finalité
Permet de connecter un symbole à la fin de la ligne.

Syntaxe
```
Symbol Object.ConnectTo Line
```

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée à laquelle le symbole de ligne sera connecté.

`Ligne`

Utilisation : obligatoire

Type de données : Symbole

Tout symbole de ligne instancié

Exemple
```
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

Finalité
Supprime le symbole dans le diagramme. Quand un diagramme est fermé, tous les symboles qui s'y trouvent sont automatiquement supprimés excepté si l'objet diagramme est sauvegardé.

GetChildDiagrams

Finalité
Cette méthode récupère la collection SAObjects des diagrammes d'un symbole auquel des enfants sont connectés.

La collection SAObjects ne sera complétée que lorsque l'indicateur Complete de la collection aura la valeur "true". Utilisez la méthode GetAllSymbols avec les méthodes ReadAll ou IsMoreThan.

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

GetField

Finalité
Caractéristiques du symbole, par exemple le type de police, la hauteur de police, etc. Vous pouvez définir certaines de ces caractéristiques, par exemple le type de police, et d'autres non, par exemple l'ID d'audit.

Syntaxe
```
Symbol Object. GetField FieldID
```

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée

`FieldID`

Utilisation : obligatoire

Type de données : SYMBOLFIELDS

Zone de symbole. Voir ci-après pour obtenir la liste complète des zones de symbole disponibles.

Get Metric

Finalité

Appelle des listes, des calculs et des fonctions internes associés aux symboles.

Syntaxe

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

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée

`Mesure`

Utilisation : obligatoire

Type de données : SYMBOLMETRIC

Mesure de symbole. Voir ci-après pour obtenir la liste complète des mesures de symbole disponibles.

`FieldType`

Utilisation : facultatif

Type de données : FLDTYPE

Type de zone. Voir le chapitre 17 pour obtenir la liste exhaustive des types de zones de Rational System Architect.

`NbrChars`

Utilisation : facultatif

Type de données : long

Si vous avez saisi un type de zone, ce paramètre indique à SA le nombre de caractères qui doivent être renvoyés avant le séparateur décimal.

`NbrDec`

Utilisation : facultatif

Type de données : long

Si vous avez saisi un type de zone, ce paramètre indique à SA le nombre de caractères qui doivent être renvoyés après le séparateur décimal.

GetProperty

Finalité

Renvoie le contenu d'une propriété donnée d'un symbole défini. Voir les fichiers usrprops.txt et saprops.cfg pour obtenir la liste complète des noms de propriété disponibles.

Paramètres

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans un diagramme intitulé Junction of a IDEF3 Process Flow/ OV-6a, la propriété "Logic" a été renommée et s'appelle en réalité “Junction Logic”. Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Junction Logic” alors que son libellé est “Logic” :

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

Syntaxe

```
Symbol Object.GetProperty Name
```

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

GetPropertyAsCollection

Finalité

Certaines propriétés définissent des relations avec d'autres propriétés. Par exemple, une entité peut référencer son modèle via sa propriété Modèle. Cette méthode renvoie une collection de définitions ou de diagrammes de type OneOf et ListOf. Voir le chapitre 14 pour plus d'informations sur les types de propriétés OneOf et ListOf.

Paramètres

Type de données : OfCollection

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans un diagramme intitulé Junction of a IDEF3 Process Flow/ OV-6a, la propriété "Logic" a été renommée et s'appelle en réalité “Junction Logic”. Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Junction Logic” alors que son libellé est “Logic” :

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

Syntaxe

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

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée

`PropName`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

Exemple

```
Dim Models As OfCollection

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

GetRelatedObjects

Finalité

Cette méthode renvoie une collection SAObjects d'objets apparentés à l'objet symbole actif.

Syntaxe

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

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée

`RelType`

Utilisation : obligatoire

Type de données : RELATETYPE

Relation SA. Pour obtenir la liste exhaustive des relations disponibles, voir le chapitre 16.

Exemple

```
Dim oCollectionOfRelatedItems As SAObjects

Set oCollectionOfRelatedItems =
  oSymbol.GetRelatedObjects(RELCONNEND)

oCollectionOfRelatedItems.ReadAll
```

Save

Finalité
Pour sauvegarder un symbole dans un diagramme après sa création, appelez la méthode **save**.

Exemple
```
oSymbol.Save
```

SetField

Finalité
Cette propriété sert à définir les valeurs de zone d'un symbole. Elle demande deux arguments qui sont la zone et sa valeur.

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

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée

`FieldID`

Utilisation : obligatoire

Type de données : SYMBOLFIELD

Zone de symbole. Voir ci-après pour obtenir la liste complète des zones de symbole disponibles.

`value`

Utilisation : obligatoire

Type de données : chaîne

Valeur de la zone de symbole

SetProperty

Finalité

Si vous connaissez le nom d'une propriété de symbole et sa valeur, vous pouvez définir cette propriété. Voir les fichiers usrprops.txt et saprops.cfg pour obtenir la liste complète des noms de propriété disponibles.

Paramètres

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans un diagramme intitulé Junction of a IDEF3 Process Flow/ OV-6a, la propriété "Logic" a été renommée et s'appelle en réalité “Junction Logic”. Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Junction Logic” alors que son libellé est “Logic” :

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

Syntaxe

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

`Symbol Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Symbol instanciée

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

`Valeur`

Utilisation : obligatoire

Type de données : chaîne

Valeur de la propriété de symbole

Zones de symbole

La propriété de la zone de symbole peut contenir plusieurs propriétés relatives au symbole. Cette propriété contient généralement des données que l'utilisateur ne peut pas saisir directement mais qui sont automatiquement renseignées pendant l'utilisation habituelle. Exemples : Heure de mise à jour, Audit, Position, Taille, Nom de type.

Le modèle d'objet contient un type énumératif appelé *SYMBOLFIELD*. Ce type est transmis comme paramètre dans les expressions **GetField(FieldID as *SYMBOLFIELD*)** et **SetField(FieldID as *SYMBOLFIELD*, Value as String)**. Ceci permet au programmeur VBA de lire et de mettre à jour des zones de symbole de bas niveau.

SYMBOLFIELD constant	Description	Type de données	
SYMFLD_AUDITID	ID d'audit du symbole	Chaîne Lecture seule	
SYMFLD_BARRANGMENT	Organisation de l'arborescence des symboles	“0” = Organiser les enfants horizontalement “1” = Organiser les enfants verticalement “2” = Organiser les enfants sous forme de bloc	
SYMFLD_BOTHERSYMBOLOGY	Affiche la forme alternative du symbole (par exemple un stéréotype)	Booléen	
SYMFLD_CBGCOLOR	Couleur d'arrière-plan d'un symbole dans un diagramme d'écran de texte.	“0” = Noir “1” = Bleu “2” = Vert “3” = Cyan	“4” = Rouge “5” = Magenta “6” = Brun/ Jaune “7” = Blanc
SYMFLD_CFGCOLOR	Couleur d'avant-plan d'un symbole dans un diagramme	“0” = Noir	“4” = Rouge

SYMBOLFIELD constant	Description	Type de données	
	d'écran de texte.	“1” = Bleu “2” = Vert “3” = Cyan	“5” = Magenta “6” = Brun/ Jaune “7” = Blanc
SYMFLD_COCCOFFSET	Détermine l'espacement entre chaque itération des valeurs dans certaines zones de saisie d'un diagramme d'écran de texte.	Numérique	
SYMFLD_COCCURS	Autorise plusieurs itérations des valeurs dans certaines zones de saisie d'un diagramme d'écran de texte.	Numérique	
SYMFLD_COMMENT	Définit la propriété de commentaire d'image du symbole.	Chaîne	
SYMFLD_CPROMPT	Caractère Cobalt Prompt du symbole dans le diagramme d'écran de texte.	Chaîne de 1 bit	
SYMFLD_CUNCLECOUNT	Renvoie le nombre de symboles qui sont directement connectés au symbole parent par des lignes de flux.	Hexadécimal Lecture seule	
SYMFLD_DDCOMMENT	Numéro d'ID du dictionnaire de données du symbole de commentaire du symbole	Numérique Lecture seule	
SYMFLD_DDIDENTITY	Numéro d'ID du dictionnaire de données du symbole	Numérique Lecture seule	
SYMFLD_DESCLOC	Emplacement du	“XPos YPos”	

SYMBOLFIELD constant	Description	Type de données
	commentaire d'image du symbole.	Numérique
SYMFLD_DESCSIZE	Taille du commentaire d'image du symbole.	“XSize YSize” Numérique
SYMFLD_DWSTYLE	Indique les options que l'utilisateur a choisies pour un symbole d'écran graphique dans un diagramme d'écran graphique.	Hexadécimal
SYMFLD_ENDLOC	Emplacement de la fin du symbole de ligne.	“XPos YPos” Numérique
SYMFLD_ERROR1	Première erreur détectée par Rational System Architect.	Numéro d'erreur
SYMFLD_ERROR2	Deuxième erreur détectée par Rational System Architect.	Numéro d'erreur
SYMFLD_FONTFLAGS	Active/désactive les options Gras, Italique, Souligné ou Barré pour la police de caractères du symbole.	Hexadécimal “0x0002” = Gras “0x0005” = Italique “0x000B” = Souligné “0x0010” = Barré
SYMFLD_FONTHEIGHT	Taille de police du symbole	Hexadécimal
SYMFLD_FONTNAME	Nom de la police du symbole (par exemple Arial, Times New Roman, etc.)	Chaîne
SYMFLD_FREXARCCHAR	Insère un arc exclusif au début d'un symbole de ligne.	Booléen
SYMFLD_FROMCARDINALIT	Renvoie la valeur de l'attribut	Chaîne

SYMBOLFIELD constant	Description	Type de données
Y	FromCardinality (par exemple exactement un, un ou plusieurs, etc.). Pour obtenir la liste exhaustive des valeurs de FromCardinality, reportez-vous à l'attribut FromCardinality dans le chapitre consacré aux attributs de la classe Symbol.	Lecture seule
SYMFLD_FROMCARDNUMBER	Renvoie la valeur constante de l'attribut FromCardinality. Pour obtenir la liste exhaustive des valeurs de FromCardinality, reportez-vous à l'attribut FromCardinality dans le chapitre consacré aux attributs de la classe Symbol.	Numérique Lecture seule
SYMFLD_FROMCONNECTCOMPASSPOINT	Renvoie N, E, S ou W pour indiquer le point de départ d'une flèche ICOM connectée au début de la ligne de symbole.	Chaîne Lecture seule
SYMFLD_HASFROMARROW	Renvoie "true" si une flèche figure au début de la ligne.	Booléen Lecture seule
SYMFLD_HASFROMTUNNEL	Renvoie "true" si un tunnel est présent sur la flèche au début de la ligne.	Booléen Lecture seule
SYMFLD_HASTOARROW	Renvoie "true" si une flèche figure à la fin de la ligne.	Booléen Lecture seule
Renvoie "true" si un tunnel est présent sur la flèche à la fin de la ligne.	Renvoie "true" si un tunnel est présent sur la flèche à la fin de la ligne.	Booléen Lecture seule

SYMBOLFIELD constant	Description	Type de données
SYMFLD_LINESTYLE	Style de trait du symbole	Valeur hexadécimale de 4 octets
SYMFLD_LOC	Emplacement du symbole dans le diagramme	“XPos YPos” Numérique
SYMFLD_NAME	Nom du symbole	Chaîne
SYMFLD_NAMECRLF	Nom du symbole de retour chariot et de saut de ligne. Vous pouvez saisir jusqu'à cinq lignes de texte dans la zone du nom.	Chaîne
SYMFLD_NAMECRLF1	Si le nom doit apparaître comme une chaîne de texte continue (par exemple JimJaneTomLouRon), indique le nombre de caractères après lequel le texte commence une deuxième ligne (4).	Numérique
SYMFLD_NAMECRLF2	Si le nom doit apparaître comme une chaîne de texte continue (par exemple JimJaneTomLouRon), indique le nombre de caractères après lequel le texte commence une troisième ligne (8).	Numérique
SYMFLD_NAMECRLF3	Si le nom doit apparaître comme une chaîne de texte continue (par exemple JimJaneTomLouRon), indique le nombre de caractères après lequel le texte commence une quatrième ligne (11).	Numérique

