

Select Component Factory OLE Automation Interface User Guide



SELECT
COMPONENT FACTORY

© 2003 by Select Business Solutions Inc. All Rights Reserved.

Information in this document is subject to change without notice and does not represent a commitment on the part of Select to provide or continue providing said functionality. This document may not, in whole or part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent of Select Business Solutions Inc. Company names, data and other information appearing in examples are fictitious in nature unless otherwise designated.

The software described in this document is furnished under license or non-disclosure agreement and may be used or copied only in accordance with the terms and conditions of that agreement. It is against the law to copy the software on any medium except as specifically allowed in the license or non-disclosure agreement.

U.S. Government Restricted Rights

This software and documentation is protected by federal copyright law and is provided with RESTRICTED RIGHTS. Use, duplication or disclosure by the government is subject to restrictions as set forth in subdivision (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 252.227-7013 or subparagraphs (c)(1) and (2) of Commercial Computer Software - Restricted Rights at FAR 52.227-19, as applicable, and the limitations set forth in Select's standard license agreement for this software and documentation. Unpublished rights reserved under the copyright laws of the United States.

Select Component Factory, Select Component Manager, Select Component Architect, Select Component Portal, Reviewer for Select Enterprise, Reviewer for Select Component Architect, Select JSync, Select C#Sync, Select VBSync, Select C++ Sync, Select UDDIServer, CBDesign, Select Process Director, Select Process Director Plus, SMac and Select Perspective are trademarks of Select.

Because of the nature of the material, numerous hardware and software products are mentioned by their trade names in the publication. In most, if not all, cases these designations are claimed as trademarks by their respective companies. It is not the publisher's intent to use any of these names generically, and the reader is cautioned to investigate all claimed trademark rights before using any of these names other than to refer to the product described.

January 2003

Select Business Solutions, Inc

US, Canada and Mexico	1-888-4-SBS DIRECT (1-888-472-7347)
UK	+44-(0)-1242-638800
Germany	+49-(0)-(721)-98653-0
Benelux	+32-(0)-16-386010

For other countries contact details please call Select UK.

E-Mail infosbs@selectbs.com

Website www.selectbs.com

Contents

Introduction	4
OLE Automation Overview	5
Connection to the Automation Interface	7
Accessing an Individual Project/Repository	7
Using the Automation Interface	9
Accessing Properties	9
Traversing Roles	9
Item	10
Items	10
Add	11
Remove	11
Examples	11
Component Architect	12
Component Manager	12

1

Introduction

Select Component Manager and Select Component Architect provide automation services which allow access to the data stored in Component Architect Models and Component Manager Repositories.

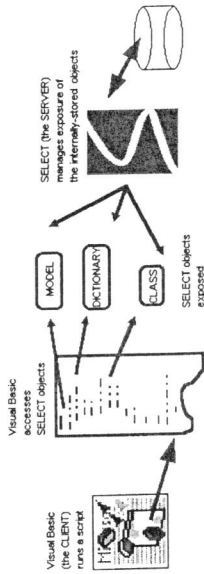
This guide provides an explanation of OLE automation and an outline of the interfaces provided by Component Architect and Component Manager.

2

OLE Automation Overview

OLE (Object Linking and Embedding) is a way to manipulate an application's objects from outside the application.

OLE Automation, which is sometimes referred to simply as Automation, is typically used to create applications that expose objects to programming tools and macro languages, to create and manipulate one application's objects from another application, or to create tools for accessing and manipulating objects.



OLE Automation is employed in Component Factory so that features of one application, such as Select Component Architect, can be used in another application, such as Select Document Generator. With OLE Automation, you can integrate features from both applications in a single procedure.

OLE automation works by obtaining a root object, for example a Select Component Architect model. Visual Basic programs can then interrogate the properties of the root object to discover its attributes, or other related objects. This process can be repeated on other objects, and in this way, the Select model can be explored as a network of connected objects.

Both Component Architect and Component Manager use a uniform interface for all objects, which simplifies access to exposed objects. The interface provides functionality to access and update the following:

Attributes for an object. For example: name, description.

Relationships/roles that the object has with other objects. For example:

Class Attribute.

Attributes of a relationship. For example Index Column as an Ascending Key.

You can also obtain help from the following sources:

Select Component Factory online help

3

Connection to the Automation Interface

The first step in using the automation interface is to make a connection. This is done by accessing the root object. Component Architect uses "Models" as the root object, all other objects are contained within a selected model. For Component Manager the root object is "Repositories".

In order to obtain access to the root object for Component Architect, the following code is used:

```
Dim oModels As Object  
Set oModels = CreateObject("SCA.Models")
```

For Component Manager the root object is accessed using:

```
Dim oRepositories as Object  
Set oRepositories = CreateObject("SCMAS.Repositories")
```

Note:

To connect to the interface provided with Select Enterprise (pre-Component Factory 4.0), use the following code:

```
Dim oModels As Object  
Set oModels = CreateObject("OWTE.Projects ")
```

Accessing an Individual Project/Repository

There are a number of roles that can be used from this point to access individual Models/Repositories. These are designed to allow a user access to either a specified model/repository or to the last used object instances used by the operator.