SYMBOLFIELD constant	Description	Type de données
SYMFLD_NAMECRLF4	Si le nom doit apparaître comme une chaîne de texte continue (par exemple JimJaneTomLouRon), indique le nombre de caractères après lequel le texte commence une cinquième ligne (14).	Numérique
SYMFLD_NAMELOC	Emplacement de la zone du nom de symbole.	“XPos YPos” Numérique
SYMFLD_NAMESIZE	Taille de la zone du nom de symbole.	“XSize YSize” Numérique
SYMFLD_ORDER	Mise en ordre des membres d'associations	“O” = Non ordonné “1” = Ordonné “2” = Trié
SYMFLD_PENSTYLE	Largeur et style du stylo du symbole	Valeur hexadécimale à 4 octets
SYMFLD_ROTATION	Rotation du symbole d'indicateur dans le diagramme de structure	Les nombres de 0 à 31 font pivoter le symbole d'indicateur dans le sens des aiguilles d'une montre, par exemple : “0” = sud “8” = ouest “16” = nord “24” = est
SYMFLD_SAMAJORTYPE	Type principal (symbole)	Chaîne

SYMBOLFIELD constant	Description	Type de données
		Lecture seule
SYMFLD_SAMAJORTYPENUMBER	Type principal (2)	Numérique Lecture seule
SYMFLD_SEQNUM	Numéro d'entité figurant dans les symboles d'entité	Numérique
SYMFLD_SIZE	Taille du symbole	“XSize YSize” Numérique
SYMFLD_STARTLOC	Emplacement du début du symbole de ligne.	“XPos YPos” Numérique
SYMFLD_STYLEFLAGS	Active les couleurs pour le symbole	Hexadécimal “0x0001” = Couleur de stylo “0x0002” = Couleur de remplissage “0x0004” = Couleur de police
SYMFLD_SUPERSUB	Définit la valeur de la relation super/sous du symbole.	“0” = Aucune “1” = Super “2” = Sous
SYMFLD_TEXTFLAGS	Propriétés de texte du symbole	Hexadécimal
SYMFLD_TOCARDINALITY	Renvoie la valeur de l'attribut ToCardinality (par exemple exactement un, un ou plusieurs, etc.). Pour obtenir la liste exhaustive des valeurs de ToCardinality, reportez-vous à l'attribut ToCardinality dans le	Chaîne Lecture seule

SYMBOLFIELD constant	Description	Type de données
	chapitre consacré aux attributs de la classe Symbol.	
SYMFLD_TOCARDNUMBER	Renvoie la valeur constante de l'attribut ToCardinality. Pour obtenir la liste exhaustive des valeurs de ToCardinality, reportez-vous à l'attribut ToCardinality dans le chapitre consacré aux attributs de la classe Symbol.	Numérique Lecture seule
SYMFLD_TOCONNECTCOMPASSPOINT	Renvoie N, E, S ou W pour indiquer le point de départ d'une flèche ICOM connectée à la fin de la ligne de symbole.	Chaîne Lecture seule
SYMFLD_TOEXARCCHAR	Insère un arc exclusif à la fin d'un symbole de ligne.	Booléen
SYMFLD_TYPE	Type SA du symbole	Valeur constante interne du type SA
SYMFLD_TYPENAME	Nom du type SA du symbole (par exemple Entité, Flèche ICOM, etc.)	Chaîne Lecture seule
SYMFLD_U_S1_WPICTYPE	Type d'image (fichier graphique ajouté au diagramme)	Chaîne Lecture seule
SYMFLD_U_S1_ZPPICFILE	Nom de chemin du fichier utilisé pour afficher l'image	Chaîne Lecture seule
SYMFLD_UPDATEDATE	Date de la dernière mise à jour	Zone de date Lecture seule
SYMFLD_UPDATEDATEINTL	Date de la dernière mise à jour (format universel)	Zone de date

SYMBOLFIELD constant	Description	Type de données
		Lecture seule
SYMFLD_UPDATETIMEINTL	Heure de la dernière mise à jour (format universel)	Zone horaire Lecture seule
SYMFLD_XPENTITY	Numéro interne affecté au symbole	Numérique Lecture seule
SYMFLD_XPGROUP	Numéro interne affecté au symbole parent du symbole	Numérique Lecture seule
SYMFLD-XPLINK	Numéro interne du symbole auquel le symbole sélectionné est connecté (par exemple un référent lié à une unité de comportement dans un diagramme de flux de processus IDEF3)	Numérique Lecture seule
SYMFLD_XPSIBLING	Numéro interne de l'élément apparenté suivant	Numérique Lecture seule
SYMFLD_XPSUBORDINATE	Numéro interne affecté au premier symbole enfant	Numérique Lecture seule
SYMFLD_ZPDESC	Définit le commentaire d'image du symbole.	Chaîne
SYMFLD-ZPSSADMSTR	Inconnu	

Mesures de symbole

Auparavant, les mesures servaient à créer des listes, à exécuter des vérifications de règle et à fournir des calculs pour divers rapports de Rational System Architect. A présent, vous pouvez exécuter des mesures individuellement au moyen de la méthode GetMetric dans la classe Symbol. L'Explorateur d'objets de SA contient une liste énumérative nommée SYMBOLMETRIC qui contient un répertoire de toutes les mesures de symbole disponibles. Le tableau suivant répertorie toutes les mesures de symbole disponibles avec leur description.

Mesure de symbole	Description
SYMMETANNOTATION	Renvoie True si le symbole est un symbole d'annotation (bloc de document, zone de texte, rectangle, connecteur de page).
SYMMETBALANCE	Compare les lignes d'entrée et de sortie du symbole avec les lignes d'entrée et de sortie de son processus enfant. Crée une liste des lignes d'entrée et de sortie non concordantes en indiquant le type et le nom des symboles concernés. (Voir le fichier d'aide de la fonction Equilibrer l'enfant). Si le symbole est un symbole de magasin de données, de connecteur AND ou de connecteur XOR, cette méthode compare les structures et les éléments définis de ces symboles. Elle crée une liste des éléments indéfinis contenus dans le symbole et indique si les flux de données entrants et sortants contiennent des éléments définis. (Voir le fichier d'aide de la fonction Equilibrer horizontalement).
SYMMETBALANCEMSPEC	Equilibre la mini-spécification de la définition du symbole. Cet élément est utilisé pour les processus des diagrammes de flux de données et les modules des diagrammes de structure. Pour plus d'informations, reportez-vous au fichier d'aide de Rational System Architect et recherchez le mot clé "Minispec".
SYMMETBOTTOM	Zone de valeur booléenne dont la valeur est dérivée. La valeur est "true" pour un symbole non développé dans

Mesure de symbole	Description
	un diagramme.
SYMMETCHARCOUNT	Renvoie le nombre de caractère contenus dans la propriété de description du symbole.
SYMMETCONNECTOR	Renvoie "true" si le symbole est un symbole de connecteur (connecteur AND, connecteur XOR ou jonction de flèche ICOM).
SYMMETCURRENT	Zone booléenne (qui contient True ou False). Renvoie "True" si le symbole figure dans le diagramme actuellement affiché.
SYMMETELEMENTLIST	Crée une liste des éléments de bas niveau de la définition du symbole. Un élément est de bas niveau s'il n'est pas étendu par des relations.
SYMMETEXPRESSION	Crée une liste des expressions dont la syntaxe est incorrecte ou des données élémentaires ou structures de données utilisées par l'expression de la définition de symbole.
SYMMETICOMDEST	Renvoie le rôle de la destination de la flèche ICOM (entrée, commande, mécanisme ou limite) sous la forme d'une chaîne.
SYMMETICOMSOURCE	Renvoie le rôle de la source de la flèche ICOM (appel, sortie ou limite) sous la forme d'une chaîne.
SYMMETINPUTLIST	Crée une liste des éléments de bas niveau utilisés comme entrée de la définition du symbole. Un élément est de bas niveau s'il n'est pas étendu par des relations.
SYMMETISFOREIGNKEY	Renvoie "True" si la définition de symbole est une clé externe.
SYMMETKEYCOMPNBR	Renvoie le numéro de composant de la clé primaire d'un symbole. L'utilisateur peut alternativement voir le numéro de composant en affichant la liste d'attributs de la définition de symbole dans la vue détaillée de l'Explorateur (par exemple @1, @2, etc.).

Mesure de symbole	Description
SYMMETLEVELNUMBER	Renvoie un nombre qui reflète la position hiérarchique du symbole (par exemple 5.3, 5.3.1, etc.) sous la forme d'une chaîne.
SYMMETLEVELNUMBERSORT	Renvoie un nombre qui reflète la position hiérarchique du symbole sous la forme d'une chaîne. Chaque nombre contient trois chiffres (par exemple 003.005.002). Ceci permet un meilleur tri des résultats.
SYMMETLINECOUNT	Renvoie le nombre de lignes contenues dans la propriété de description d'une définition de symbole.
SYMMETNORMALIZE1	Exécute une vérification pour savoir si la définition de symbole est au format First Normal. Une entité est au format First Normal si elle ne contient pas de groupes récurrents.
SYMMETNORMALIZE23	Exécute une vérification pour savoir si la définition de symbole est au format Second Normal ou Third Normal. Une entité est au format Second Normal si elle est au format First Normal et que le fonctionnement de chaque attribut non-clé dépend entièrement de la clé primaire. Une entité est au format Third Normal si elle est au format Second Normal et que le fonctionnement de chaque attribut non-clé dépend exclusivement de la clé primaire.
SYMMETOUTPUTLIST	Crée une liste des éléments de bas niveau utilisés comme sortie de la définition du symbole. Un élément est de bas niveau s'il n'est pas étendu par des relations.
SYMMETPARENTSLASHDATA	Renvoie les données de barre oblique de la clé externe de la définition de symbole, qui contiennent des informations sur le lieu de saisie de l'attribut. Ces données sont également accessibles en affichant la liste d'attributs de la définition de symbole dans la vue détaillée de l'Explorateur. Les données de barre oblique apparaîtront comme dans l'exemple suivant : Row_Number / FKFROM “Stock_Location.Row_Number(stores)” /

Mesure de symbole	Description
SYMMETREFERENCE	Renvoie True si la définition de symbole est référencée par un autre objet.
SYMMETRULES	Exécute des vérifications de règle qui recherchent les violations des règles méthodologiques standard dans le symbole.
SYMMETSELECTED	Renvoie True si le symbole figure dans un diagramme actuellement ouvert.
SYMMETSEQINPARENTSLIST	Vérifie l'ensemble de propriétés de l'objet parent pour la liste des objets enfants. Renvoie le nombre d'occurrences de la définition de symbole dans la liste.
SYMMETTOP	Renvoie False si le diagramme dans lequel figure le symbole est un enfant d'autres symboles. Renvoie True si le diagramme ne provient pas d'un autre symbole.
SYMMETUNMARKEDLIST	Crée une liste des éléments de bas niveau utilisés par les définitions de tous les symboles de ligne qui ne sont marqués ni comme entrée ni comme sortie (pas de pointe de flèche). Un élément est de bas niveau s'il n'est pas étendu par des relations.
SYMMETUPDATEUSES	Met à jour la table des relations du dictionnaire de données (RELATN.DBF). Cette expression n'a pas de valeur de retour.
SYMMETWORDCOUNT	Renvoie le nombre de mots contenus dans la propriété de description de la définition de symbole.

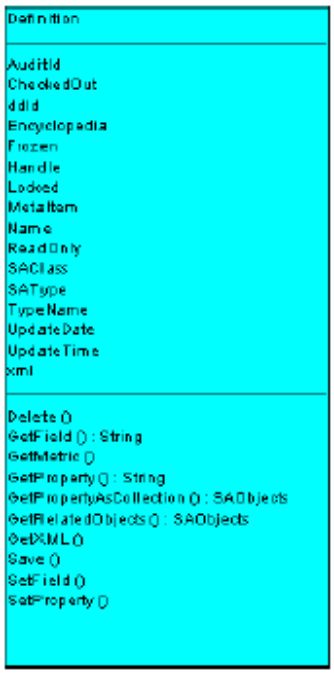
8

Classe Definition

Rubriques contenues dans ce chapitre	Page

Introduction

Ce chapitre décrit la classe Definition avec ses attributs et ses méthodes.



Attributs

AuditID

Finalité
Toutes les définitions enregistrées dans Rational System Architect sont assorties de l'identité de la personne qui les a créées ou les a modifiées en dernier. Cette identité est stockée dans chaque définition en tant qu'ID d'audit (AuditID).

Paramètres
Type de données : chaîne

Lecture seule

CheckedOut

Finalité
Si ce paramètre a la valeur True, la définition passe en lecture seule pour tous les utilisateurs sauf pour l'ID d'audit qui l'a réservée.

Paramètres
Type de données : booléen

ddID

Finalité
Toutes les définitions enregistrées dans Rational System Architect sont identifiées de manière unique et interne au moyen d'un identifiant de dictionnaire de données.

Paramètres
Type de données : long

Lecture seule

Encyclopedia

Finalité
Facilite l'accès aux méthodes et aux attributs de la classe Encyclopedia parente.

Paramètres
Lecture seule

Frozen

Finalité
L'utilisateur doit disposer du droit de figer les objets pour définir cet attribut. Si ce paramètre a la valeur True, la définition passe en lecture seule pour tous les utilisateurs, y compris pour l'ID d'audit qui l'a figée.

Paramètre
Type de données : booléen

Descripteur

Finalité
Il s'agit du descripteur de mémoire de la définition (disponible uniquement au moment de l'exécution). Ce descripteur n'est pas unique et est rarement le même.

Paramètres
Type de données : long

Lecture seule

Exemple

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

Locked

Finalité
Renvoie la valeur “True” ou “False” pour indiquer respectivement si la définition est verrouillée ou non verrouillée (en cours d'utilisation). Ceci requiert un objet définition valide.

Paramètres
Type de données : booléen

Lecture seule

MetaItem

Finalité
Facilite l'accès aux attributs du méta-élément.

Paramètres
Lecture seule

Nom

Finalité
Indique le nom de l'objet définition spécifié.

Paramètres
Type de données : chaîne

Lecture seule

ReadOnly

Finalité
Indique si la définition est en lecture seule ou non.

Paramètres
Type de données : booléen

Lecture seule

SAClass

Finalité
Type de classe de la définition. Cette propriété est également appelée le "numéro de type principal".

Paramètres
Type de données : long

Lecture seule

Remarque : La valeur renvoyée est toujours “3” pour une définition.

SAType

Finalité
Entier constant de la définition. Toutes les définitions enregistrées dans Rational System Architect possèdent un identifiant composé d'une valeur numérique constante et unique.

Paramètres
Type de données : long

Lecture seule

TypeName

Finalité
Type de la définition, exprimé sous la forme d'une chaîne, par exemple "processus".

Paramètres
Type de données : chaîne

Lecture seule

UpdateDate

Finalité
Date de la dernière modification de la définition.

Paramètres
Type de données : chaîne

Lecture seule

UpdateTime

Finalité
Heure de la dernière modification de la définition.

Paramètres
Type de données : chaîne

Lecture seule

xml

Finalité
Chaîne XML de la définition. Ce paramètre est utilisé par la méthode GetXML.

Paramètres
Type de données : chaîne

Lecture seule

Méthodes

Delete

Finalité
Supprime la définition désignée par l'objet définition spécifié.

Exemple
```
Call oDefinition.Delete
```

GetField

Finalité
Caractéristiques de la définition, par exemple le numéro de type, l'indicateur indéfini, le nom du type principal.

Syntaxe
```
Definition Object. GetField FieldID
```

`Definition Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Definition instanciée

`FieldID`

Utilisation : obligatoire

Type de données : DEFFLD

Zone de définition. Voir ci-après pour obtenir la liste complète des zones de définition disponibles.

Get Metric

Finalité

Appelle des listes, des calculs et des fonctions internes associés aux définitions.

Syntaxe

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

`Definition Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Definition instanciée

`Mesure`

Utilisation : obligatoire

Type de données : DEFINITIONMETRIC

Mesure de définition. Voir ci-après pour obtenir la liste complète des mesures de définition disponibles.

`FieldType`

Utilisation : facultatif

Type de données : FLDTYPE

Type de zone. Voir le chapitre 17 pour obtenir la liste exhaustive des types de zones de Rational System Architect.

`NbrChars`

Utilisation : facultatif

Type de données : long

Si vous avez saisi un type de zone, ce paramètre indique à SA le nombre de caractères qui doivent être renvoyés avant le séparateur décimal.

`NbrDec`

Utilisation : facultatif

Type de données : long

Si vous avez saisi un type de zone, ce paramètre indique à SA le nombre de caractères qui doivent être renvoyés après le séparateur décimal.

GetProperty

Finalité

Renvoie le contenu d'une propriété donnée pour une définition indiquée.

Paramètres

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans le diagramme intitulé Service Time Profile, la propriété "Time Units" a été renommée et s'appelle en réalité "Duration Time Units". Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Duration Time Units” alors que son libellé est “Time Units” :

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

Syntaxe

```
Definition Object.GetProperty Name
```

`Definition Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Definition instanciée

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

GetPropertyAsCollection

Finalité

Certaines propriétés définissent des relations avec d'autres propriétés. Par exemple, une définition d'entité référence les attributs via sa propriété Description. Cette méthode renvoie une collection de définitions ou de diagrammes de type OneOf et ListOf. Voir le chapitre 14 pour plus d'informations sur les types de propriétés OneOf et ListOf.

Paramètres

Type de données : OfCollection

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans le diagramme intitulé Service Time Profile, la propriété "Time Units" a été renommée et s'appelle en réalité "Duration Time Units". Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Duration Time Units” alors que son libellé est “Time Units” :

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

Syntaxe

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

`Definition Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Definition instanciée

`PropName`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

Exemple

```
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")

Boucle
```

GetRelatedObjects

Finalité

Cette méthode renvoie une collection SAObjects d'objets apparentés à l'objet définition actif.

Syntaxe

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

`Definition Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Definition instanciée

`RelType`

Utilisation : obligatoire

Type de données : RELATETYPE

Relation SA. Pour obtenir la liste exhaustive des relations disponibles, voir le chapitre 16.

GetXML

Finalité
Exporte la chaîne XML de la définition dans un fichier .xml valide.

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

`Definition Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Definition instanciée

`StrXMLTextOut`

Utilisation : obligatoire

Type de données : chaîne

Fichier XML valide dans lequel SA doit exporter la chaîne XML de la définition.

Save

Finalité
Sauvegarde une instance d'une définition.

Exemple
```
Call oDefinition.Save
```

SetField

Finalité
Permet de définir une zone de définition avec une valeur spécifiée.

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

`Definition Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Definition instanciée

`FieldID`

Utilisation : obligatoire

Type de données : DGMFLD

Zone de définition. Voir ci-après pour obtenir la liste complète des zones de définition disponibles.

`Valeur`

Utilisation : obligatoire

Type de données : chaîne

Valeur de la zone de définition

SetProperty

Finalité

Pour définir la valeur d'une propriété de définition, indiquez le nom de la propriété comme premier argument et sa valeur comme deuxième argument. Les noms des propriétés figurent dans les fichiers saprops.cfg et usrprops.txt.

Paramètres

Il arrive souvent que le véritable nom d'une propriété soit différent de celui qui apparaît dans une boîte de dialogue de définition. Par exemple, dans le diagramme intitulé Service Time Profile, la propriété "Time Units" a été renommée et s'appelle en réalité "Duration Time Units". Vous ne pouvez le savoir que si vous examinez la définition de ce diagramme dans le fichier saprops.cfg et que vous constatez que cette propriété s'appelle en réalité “Duration Time Units” alors que son libellé est “Time Units” :

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

Syntaxe

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

`Definition Object`

Utilisation : obligatoire

Type de données : objet

Toute classe Definition instanciée

`Nom`

Utilisation : obligatoire

Type de données : chaîne

Nom de la propriété tel qu'il figure dans le fichier saprops.cfg

`Valeur`

Utilisation : obligatoire

Type de données : chaîne

Valeur de la propriété de définition

Definition Fields

La propriété de la zone de définition peut contenir plusieurs propriétés relatives à la définition. Ces propriétés contiennent généralement des données que l'utilisateur ne peut pas saisir directement mais qui sont automatiquement renseignées pendant l'utilisation habituelle. Le modèle d'objet contient un type énumératif appelé *DEFFLD*. Ce type est transmis comme paramètre dans les expressions **GetField(FieldID as *DEFFLD*)** et **SetField(FieldID as *DEFFLD*, Value as String)**. Ceci permet au programmeur VBA de lire et de mettre à jour des zones de bas niveau de la définition.

DEFFLD constant	Description	Type de données
DEFNFLD_SAAUDITID	ID d'audit de la définition.	Lecture seule
DEFNFLD_SAIDENTITY	ID du dictionnaire de données de la définition.	Lecture seule
DEFNFLD_SAIDENTITY4	ID du dictionnaire de données de la définition.	Nombre binaire à 4 octets Lecture seule
DEFNFLD_SAISUNDEFINED	Renvoie "T" si la définition est indéfinie, renvoie "F" si elle est définie.	Lecture seule
DEFNFLD_SALOCK	Verrouille la définition.	“T” = verrouillé “F” = déverrouillé
DEFNFLD_SAMAJORTYPE	Type principal (définition)	Lecture seule
DEFNFLD_SAMAJORTYPENUMBER	Numéro de type principal (“3”)	Lecture seule
DEFNFLD_SANAME	Nom de la définition	Chaîne
DEFNFLD_SATYPE	Type SA de la définition (Classe UML, Entity, etc.)	Lecture seule
DEFNFLD_SATYPENUMBER	Numéro constant interne du type de définition. Pour obtenir une liste complète, reportez-vous au fichier DEFNS.BAS dans le	Lecture seule

DEFFLD constant	Description	Type de données
	répertoire de SA.	
DEFNFLD_SAUPDATEDATE	Date de la dernière mise à jour	Lecture seule
DEFNFLD_SAUPDATETIME	Heure de la dernière mise à jour	Lecture seule

Mesures de définition

Auparavant, les mesures servaient à créer des listes, à exécuter des vérifications de règle et à fournir des calculs pour divers rapports de Rational System Architect. A présent, vous pouvez exécuter des mesures individuellement au moyen de la méthode GetMetric dans la classe Definition. L'Explorateur d'objets SA contient une liste énumérative nommée DEFINITIONMETRIC qui contient un répertoire de toutes les mesures de définition disponibles. Le tableau suivant répertorie toutes les mesures de définition disponibles avec leur description.

Mesure de définition	Description
DEFMETBOTTOM	Zone de valeur booléenne dont la valeur est dérivée. La valeur est True pour les définitions qui ne sont pas des expressions et pour les expressions qui ne contiennent ni des données élémentaires ni des structures de données.
DEFMETCURRENT	Zone booléenne (qui contient True ou False). Renvoie True si le symbole de la définition figure dans un diagramme actuellement affiché.
DEFMETEXPRESSION	Crée une liste des expressions dont la syntaxe est incorrecte ou des données élémentaires ou structures de données utilisées par l'expression de la définition.
DEFMETISFOREIGNKEY	Renvoie "True" si la définition est une clé externe.
DEFMETKEYCOMPNBR	Renvoie le numéro de composant de la clé primaire d'une définition. L'utilisateur peut alternativement voir le numéro de composant en affichant la liste d'attributs de la définition dans la vue détaillée de l'Explorateur (par exemple @1, @2, etc.).
DEFMETNORMALIZE1	Exécute une vérification pour savoir si la définition est au format First Normal. Une entité est au format First Normal si elle ne contient pas de groupes récurrents.
DEFMETNORMALIZE23	Exécute une vérification pour savoir si la définition

Mesure de définition	Description
	est au format Second Normal ou Third Normal. Une entité est au format Second Normal si elle est au format First Normal et que le fonctionnement de chaque attribut non-clé dépend entièrement de la clé primaire. Une entité est au format Third Normal si elle est au format Second Normal et que le fonctionnement de chaque attribut non-clé dépend exclusivement de la clé primaire.
DEFMETPARENTSLASHDATA	Renvoie les données de barre oblique de la clé externe de la définition, qui contiennent des informations sur le lieu de saisie de l'attribut. Ces données sont également accessibles en affichant la liste d'attributs de la définition dans la vue détaillée de l'Explorateur. Les données de barre oblique apparaîtront comme dans l'exemple suivant : Row_Number / FKFROM “Stock_Location.Row_Number(stores)” /
DEFMETREFERENCE	Renvoie True si la définition est référencée par un autre objet.
DEFMETSELECTED	Renvoie True si un symbole défini par la définition est sélectionné.
DEFMETSEQINPARENTSLIST	Vérifie l'ensemble de propriétés de l'objet parent pour la liste des objets enfants. Renvoie le nombre d'occurrences de la définition dans la liste.
DEFMETSYNCHRONIZE	Si l'utilisateur a indiqué que la définition doit être synchronisée avec un autre objet, cette mesure exécute la synchronisation. La synchronisation peut être définie dans le fichier de l'Editeur SA2001.INI.
DEFMETUPDATEUSES	Met à jour la table des relations du dictionnaire de données (RELATN.DBF). Cette expression n'a pas de valeur de retour.
DEFNMETCHARCOUNT	Renvoie le nombre de caractère contenus dans la propriété de description de la définition.

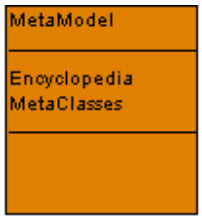
Mesure de définition	Description
DEFNMETLINECOUNT	Renvoie le nombre de lignes contenues dans la propriété de description de la définition.
DEFNMETWORDCOUNT	Renvoie le nombre de mots contenus dans la propriété de description de la définition.

9

Classe MetaModel

Introduction

L'objet Métamodèle donne accès aux objets des métaclasses d'une encyclopédie.



Rubriques contenues dans ce chapitre	Page

Attributs

Encyclopedia

Finalité
Fait référence à l'objet encyclopédie parent de l'objet métamodèle.

Paramètres
Lecture seule

MetaClasses

Finalité
Donne accès à la collection des objets métaclasse d'une encyclopédie.

Paramètres
Type de données : collection SA

Lecture seule

Exemple

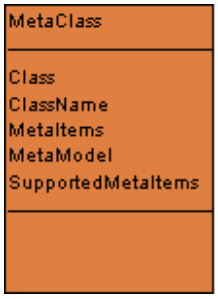
```
Dim coll As SACollection, i As Integer
Set coll = sa.Encyclopedia.metamodel.MetaClasses
  For i = 1 To coll.Count
    Debug.Print coll.Item(i).Class
  Next i
```

10

Classe MetaClass

Introduction

L'objet MetaClass renvoie des informations sur le niveau de classe des objets présents dans le référentiel.



Rubriques contenues dans ce chapitre	Page

Attributs

Class

Finalité
Numéro de classe de l'objet contenu dans le référentiel. Les valeurs de retour possibles sont 1 pour un diagramme, 2 pour un symbole et 3 pour une définition. Cette propriété est également appelée le "numéro de type principal".

Paramètres
Type de données : long

Lecture seule

Exemple

```
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

Finalité
Nom de la classe. Les noms de classe valides dans une encyclopédie sont Diagrams, Symbols et Definitions. Cette propriété est également appelée le "type principal".

Paramètres
Type de données : chaîne

Lecture seule

Exemple

```
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

Finalité
Cette propriété donne accès à la classe MetaItems de tous les méta-éléments.

Paramètres
Type de données : SACollection

Lecture seule

Exemple

```
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

Finalité
Cette propriété donne accès à la classe MetaModel parente.

Paramètres
Lecture seule

SupportedMetaItems

Finalité
Cette propriété donne uniquement accès aux méta-éléments activés pour cette encyclopédie.

Paramètres
Type de données : SACollection

Lecture seule

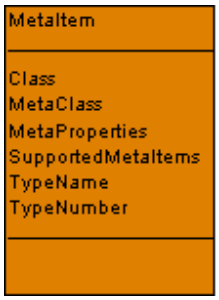
Exemple
```
Dim sa As SA2001.Application

Set sa = New SA2001.Application

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

Set sa = Nothing
```

11

Classe MetaItem

Introduction

Cet objet fournit des informations sur un type d'objet particulier présent dans l'encyclopédie. Il correspond aux informations contenues dans les fichiers saprops.cfg etusrprops.txt de certains types d'objets.



Attributs

Class

Finalité
Classe de l'objet. Les valeurs possibles sont 1 pour un diagramme, 2 pour un symbole et 3 pour une définition. Cette propriété est également appelée le "numéro de type principal".

Paramètres
Type de données : long

Lecture seule

Exemple

```
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

Finalité
Cette propriété donne accès à l'objet de la classe MetaClass parente.

Paramètres
Lecture seule

MetaProperties

Finalité
Cette propriété donne accès à toutes les propriétés prises en charge par ce type d'objet sous la forme d'une collection.

Paramètres
Type de données : SACollection

Lecture seule

Exemple

```
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

Finalité
Cette propriété donne uniquement accès aux méta-éléments activés pour cette encyclopédie.

Paramètres
Type de données : SACollection

Lecture seule

TypeName

Finalité
Nom du type d'objet.

Paramètres
Type de données : chaîne

Lecture seule

Exemple

```
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

Finalité
Constante numérique affectée à ce type d'objet par Rational System Architect.

Paramètres
Type de données : long

Lecture seule

Exemple

```
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

Classe MetaProperty

Rubriques contenues dans ce chapitre	Page

Introduction

L'objet MetaProperty permet d'obtenir des informations sur chaque propriété d'un objet contenu dans l'encyclopédie. Cet objet correspond au mot clé Property dans les fichiers saprops.cfg et usrprops.txt.

Attributs

Vous pouvez extraire les propriétés suivantes en vous servant de l'exemple qui suit et en remplaçant le nom indiqué par l'une des propriétés décrites ci-après.

```
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
Libellé long alternatif affecté par Rational System Architect à la propriété.

Type de données : chaîne

Lecture seule

AltLabelShort
Libellé court alternatif affecté par Rational System Architect à la propriété.

Type de données : chaîne

Lecture seule

Class
Type de classe de l'objet parent des propriétés. Cette propriété est également appelée le "numéro de type principal".

Type de données : long

Lecture seule

Default
Valeur par défaut définie pour la première utilisation de la propriété.

Type de données : chaîne

Lecture seule

EditFlags
Nombre de propriétés à renseigner pour une propriété.

Type de données : long

Lecture seule

EditLength
Longueur du texte de la propriété.

Type de données : long

Lecture seule

EditType
Reportez-vous au tableau ci-après pour obtenir la liste des types SAEditTypes.

Type de données : chaîne

Lecture seule

EditTypeNum
Reportez-vous au tableau ci-après pour obtenir la liste des constantes numériques SAEditType.

Type de données : long

Lecture seule

Help
Texte d'aide défini pour cette propriété.

Type de données : chaîne

Lecture seule

HelpID
Numéro d'ID du texte de l'aide.

Type de données : long

Lecture seule

Key
Indique si la propriété est une clé ou non.

Type de données : booléen

Lecture seule

KeyedBy
Indique si la propriété est renseignée par une autre propriété ou non.

Type de données : SACollection

Lecture seule

Libellé
Libellé standard affecté par Rational System Architect à la propriété.

Type de données : chaîne

Lecture seule

MetaItem
Donne accès à l'objet MetaItem parent.

Lecture seule

Name
Nom de la propriété.

Type de données : chaîne

Lecture seule

OfFlags
Nombre d'indicateurs associés à la propriété.

Type de données : long

Lecture seule

OfMajorType
Constante numérique de la classe des objets parents.

Type de données : long

Lecture seule

OfMajorTypeName
Nom de type de la classe des objets parents.

Type de données : chaîne

Lecture seule

OfMinorType
Constante actuelle du type numérique des objets parents.

Type de données : long

Lecture seule

OfMinorTypeName
Nom actuel du type des objets parents.

Type de données : chaîne

Lecture seule

OfRelateType
Numéro de la relation des objets parents. Pour obtenir la liste complète des numéros de relation, voir le chapitre 15.

Type de données : long

Lecture seule

RangeMax
Valeur maximale pour la plage de valeurs.

Type de données : long

Lecture seule

RangeMin
Valeur minimale pour la plage de valeurs.

Type de données : long

Lecture seule

Obligatoire
Propriété obligatoire pour l'objet.

Type de données : long

Lecture seule

TypeNumber
Numéro de type de l'objet parent.

Type de données : long

Lecture seule

EditType	Numéro	Description
Texte	1	La propriété spécifiée comme texte peut se constituer d'une liste ou de caractères alphanumériques saisis par l'utilisateur.
Date	2	La longueur de la propriété doit être de 10 caractères et dépend du format de date défini dans Windows.
Numérique	3	Indique que la propriété doit contenir des valeurs numériques.
Booléen	4	La propriété peut avoir l'une des deux valeurs suivantes : True (T) ou False (F). Elle apparaît dans une boîte de dialogue de définition sous la forme d'une case à cocher.
Expression	5	La valeur de la propriété doit être saisie comme une série de chaînes séparées par un signe +. Ce terme a été remplacé par EXPRESSIONOF.
Minispec	6	Propriété dont la valeur indique la logique de traitement d'un symbole de processus. Les minispecs sont rédigées dans une syntaxe formelle souvent appelée "Structured English" (langage structuré en anglais).
Temps	7	La propriété comportera une horodate dans la notation correspondant au format horaire défini dans Windows.
ListOf	8	Forme une relation de type "un à un" ou "un à plusieurs" entre le type de définition actif et le type de définition indiqué après le paramètre ListOf pour tous les éléments figurant dans la propriété de la liste.
ExpressionOf	9	Forme une relation de type "un à un" ou "un à plusieurs" entre le type de définition actif et le type de définition indiqué après le paramètre ExpressionOf pour tous les éléments figurant dans la propriété.

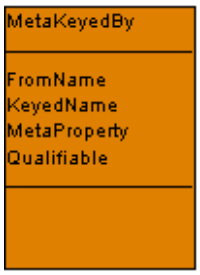
EditType	Numéro	Description
OneOf	10	Forme une relation de type "un à un" entre le type de définition actif et le type de définition indiqué après le paramètre OneOf pour l'élément figurant dans la propriété.
TemplateOf	11	Syntaxe d'un modèle de déclencheur.
ParmListOf	12	Forme une relation de type "un à un" ou "un à plusieurs" entre le type de définition actif et le type de définition indiqué par un nom et des paramètres après le paramètre ParmListOf pour tous les éléments figurant dans la propriété de la liste.
ParmOneOf	13	Forme une relation de type "un à un" entre le type de définition actif et le type de définition indiqué par un nom et des paramètres après le paramètre OneOf pour l'élément figurant dans la propriété.

13

Classe MetaKeyedBy

Introduction

Cette rubrique décrit la classe MetaKeyedBy avec ses attributs illustrés ci-après.



Attributs

FromName

Finalité
Nom d'origine utilisé pour qualifier la clé d'une métapropriété.

Paramètres
Type de données : chaîne

Lecture seule

KeyedName

Finalité
Nom saisi de la métapropriété.

Paramètres
Type de données : chaîne

Lecture seule

MetaProperty

Finalité
Donne accès à la classe MetaProperty parente.

Paramètres
Lecture seule

Qualifiable

Finalité
Indique si la métapropriété est qualifiable ou non, c'est-à-dire si elle garde une référence avec sa structure de clé.

Paramètres
Type de données : booléen

Lecture seule

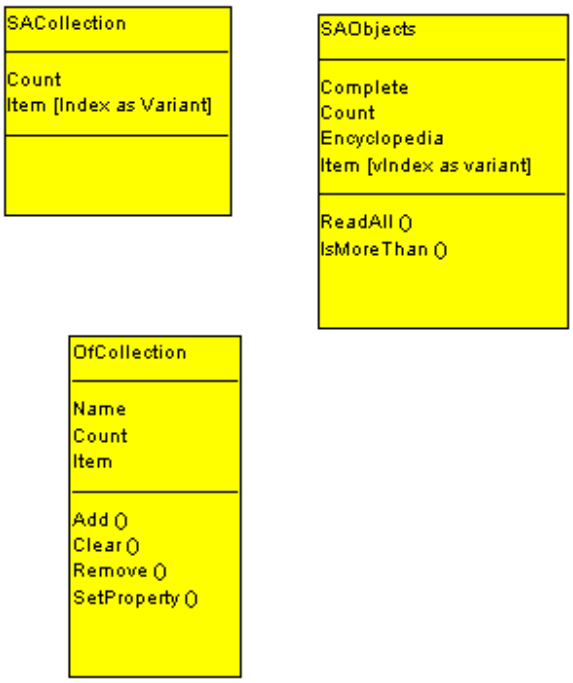
14

Collections de Rational System Architect

Rubriques contenues dans ce chapitre	Page

Introduction

Voici une illustration des classes de collections de Rational System Architect avec leurs attributs.



SAObjects

Cette classe de collection permet de manipuler un groupe d'objets Rational System Architect de même nature (diagrammes, symboles ou définitions).

Complete

Finalité
Valeur booléenne égale à "true" quand tous les membres possibles de la collection ont été lus.

Paramètres
Type de données : booléen

Lecture seule

Exemple
```
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
  Boucle
Debug.Print coll.Complete
Debug.Print coll.Count
```

Nombre

Finalité
Valeur qui représente le nombre de membres de la collection.

Paramètres
Type de données : long

Lecture seule

Exemple

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

Encyclopedia

Finalité
Référence récursive à la classe d'encyclopédie de l'objet actif dans la collection.

Paramètres
Lecture seule

Item(Index)

Finalité
Chaque membre de la collection possède un numéro d'index spécifique qui lui est attribué lors de la création de la collection. La variable item est un objet basé sur l'index de la collection. Son numéro d'index, ou son nom s'il est connu, peut référencer cet objet.

Paramètres
Type de données : booléen

Lecture seule

Exemple

```
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)

Finalité
Renvoie la valeur "true" si le nombre d'éléments présents dans la collection est supérieur à la valeur actuelle de l'index (s'utilise quand la collection est lu élément par élément).

Paramètres
Type de données : booléen

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

ReadAll

Finalité
Lit toutes les occurrences possibles du type indiqué dans la collection.

Exemple
```
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

Cette classe de collection permet de manipuler un groupe de propriétés Rational System Architect de même nature.

Nombre

Finalité
Renvoie une valeur numérique qui représente le nombre de membres de la collection.

Paramètres
Type de données : long

Lecture seule

Exemple
```
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)

Finalité
Chaque membre de la collection possède un numéro d'index spécifique qui lui est attribué lors de la création de la collection. La variable item est un objet basé sur l'index de la collection. Son numéro d'index, ou son nom s'il est connu, peut référencer cet objet.

Paramètres
Type de données : booléen

Lecture seule

Exemple

```
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

Cette classe de collection permet de manipuler un groupe de diagrammes ou de définitions qui constituent des composants ayant un type de propriété spécifique répertorié dans le tableau ci-après.

Type de propriété	Description
ListOf	Forme une relation de type "un à un" ou "un à plusieurs" entre le type de définition actif et le type de définition indiqué après le paramètre ListOf pour tous les éléments figurant dans la propriété de la liste.
OneOf	Forme une relation de type "un à un" entre le type de définition actif et le type de définition indiqué après le paramètre OneOf pour l'élément figurant dans la propriété.

Nom

Finalité
Nom de la liste de propriétés

Paramètres
Type de données : chaîne

Lecture seule

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

Nombre

Finalité
Valeur numérique qui représente le nombre de membres de la collection.

Paramètres
Type de données : long

Lecture seule

Exemple
```
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

Finalité
Chaque membre de la collection possède un numéro d'index spécifique qui lui est attribué lors de la création de la collection. La variable item est un objet basé sur l'index de la collection. Son numéro d'index, ou son nom s'il est connu, peut référencer cet objet.

Paramètres
Type de données : booléen

Lecture seule

Exemple
```
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

Finalité
Ajoute un objet de diagramme ou de définition dans la liste de propriétés.

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

`OfCollection Object`

Utilisation : obligatoire

Type de données : objet

Toute classe OfCollection instanciée

`Item`

Utilisation : obligatoire

Type de données : chaîne

Objet de diagramme ou de définition à ajouter dans la liste de propriétés

`Before`

Utilisation : facultatif (inutilisable si le paramètre After est déjà défini)

Type de données : long

Numéro de l'objet de diagramme ou de définition après lequel le nouvel objet de diagramme ou de définition sera ajouté dans l'objet de collection.

`After`

Utilisation : facultatif (inutilisable si le paramètre Before est déjà défini)

Type de données : long

Numéro de l'objet de diagramme ou de définition avant lequel le nouvel objet de diagramme ou de définition sera ajouté dans l'objet de collection.

Exemple
```
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

Finalité
Supprime l'objet OfCollection de tous les éléments contenus dans la collection.

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

Remove

Finalité
Supprime un objet de diagramme ou de définition dans la collection.

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

`OfCollection Object`

Utilisation : obligatoire

Type de données : objet

Toute classe OfCollection instanciée

`Index`

Utilisation : obligatoire

Type de données : long

Numéro de l'élément de diagramme ou de définition à supprimer dans l'objet collection.

Exemple

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

SetProperty

Finalité

Enregistre la liste de propriétés dans l'objet OfCollection. Il est également important d'enregistrer l'objet de définition lui-même car il contient la liste de propriétés dans laquelle la collection est enregistrée.

Exemple

```
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 Events

Introduction

Dans certains cas, il peut être nécessaire d'exercer un certain contrôle sur les actions que les utilisateurs exécutent et sur les réponses de Rational System Architect à ces actions. Il peut s'agir par exemple d'interdire certaines opérations, d'émettre des avertissements, d'imposer des modes opératoires ou d'exécuter des opérations supplémentaires. Dans ces situations, il est nécessaire de pouvoir identifier ces actions de manière explicite au moment où elles se produisent et d'y répondre en conséquence. Les actions qui se produisent à un moment déterminé sont appelées des “"événements".” ". Ces événements peuvent être gérés dans Rational System Architect dès qu'ils se produisent. Rational System Architect reconnaît un certain nombre d'événements, par exemple lorsqu'un symbole est disposé sur un diagramme, quand un diagramme est enregistré ou qu'une encyclopédie est ouverte. Le tableau suivant répertorie tous les événements reconnus et détaille les variables associées.

Application Events

AuditIDChanged

Finalité

Modification de l'ID d'audit. Le nouvel ID d'audit est indiqué dans l'argument NewAuditID. Vous pouvez modifier l'ID d'audit au moyen de **l'option de menu Fichier,…** ID audit.

Syntaxe

```
AuditIDChanged(NewAuditID)
NewAuditID
```

Utilisation : obligatoire
Type de données : chaîne
Nom d'utilisateur indiqué pour remplacer la valeur de l'ID d'audit.

Exemple

DiagramClose

Finalité

Fermeture d'un diagramme.

Syntaxe

```
DiagramClose(hdgm)
hdgm
```

Utilisation : obligatoire
Type de données : long
Le descripteur du diagramme est indiqué dans l'argument hDgm.

Exemple

DiagramOpen

Finalité

Ouverture d'un diagramme.

Syntaxe

DiagramOpen(hdgm)

hDgm

Utilisation : obligatoire
Type de données : long
Le descripteur du diagramme est indiqué dans l'argument hDgm.

Exemple

DiagramSave

Finalité
Sauvegarde d'un diagramme.
Exemple

EncyClose

Finalité
Fermeture d'une encyclopédie.
Exemple

EncyOpen

Finalité
Ouverture d'une encyclopédie.
Exemple

MainMenuUpdate

Finalité
Mise à jour du menu principal. Il s'agit des entrées du menu Fichier et des autres menus associés.
Exemple

MethodMenuUpdated

Finalité
Mise à jour du menu Dessin. Cet événement se produit quand un nouveau type de menu déclenche une réorganisation du menu et l'ajout de nouvelles options de menu.
Exemple

ReportsMenuUpdate

Finalité
Mise à jour du menu Rapports. Cet événement se produit quand le menu Rapports est modifié parRational System Architect suite aux vérifications des règles ou des rapports spécifiques aux différentes méthodes.
Exemple

ToolsMenuUpdate

Finalité
Mise à jour du menu Outils. Cet événement se produit quand le menu Outils est mis à jour pour refléter les options d'outil applicables à certaines des méthodes utilisées.
Exemple

ShowNode

Finalité
Cet événement se produit quand le filtre VBA de l'explorateur est actif. Définissez la valeur de RetCode pour afficher ou masquer cet élément.

Syntaxe

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

`TabIndex`

Utilisation : obligatoire
Type de données : entier

Nom d'onglet	Numéro d'index d'onglet
Toutes les méthodes	1
Modélisation des données	2
Processus métier	3
OO existant	4
Application	5
Structuré	6
Organisation	7
Technologie	8
Emplacement	9
Direction du métier	10
UML	11
XML	12

`TabName`
Utilisation : obligatoire
Type de données : chaîne
Voir le tableau ci-dessus.

`ddId`
Utilisation : obligatoire
Type de données : long

`Major`
Utilisation : obligatoire
Type de données : long

`Minor`
Utilisation : obligatoire
Type de données : long

`Nom`
Utilisation : obligatoire
Type de données : chaîne

`Memo`
Utilisation : obligatoire
Type de données : chaîne

`LockFlags`
Utilisation : obligatoire
Type de données : long

`RetCode`

Utilisation : obligatoire
Type de données : état de noeud

Etat de noeud	Numéro
StatusDONTSHOW	0
StatusSHOW	1
StatusDONTCARE	2

Exemple

ShutDown

Finalité
Rational System Architect est arrêté.
Exemple

StartUp

Finalité
Rational System Architect est démarré.
Exemple

SymbolEvent

Finalité
Placement d'un symbole sur un diagramme.

Syntaxe

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

Utilisation : obligatoire
Type de données : long
Descripteur de diagramme

hSym

Utilisation : obligatoire
Type de données : long
Descripteur de symbole

hSymOther

Utilisation : obligatoire
Type de données : long
Descripteur du symbole de ligne (auquel le symbole de noeud est en cours d'attachement/détachement)

SymEvent

Utilisation : obligatoire
Type de données : SYMEVENTS

SYMEVENTS	Numéro	Description
ADDCONN	64	Le symbole de noeud est attaché à un symbole de ligne.
BREAKCONN	65	Suppression de la liaison entre le symbole de noeud et le symbole de ligne.
SYM_DESELECTED	257	Désélection d'un symbole.
SYM_SELECTED	256	Sélection d'un symbole.
TRANSFORMED	66	Le symbole est transformé en un autre symbole. Pour certains symboles, vous pouvez exécuter cette opération en cliquant avec le bouton droit sur le symbole puis en sélectionnant l'option Transformer.

`Ldata`

Utilisation : obligatoire
Type de données : long

Instructions pour ajouter des macros aux menus à l'aide d'un programme

Depuis Rational System Architect version 10.1, les menus et les barres d'outils ne sont plus soumis à un format fixe comme ils l'étaient auparavant. Cela signifie que l'utilisateur peut désormais personnaliser les menus et conserver ces personnalisations. Les seules exceptions sont la barre d'outils et le menu Dessin pour lesquels les personnalisations sont limitées à la liste par défaut des commandes et des outils de dessin.
La finalité de cette modification est d'autoriser des configurations personnalisées des menus et des barres d'outils pour des groupes d'utilisateurs spécifiques au moyen du Gestionnaire de catalogues SA.
Dans Rational System Architect version 10.0 et les versions précédentes, les codes de menu de macro étaient écrits en tenant compte du fait que Rational System Architect pouvait supprimer certains menus puis les recréer. Cette suppression automatique signifiait que, si des utilisateurs souhaitaient la disparition de certaines options de menu au cours d'une session, ils n'avaient pas besoin de les supprimer explicitement à l'aide d'un programme. Par exemple, lors de la fermeture d'un diagramme, – le menu Outils était supprimé puis recréé. Il suffisait alors à l'utilisateur de réinsérer l'option de menu désirée s'il en avait besoin à un moment donné. A présent que Rational System Architect ne supprime plus les menus, ces options de menu demeurent visibles sauf si le code de la macro est modifié de manière à les supprimer (masquer). Si vous souhaitez modifier des macros de ce type intégrées dans Rational System Architect version 10.0 ou antérieure, vous trouverez les instructions requises à cet effet dans le manuel de conversion de Rational System Architect (fichier conversion.pdf) , disponible sur le site de support IBM.
Cette section fournit des instructions pour ajouter des macros aux menus deRational System Architect version 10.1 et ultérieures.

Comment ajouter une macro à un menu standard de Rational System Architect ?

La méthode principale SA2001.Application qui permet d'ajouter des macros est la suivante :

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

Cette fonction ajoute la macro indiquée par MacroName (qui comprend les arguments"<Project>.<Module>.<Subroutine>") avec la légende MacroItemCaption dans le menu SA/Utilisateur indiqué par InMenuTitleCaption avant l'option de menu indiquée par BeforeMenuItemCaption. Si ce dernier paramètre n'est pas spécifié, l'option de menu est ajoutée à la fin du menu. S'il a la valeur #TOP#, l'option de menu est insérée au début du menu.

Si elle aboutit, cette fonction renvoie la valeur 0. Si elle échoue, elle renvoie une autre valeur.

Exemple :

```
    Set App = New SA2001.Application
        x = App.InsertMacroItemInMenu("MyProject.MyModule.MySub", "&Test
Item", "&Tools","Ma&cros")
```

Cette syntaxe insère l'élément "Test Item" dans le menu Tools (Outils) juste avant l'option de menu Macros. Lorsque l'utilisateur clique sur cette option, le système exécute le sous-programme "MySub" dans le module "MyModule" dans le projet "MyProject".

Le sous-programme indiqué doit exister préalablement pour que cette opération aboutisse. De même les légendes doivent impérativement être spécifiées exactement telles qu'elles figurent dans le menu.

Par exemple, Tools sera représenté par la chaîne &Tools

La perluète (signe &) place un trait de soulignement sous le caractère qui suit. Cette commande distingue les majuscules des minuscules.

Le nom du sous-programme ne doit pas être suivi de parenthèses ().

L'ajout d'images BMP aux options de menu se fait au moyen de la méthode AssignBMPtoMacroItem.

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

Si elle aboutit, cette fonction renvoie la valeur 0. Si elle échoue, elle renvoie une autre valeur.

Elle doit être exécutée une seule fois avant d'ajouter la macro au menu.

Par exemple :

```
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")
```

Cette syntaxe ajoute l'image piccy.bmp à la macro MySub puis ajoute cette macro aux menus comme dans l'exemple précédent. Si vous souhaitez par la suite ajouter cette option de menu ailleurs dans vos menus, notez que les menus Personnaliser (accessibles en cliquant sur un menu avec le bouton droit) afficheront maintenant l'image Piccy.BMP et la légende "Test Item".

Utilisation des événements
Les événements SA2001.Application suivants peuvent s'utiliser avec les codes de macro pour modifier la visibilité des menus :

- MainMenuUpdate : quand l'ensemble d'un menu est affecté.
- MethodMenuUpdate : quand des options de menu de type dictionnaire sont mises à jour.
- ToolsMenuUpdate : quand des options de menu de type outil sont mises à jour.
- ReportsMenuUpdate : quand des options de menu de type rapport sont mises à jour.
- App_ShutDown : quand SA s'arrête.

Les événements ne sont plus nécessaires pour conserver les options définies par l'utilisateur dans les menus. Toutefois, si vous souhaitez supprimer des options dans un menu, les extraits de code suivants donnent une indication de la méthode à suivre pour gérer ce type d'événement. Remarque : Le projet a pour nom "MyProject".

MODULE - AutoExec

```
' Module principal du gestionnaire d'événements basé sur VBA
Dim EventHandler As EventCls
' Sous-programme principal qui démarre le gestionnaire d'événements
Sub Main()
  Set EventHandler = New EventCls     ' Créer un gestionnaire d'événements
  Set EventHandler.App = Application  ' Relier le gestionnaire d'événements à SA
  ' Exécuter ici les autres tâches liées au démarrage
  ' Ajouter ici les images BMP et créer les menus instantanés
  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
  ' Remarque : Quand un menu instantané est créé, il est conservé jusqu'à sa
suppression avec 'RemovePopupMenu' et peut être ajouté ou supprimé des menus
sans limite.

  ' Créer le menu instantané avec son fichier bitmap
  x = Application.CreatePopUpMenu("Sample Macros","SAWORD.BMP")
  ' Ajouter l'option dans le menu instantané
  x = Application.InsertMacroItemInMenu("MyProject.MyModule.PRINTDIAGRAMS",
"&Print a Diagram", "Sample Macros")
End Sub
```

CLASS - EventCls

```
Public WithEvents App As Application  ' Application qui déclenchera les événements

Private Sub Class_Initialize()
  ' Aucune action requise
End Sub

Private Sub Class_Terminate()
  ' Aucune action requise
```

```
End Sub

Private Sub App_ReportsMenuUpdate()
  ' Le menu Rapports est recréé
  Dim x As Long
  x = App.SetSeparatorBefore("&Report Generator...", "&Reports", True)

  ' Insérer dans le menu Reports, avant l'option de menu Report Generator..., une
option de menu libellée "Print all Diagrams" qui appellera la méthode PrintDiagrams2
  x = App.InsertMacroItemInMenu("MyProject.MyModule.PRINTDIAGRAMS2", "Print
all &Diagrams", "&Reports", "&Report Generator...")

  ' Insérer le menu instantané Sample Macros dans le menu Reports avant l'option
Print all Diagrams.
  ' Remarque : Ne conservez pas ici le mode opératoire suppression/recréation des
menus instantanés sauf si cela s'impose. Faites-le une seule fois à l'initialisation.
  x = App.InsertPopupMenuItemInMenu("Sample Macros", "&Reports", "Print all
&Diagrams")
End Sub
```

MODULE - MyModule

```
Public Sub PrintDiagrams()
        ' Code utilisateur à exécuter
        MsgBox "Print Diagrams"
End Sub
Public Sub PrintDiagrams2()
        ' Code utilisateur à exécuter
        MsgBox "Print Diagrams2"
End Sub
```

Comment ajouter un menu instantané ?
Utilisez la méthode CreatePopUpMenu en indiquant le nom que vous souhaitez attribuer au menu instantané et un fichier .bmp le cas échéant. Le menu instantané vient s'ajouter à la collection racine des menus instantanés dans SA.
Lorsque vous quittez SA, il n'est plus nécessaire que la classe de gestionnaire d'événements supprime les menus instantanés ajoutés à l'aide de la méthode RemovePopUpMenu.

Comment ajouter des macros à un menu instantané ?

La méthode est la même que pour les ajouter aux menus SA.

Comment ajouter un menu instantané à un menu SA ?
Vous pouvez ajouter un menu instantané dans un menu Rational System Architect exactement comme vous ajoutez une macro dans un menu, excepté que vous devez employer la méthode InsertPopupMenuItemInMenu et que vous ne devez pas indiquer de nom de macro.
Par exemple :
InsertPopupMenuItemInMenu(PopUpName as String, InMenuTitleCaption as String, [BeforeMenuItemCaption as string]) as long

Comment ajouter un séparateur dans une option de menu ?
Utilisez la méthode SetSeparatorBefore, indiquez le nom de menu et le nom d'option, puis ajoutez "True" pour insérer un séparateur, ou "False" si vous n'en voulez pas.

Comment supprimer une option dans un menu ?
Vous pouvez supprimer des menus instantanés ou des options de menu avec la même méthode : utilisez la méthode RemoveItemFromMenu et indiquez le nom du menu et celui de l'option de menu.
Remarque : Vous pouvez uniquement supprimer les options que vous avez vous-même ajoutées. En particulier, vous ne pouvez pas supprimer les options de menu propres àRational System Architect. Les options ne sont pas réellement supprimées ; elles sont uniquement masquées. Ceci permet notamment de placer ces options à un autre emplacement dans le système de menus. Notez que vous pouvez indiquer“” à la place du nom de menu puisque ce nom de menu sera ignoré désormais. L'option sera masquée où qu'elle se trouve dans le système de menus.
Dans la mesure où tous les menus instantanés et les outils qui ont été ajoutés au système de menus sont désormais gardés, même après le redémarrage de Rational System Architect, si un menu instantané devient inutile, vous devez le supprimer explicitement. TLa méthode App.RemovePopUpMenu(<PopupName>), permet de supprimer totalement un menu instantané du système de menus.
Si une macro utilise appelle cette méthode, par exemple à l'arrêt de Rational System Architect, le menu instantané concerné et toutes ses personnalisations seront supprimés à chaque arrêt du programme. Ceci aura pour conséquence que la personnalisation de ce menu instantané ne sera pas conservée entre deux sessions SA.

16

Relations de Rational System Architect

Introduction

Ce chapitre décrit les relations existantes entre les objets du référentiel de Rational System Architect. L'Explorateur d'objets SA contient un type de données énumératif appelé RELATETYPE. Ce type de données permet de stocker toutes les relations que vous pouvez utiliser dans les macros VBA.

Toutes les classes de définition, de symbole et de diagramme utilisent la méthode **GetRelatedObjects**. Pour exécuter cette méthode, vous devez indiquer la relation existante parmi celles qui sont répertoriées dans le tableau ci-dessous. En outre, la classe d'encyclopédie utilise la méthode**GetRelationMetric**. Vous devez indiquer l'un des types de relations comme paramètre obligatoire afin d'exécuter la mesure de relation désirée.

Relation Types

RELATETYPE	Description	Numéro
RELNULL	NULL	0
RELNULL2	NULL	1
RELDIAGRAMCON	Le diagramme contient des symboles	2
RELCONDIAGRAM	Symbole contenu dans les diagrammes	3
RELSHOWTO	Le symbole est développé sur un diagramme (enfant)	4
RELSHOWFROM	Un diagramme est développé à partir d'un symbole (parent)	5
RELCONNSTART	Un symbole de noeud se connecte au début d'une ligne de symbole	6
RELSTARTAT	Le début d'une ligne se connecte à un symbole de noeud	7
RELCONNEND	Un symbole de noeud se connecte à la fin d'une ligne	8
RELENDAT	La fin d'une ligne se connecte à un symbole de noeud	9
RELFLAGSENDS	Un module se connecte à et envoie des données via un symbole d'indicateur	10
RELFLAGSTR	Un symbole d'indicateur se connecte à et reçoit des données d'un module	11
RELFLAGRECVS	Un module se connecte à et reçoit des données d'un symbole d'indicateur	12
RELFLAGEND	Un symbole d'indicateur se connecte à et fournit des données à un module	13

RELATETYPE	Description	Numéro
RELDFEELEMENT	Une expression utilise des données (éléments ou structures)	14
RELELEMENTDFE	Des données (éléments ou structures) sont utilisées par une expression	15
RELEXPLAINEDBY	Un symbole est expliqué par un commentaire	16
RELEXPLAINS	Un commentaire explique un symbole	17
RELPARTFULFILLS	Un symbole adresse une exigence, un plan de test, etc.	18
RELPARTFULFILLEDBY	Une exigence, un plan de test, etc. est adressé par un symbole	19
RELCOMPFULFILLS	Un symbole est défini par une définition	20
RELDEFINEDBY	Un symbole est défini par une définition	20
RELCOMPFULFILLEDBY	Une définition définit un symbole	21
RELDEFINES	Une définition définit un symbole	21
RELISQUALIFIEDBY	Un symbole de ligne ou de noeud est qualifié par un symbole d'indicateur	22
RELQUALIFIES	Un symbole d'indicateur qualifie un symbole de ligne (ou un symbole de noeud)	23
RELISA	Une définition est une instance d'une définition	24
RELINSTBY	Une définition est instanciée par une définition	25
RELIDENTIFIES	Une définition identifie une définition	26
RELKEYEDBY	Une définition est identifiée par une définition	27

RELATETYPE	Description	Numéro
RELEMBEDS	Un symbole de noeud imbrique totalement un symbole	28
RELISEMBEDDEDBY	Un symbole de noeud est totalement imbriqué par un symbole de noeud	29
RELISPARENTIN	Une définition est un parent d'une définition dans une relation parent enfant	30
RELHASPARENTOF	Une définition a une définition parent dans une relation parent enfant	31
RELISCHILDIN	Une définition est un enfant d'une définition dans une relation parent enfant	32
RELHASCHILDOF	Une définition a une définition enfant dans une relation parent enfant	33
RELCOMPRISES	Une définition comprend une définition	34
RELISPARTOF	Une définition fait partie d'une définition	35
RELISCATZNOF	Une définition de catégorisation catégorise une définition d'entité générique	36
RELHASCATZN	Une définition d'entité générique a une catégorisation d'une définition de catégorisation	37
RELISCATGRYIN	Une définition d'entité est une catégorie dans une catégorisation	38
RELHASCATGRY0F	Une catégorisation a une catégorie d'une définition d'entité	39
RELISCHILDOF	Le symbole est un enfant d'un autre symbole dans un diagramme hiérarchique uniquement	40
RELISPARENTOF	Le symbole est le parent d'un autre symbole dans un diagramme hiérarchique uniquement	41

RELATETYPE	Description	Numéro
RELISFIRSTCHILDOF	Le symbole est le premier enfant d'un autre symbole (par exemple, le symbole le plus à gauche) dans un diagramme hiérarchique uniquement	42
RELHASFIRSTCHILD	Le symbole a un autre symbole comme premier enfant dans un diagramme hiérarchique uniquement	43
RELISNEXTSIBLING	Le symbole est l'élément apparenté suivant d'un autre élément apparenté dans un diagramme hiérarchique uniquement	44
RELISPRIORSIBLING	Le symbole est l'élément apparenté précédent d'un autre symbole dans un diagramme hiérarchique uniquement	45
RELISINDEXOF	Modèle de données : chemin d'accès, index d'entité ou table	46
RELISINDEXEDBY	Une définition est indexée par une autre définition	47
RELORIGINATESFROM	Une définition provient d'une définition (par exemple, diagramme d'écran graphique)	48
RELISORIGINOF	Une définition est l'origine d'une définition	49
RELISBASEDON	Une définition est basée sur une définition (généralement un élément de données)	50
RELISBASISFOR	Une définition est la base d'une définition dérivée	51
RELISLINKEDTO	Un symbole est lié à un autre symbole	52
RELISLINKEDWITH	Un symbole est lié avec un autre symbole	53
RELUSER AND RELUSERCOMPLEMENT	Utilisateur et complément d'utilisateur	54 à 83

RELATETYPE	Description	Numéro
RELPOPKIN AND RELPOPKINCOMPLEMENT	IBM et complément IBM	84 à 111
RELREPRESENTS	Le symbole de diagramme de l'explorateur représente un objet	112
RELISPRESENTEDBY	L'objet est représenté par un symbole de diagramme de l'explorateur	113
RELPOPKIN AND RELPOPKINCOMPLEMENT	IBM et complément IBM	114 à 125
RELUSER AND RELUSERCOMPLEMENT	Utilisateur et complément d'utilisateur	126 à 135
RELLINKS TO	L'objet est lié à l'objet dans DOORS	136
RELISLINKEDFROM	L'objet est lié à partir de l'objet dans DOORS	137
RELISTOBESENTTO	L'objet est à envoyer au module DOORS	138
RELISTORECEIVE	Le module DOORS doit recevoir l'objet	139
RELHASBEENSENTTO	L'objet a été envoyé au module DOORS	140
RELHASRECEIVED	Le module DOORS a reçu l'objet	141

17

Types de zones de Rational System Architect

Introduction

Ce chapitre décrit tous les types de zones présentés dans l'Explorateur d'objets de Rational System Architect sous le type énumératif FLDTYPE. Ces constantes déterminent le format dans lequel les données sont renvoyées. Dans la plupart des cas, Rational System Architect définit le type de zone de façon interne. Ce type énumératif est utilisé comme paramètre obligatoire pour la méthode GetRelationMetric de la classe Encyclopedia. Il constitue également un paramètre facultatif pour les méthodes GetMetric des classes Diagram, Symbol et Definition.

Le tableau ci-après répertorie tous les types de zones avec leur description.

Types de zones

FLDTYPE	Numéro	Description
FLDTYPAUTO	65	Il s'agit de la zone par défaut sélectionnée de façon interne par Rational System Architect.
FLDTYPCHARACTER	67	Contient une chaîne de 256 caractères maximum.
FLDTYPDATE	68	Zone de date (par exemple MM/JJ/AAAA)
FLDTYPLOGICAL	76	Zone de valeur booléenne
FLDTYPMEMO	77	Les zones de mémo contiennent des blocs de texte pouvant aller jusqu'à 4095 caractères.
FLDTYPNUMERIC	78	Indique que la zone doit contenir des valeurs numériques.
FLDTYPTIME	84	Indique une zone de valeur horaire (par exemple heures:minutes:secondes)

18

Erreurs de Rational System Architect

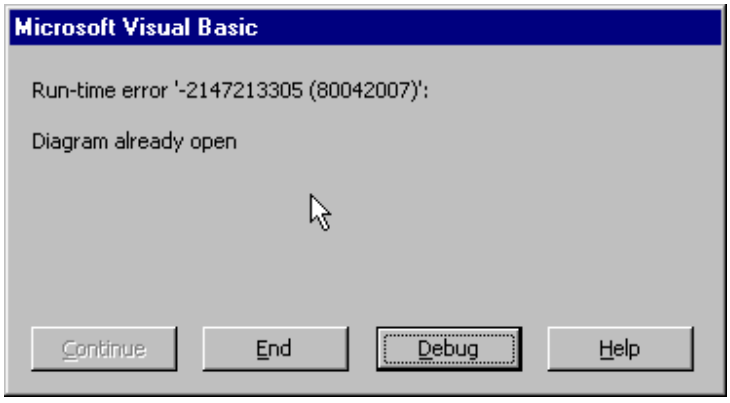
Introduction

Ce chapitre décrit les erreurs générées par Rational System Architect qui peuvent être interceptées pendant l'exécution d'un code. Il ne détaille pas les erreurs pouvant exister dans le code lui-même (par exemple, l'erreur de code Visual Basic numéro 91 "Variable Object ou variable With block non définie" quand aucune instance de la variable n'a été créée).

Gestion des erreurs

Les propriétés et les méthodes du modèle d'objet de Rational System Architect exécutent leur code de manière cohérente mais, malheureusement, les données de base ne sont pas toujours celles attendues et des erreurs peuvent se produire pendant l'exécution normale d'une macro VBA. Plusieurs incidents peuvent apparaître pendant l'exécution et ces derniers doivent être interceptés et traités, sans quoi la macro s'arrête avant la fin de son exécution.

Quand une erreur se produit pendant l'exécution d'un code, la boîte de dialogue suivante apparaît. La méthode **Diagram.Show** a été exécutée pour ouvrir l'image du diagramme actif. Toutefois, le diagramme a déjà été ouvert à une autre étape du code et une erreur s'est produite.



L'utilisateur peut cliquer sur Terminer, pour arrêter le programme à l'étape actuelle, ou sur Déboguer. L'option Déboguer ouvre l'interface de l'Editeur de VBA et met en évidence l'emplacement du code auquel l'erreur s'est produite. S'il veut obtenir d'autres explications, l'utilisateur peut cliquer sur le bouton Aide.

Rational System Architect Erreurs

Plusieurs erreurs courantes peuvent être identifiées dans Rational System Architect. VBA peut suivre ces erreurs quand elles figurent dans une macro et envoyer un message au gestionnaire d'erreurs de VBA qui communiquera alors ces erreurs à l'utilisateur. Les différentes erreurs génèrent différents incidents. Quand une méthode s'exécute, elle renvoie une valeur à VBA pour indiquer l'état de l'appel. Si une exception se produit, VBA signale l'erreur correspondante. Le type énumératif SA2001Errors contient la liste complète des erreurs courantes de Rational System Architect.

Erreur	Numéro	Description
SAERR_BADDDID	8195	
SAERR_DIAGRAMNOTOPEN	8198	Le diagramme n'est pas ouvert. Soit le diagramme ne s'est ouvert à aucun moment, soit la méthode **.Hide** a déjà été appelée.
SAERR_DIAGRAMNOTSAME	8201	
SAERR_DIAGRAMOPEN	8199	Le diagramme est déjà ouvert. Soit le diagramme a déjà été ouvert, soit la méthode **.Show** a déjà été appelée.
SAERR_ENUMVARIANTERROR	8193	
SAERR_INVALIDCLASS	8202	
SAERR_INVALIDOBJECT	8194	
SAERR_INVALIDPROPERTY	8204	
SAERR_INVALIDTYPE	8207	
SAERR_NOTIMPLEMENTED	8196	
SAERR_OBJECTDOESNOTEXIST	8197	Soit l'objet n'a été instancié à aucun moment, soit il a été supprimé.
SAERR_OBJECTISLOCKED	8205	Soit la méthode OpenLock a été appelée précédemment, soit l'objet a été réservé ou figé par un autre utilisateur.

Erreur	Numéro	Description
SAER_OBJECTNOTFOUND	1025	
SAERR_OPENEDASREADONLY	8206	L'objet a été ouvert en mode lecture seule.
SAERR_REQUIREDPROPERTYABSENT	8192	La propriété référencée n'a pas de valeur ou n'existe pas.
SAERR_SA20001_IMF_ERROR	4096	Une erreur SAIMF s'est produite.
SAERR_SANOTRUNNING	1024	Rational System Architect n'a pas été démarré ou a été arrêté.
SAERR_SYMBOLHASNODIAGRAM	8200	Le symbole n'est référencé par aucun diagramme.
SAERR_TOOMANYOBJECTS	8203	

19

Support IBM

Introduction

Un certain nombre de ressources et d'outils d'aide vous permettant d'identifier et de résoudre les incidents sont à votre disposition. En cas d'incident avec le produit, vous pouvez :

vous reporter aux informations sur l'édition du produit pour en savoir plus sur les incidents connus, les solutions existantes et les informations d'identification et de résolution des incidents,

vérifier si un module téléchargeable ou un correctif est disponible pour résoudre l'incident,

consulter les bases de connaissances disponibles pour voir si votre problème a déjà été rencontré, résolu et expliqué.

Si vous avez besoin d'une assistance, prenez contact avec le service de support logiciel IBM® et signalez votre incident.

Rubriques contenues dans ce chapitre	Page

Pour contacter le service de support logiciel IBM Rational

Si vous ne parvenez pas à résoudre un incident à l'aide des ressources d'aide, contactez le service de support logiciel IBM® Rational®.

Remarque : si vous êtes un ancien client de Telelogic, consultez le site unique de référence pour toutes les ressources de support à l'adresse http://www.ibm.com/software/rational/support/telelogic/

Configuration requise :

Pour soumettre un incident au service de support logiciel IBM Rational, vous devez être détenteur d'un contrat de maintenance logicielle Passport Advantage® actif. Passport Advantage constitue l'offre globale de licence et de maintenance logicielle d'IBM (mises à niveau des produits et support technique). Vous pouvez vous inscrire à Passport Advantage en ligne à l'adresse http://www.ibm.com/software/lotus/passportadvantage/howtoenroll.html

- Pour en savoir plus, consultez les foires aux questions relatives à Passport Advantage à l'adresse http://www.ibm.com/software/lotus/passportadvantage/brochures_faqs_quickguides.html.
- Pour obtenir de l'aide, prenez contact avec votre interlocuteur IBM habituel.

Pour soumettre un incident en ligne (à partir du site Web IBM) au service de support logiciel IBM Rational, procédez comme suit :

- Enregistrez-vous en tant qu'utilisateur sur le site Web du service de support logiciel IBM Rational. Pour plus d'informations sur cet enregistrement, consultez la page http://www.ibm.com/software/support/.
- Inscrivez-vous comme appelant autorisé dans l'outil de demande de service.

Autres informations

Pour en savoir plus sur les actualités, les événements et autres informations relatives au produit, consultez le site Web IBM Rational : http://www.ibm.com/software/rational/.

Soumission d'incidents

Pour soumettre un incident au service de support logiciel IBM Rational, procédez comme suit :

1. Déterminez l'impact de l'incident. Lorsque vous soumettez un incident à IBM, vous devez indiquer un niveau de gravité. Vous devez donc comprendre et évaluer l'impact de cet incident sur les opérations.

Pour déterminer le niveau de gravité, utilisez le tableau suivant.

Gravité	Description
1	L'impact de l'incident est critique : l'utilisation du programme est impossible et l'impact sur les opérations est critique. La résolution de l'incident doit être immédiate.
2	L'impact de l'incident est important : le programme peut être utilisé, mais de façon très limitée.
3	L'incident a un certain impact sur les opérations : le programme peut être utilisé, mais certaines fonctions moins importantes ne sont pas disponibles.
4	L'impact de l'incident est minime : l'impact sur les opérations est minime ou une solution palliative acceptable a été mise en place.

2. Décrivez l'incident et rassemblez les informations nécessaires. Lorsque vous décrivez l'incident à IBM, soyez aussi précis que possible. Donnez toutes les informations utiles afin que les spécialistes du service de support logiciel IBM Rational puissent vous aider efficacement à résoudre l'incident. Pour gagner du temps, préparez les réponses aux questions suivantes :

    - Quelles versions du logiciel étaient en cours d'exécution lorsque l'incident s'est produit ?

    - Pour déterminer le nom et la version exacts du produit, utilisez l'option qui vous concerne :
        - Démarrez le gestionnaire d'installation IBM et cliquez sur **File** > **View Installed Packages**. Développez un groupe de package et sélectionnez un package pour en voir le nom et le numéro de version.
        - Démarrez le produit et cliquez sur **Aide** > **A propos de** pour voir le nom et le numéro de version de l'offre.

    - Quel système d'exploitation utilisez-vous ? Quelle est sa version (et celle des service packs ou des correctifs installés) ?

    - Avez-vous des fichiers journaux, traces et messages relatifs à l'incident ?

    - Pouvez-vous recréer l'incident ? Si oui, quelles étapes avez-vous exécuté pour recréer l'incident ?

    - Avez-vous apporté des modifications au système ? Avez-vous par exemple modifié le matériel, le système d'exploitation, le logiciel réseau ou d'autres composants du système ?

3. Utilisez-vous actuellement une solution palliative ? Si oui, préparez une description de cette solution.

4. Soumettez l'incident au service de support logiciel IBM Rational de l'une des façons suivantes :

    - En ligne : accédez au site Web du service de support logiciel IBM Rational à la page https://www.ibm.com/software/rational/support/. Dans le navigateur de tâches du support Rational, cliquez sur **Open Service Request**. Sélectionnez l'outil électronique de rapport d'incident et ouvrez un PMR (Problem Management Record) pour décrire l'incident.

    - Pour plus d'informations sur l'ouverture d'une demande de service, consultez la page http://www.ibm.com/software/support/help.html.

    - Vous pouvez également ouvrir une demande de service en ligne avec IBM Support Assistant. Pour plus d'informations, consultez la page http://www.ibm.com/software/support/isa/faq.html.
    - Par téléphone : pour identifier le numéro de téléphone à appeler dans votre pays ou dans votre région, consultez le répertoire IBM des contacts dans le monde à l'adresse http://www.ibm.com/planetwide/ et cliquez sur le nom de votre pays ou de votre région géographique.

    - Via votre interlocuteur IBM habituel : si vous ne parvenez pas à contacter le service de support logiciel IBM Rational en ligne ou par téléphone, contactez votre interlocuteur IBM habituel. Il pourra si nécessaire ouvrir une demande de service pour vous. Pour plus d'informations sur les contacts de chaque pays, consultez la page http://www.ibm.com/planetwide/.

20

Annexe :

Introduction

Ce chapitre contient des informations légales relatives à l'utilisation et aux marques d'IBM Rational System Architect.

Remarques

Le présent document peut contenir des informations ou des références concernant certains produits, logiciels ou services IBM non annoncés dans ce pays. Pour plus de détails, référez-vous aux documents d'annonce disponibles dans votre pays, ou adressez-vous à votre partenaire commercial IBM. Toute référence à un produit, logiciel ou service IBM n'implique pas que seul ce produit, logiciel ou service puisse être utilisé. Tout autre élément fonctionnellement équivalent peut être utilisé, s'il n'enfreint aucun droit d'IBM. Il est de la responsabilité de l'utilisateur d'évaluer et de vérifier lui-même les installations et applications réalisées avec des produits, logiciels ou services non expressément référencés par IBM.

IBM peut détenir des brevets ou des demandes de brevet couvrant les produits mentionnés dans le présent document. La remise de ce document ne vous donne aucun droit de licence sur ces brevets ou demandes de brevet. Si vous désirez recevoir des informations concernant l'acquisition de licences, veuillez en faire la demande par écrit à l'adresse suivante :

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785

Les informations sur les licences concernant les produits utilisant un jeu de caractères double octet peuvent être obtenues par écrit à l'adresse suivante :

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

Le paragraphe suivant ne s'applique ni au Royaume-Uni, ni dans aucun pays dans lequel il serait contraire aux lois locales.LE PRESENT DOCUMENT EST LIVRE EN L'ETAT SANS AUCUNE GARANTIE EXPLICITE OU IMPLICITE. IBM DECLINE NOTAMMENT TOUTE RESPONSABILITE RELATIVE A CES INFORMATIONS EN CAS DE CONTREFACON AINSI QU'EN CAS DE DEFAUT D'APTITUDE A L'EXECUTION D'UN TRAVAIL DONNE.

Certaines juridictions n'autorisent pas l'exclusion des garanties implicites. Dans ce cas, l'exclusion ci-dessus ne vous sera pas applicable.

Le présent document peut contenir des inexactitudes ou des coquilles. Ce document est mis à jour périodiquement. Chaque nouvelle édition inclut les mises à jour. IBM peut, à tout moment et sans préavis, modifier les produits et logiciels décrits dans ce document.

Les références à des sites Web non IBM sont fournies à titre d'information uniquement et n'impliquent en aucun cas une adhésion aux données qu'ils contiennent. Les éléments figurant sur ces sites Web ne font pas partie des éléments du présent produit IBM et l'utilisation de ces sites relève de votre seule responsabilité.

IBM pourra utiliser ou diffuser, de toute manière qu'elle jugera appropriée et sans aucune obligation de sa part, tout ou partie des informations qui lui seront fournies.

Les licenciés souhaitant obtenir des informations permettant : (i) l'échange des données entre des logiciels créés de façon indépendante et d'autres logiciels (dont celui-ci), et (ii) l'utilisation mutuelle des données ainsi échangées, doivent adresser leur demande à :

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

Ces informations peuvent être soumises à des conditions particulières, prévoyant notamment le paiement d'une redevance.

Le logiciel sous licence décrit dans ce document et tous les éléments sous licence disponibles s'y rapportant sont fournis par IBM conformément aux dispositions de l'ICA, des Conditions internationales d'utilisation des logiciels IBM ou de tout autre accord équivalent.

Les données de performance indiquées dans ce document ont été déterminées dans un environnement contrôlé. Par conséquent, les résultats peuvent varier de manière significative selon l'environnement d'exploitation utilisé. Certaines mesures évaluées sur des systèmes en cours de développement ne sont pas garanties sur tous les systèmes disponibles. En outre, elles peuvent résulter d'extrapolations. Les résultats peuvent donc varier. Il incombe aux utilisateurs de ce document de vérifier si ces données sont applicables à leur environnement d'exploitation.

Les informations concernant des produits non IBM ont été obtenues auprès des fournisseurs de ces produits, par l'intermédiaire d'annonces publiques ou via d'autres sources disponibles. IBM n'a pas testé ces produits et ne peut confirmer l'exactitude de leurs performances ni leur compatibilité. Elle ne peut recevoir aucune réclamation concernant des produits non IBM. Toute question concernant les performances de produits non IBM doit être adressée aux fournisseurs de ces produits.

Le présent document peut contenir des exemples de données et de rapports utilisés couramment dans l'environnement professionnel. Ces exemples mentionnent des noms fictifs de personnes, de sociétés, de marques ou de produits à des fins illustratives ou explicatives uniquement. Toute ressemblance avec des noms de personnes, de sociétés ou des données réelles serait purement fortuite.

Licence sur les droits d'auteur

Le présent logiciel contient des exemples de programmes d'application en langage source destinés à illustrer les techniques de programmation sur différentes plateformes d'exploitation. Vous avez le droit de copier, de modifier et de distribuer ces exemples de programmes sous quelque forme que ce soit et sans paiement d'aucune redevance à IBM, à des fins de développement, d'utilisation, de vente ou de distribution de programmes d'application conformes aux interfaces de programmation des plateformes pour lesquels ils ont été écrits ou aux interfaces de programmation IBM. Ces exemples de programmes n'ont pas été rigoureusement testés dans toutes les conditions. Par conséquent, IBM ne peut garantir expressément ou implicitement la fiabilité, la maintenabilité ou le fonctionnement de ces programmes.

Toute copie totale ou partielle de ces programmes exemples et des oeuvres qui en sont dérivées doit comprendre une notice de copyright, libellée comme suit :

Si vous visualisez ces informations en ligne, il se peut que les photographies et illustrations en couleur n'apparaissent pas à l'écran.

Marques

IBM, le logo IBM et ibm.com sont des marques d'International Business Machines Corp. dans de nombreux pays. Les autres noms de produits et de services peuvent appartenir à IBM ou à des tiers. La liste actuelle des marques IBM est disponible sur le site Web Copyright and trademark information à l'adressewww.ibm.com/legal/copytrade.html

Microsoft et Windows sont des marques de Microsoft Corporation aux Etats-Unis et/ou dans certains autres pays.

Les autres noms de sociétés, de produits et de services peuvent appartenir à des tiers.