The roles available from the Component Architect's "Models" object are:

Project

Which can be used to specify a model using either model name or index

Active Project

Provides the last Model used either via the Automation Interface or Component Architect

Active Diagram

Provides the last Diagram used either via the Automation Interface or Component Architect

Active Dictionary Item

Provides the last Dictionary Item used either via the Automation Interface or Component Architect

From either the Dictionary Item or Diagram objects it is possible to navigate to the Model object, should that be necessary.

The roles available from the Component Manager's "Repositories" object are:

Repository

Can be used to specify a repository using either the repository name or index.

Active Repository

Provides the currently selected Repository object instance being used by Component Manager.

Active Object

Provides the currently selected Component Manager object instance used by Component Manager.

Further information on the properties and roles that are available is contained within [The Component Architect OLE Automation reference guide](#) and [The Component Manager SDK reference guide](#)

Using the Automation Interface

Now that the model/repository object is selected, it is possible to traverse roles to objects within the model/repository. It is also possible to access the properties available on those objects using a set of standard functions.

Accessing Properties

The two basic functions that are used for accessing properties are `get` and `set`, similar to a standard property in Visual Basic. The formats of these calls are as follows:

```
PropertyGet( Optional vType as Variant,
             Optional Index as Variant) As Variant

PropertySet( vType as Variant,
             Optional Index as Variant,
             Optional Data as Variant)
```

Here is an example of the code required for Component Architect:

```
Dim sName as String
sName = oAttribute.Property("Name")
oAttribute.Property("Name") = sName
```

The following is an example of the code required for Component Manager:

```
Dim sName as String
sName = oInterfaceSpec.Property("Name")
```

Note:

For Component Factory 4.2 the Component Manager Automation Services provide Read-Only access to a Repository so properties cannot be set.

Traversing Roles

There are four standard functions that can be used to traverse a role. For a list of the roles available see [The Component Architect OLE Automation reference guide](#) and [The Component Manager SDK reference guide](#).

Item

This function enables the user to retrieve a specific instance of an object via a role. This can be done by name.

Using the Component Architect interface obtaining the third Attribute called "Attr3" from a Class could be done using either of the following lines of code:

```
Dim oAttribute as Object
Set oAttribute = oClass.Item("Attribute", "Attr3")
```

Component Manager Automation Services provides the same type of interface.

```
Dim oCompSpec as Object
Set oCompSpec = oCatalog.Item("ComponentSpec", "CS1")
```

Items

Similar to "Item" this function returns a collection of the object instances available down the role specified.

Using the Component Architect interface to retrieve all of the Operations on a Class is done as follows:

```
Dim oOperations as Object
Set oOperations = oClass.Items("Operation")
```

Once the collection has been obtained then it is possible to loop through the items in the collection using the "For Each" statement. For example:

```
Dim oOperations as Object
Dim oOperation as Object
Set oOperations = oClass.Items("Operation")
For each oOperation in oOperations
    Debug.Print oOperation.Property("Name")
Next
```

The Component Manager interface operates in a similar manner. To retrieve all Interface Specifications in a Component Specification:

```
Dim oIntSpecs as Object
Set oIntSpecs = oCompSpec.Items("Interface Spec")
```

Again, this collection can be iterated through and each object examined in turn:

```
Dim oIntSpecs as Object
Dim oIntSpec as Object
Set oIntSpecs = oCompSpec.Items("Interface Spec")
For each oIntSpec in oIntSpecs
    Debug.Print oIntSpec.Property("Name")
Next
```

The Component Manager interface also has some sorting facilities for the Items collection. The following commands are available:

SORT: <-property>

Will sort the collection alphabetically on the value of <-property> in ascending order

SORTD: <-property>

Will sort the collection alphabetically on the value of <-property> in a descending order

SORTN:<property>

Will sort the collection numerically on the value of <property> in an ascending order.

SORTND:<property>

Will sort the collection numerically on the value of <property> in a descending order.

For example to retrieve all interface specifications sorted in alphabetical, ascending order:

```
Dim oIntSpecs as Object
Set oIntSpecs = oCompSpec.Items("InterfaceSpec", "SORT:Name")
```

For example to retrieve all components sorted in numerical, descending order:

```
Dim oROIComps as Object
Set oROIComps = oCatalog.Items("Component", "SORTND:ROI")
```

Add

Add may be used to link two existing objects or to create a new object linked to the first. For example in Component Architect, adding an Operation to a Class down the role Class to Operation will create an Operation like so.

```
Dim oOperation as Object
Set oOperation = oClass.add("Operation")
```

This would create an operation with a default name of "Operation". It is possible to specify a name at the time of creation by adding the name as a second parameter:

```
Dim oOperation as Object
Set oOperation = oClass.add("Operation", "Op1")
```

If the objects already existed they could be linked by:

```
oClass.add "Operation", oOperation
```

This links the Class to the Operation down the role "Operation"

Note:

Add functionality is not available via the Component Manager Automation services in the 4.2 release.

Remove

Like "Add" this function has two ways of operating. The first breaks the link between two specific objects instances and the second removes the objects of a specified type from an object instance.

In Component Architect to disconnect two different objects:

```
oClass.remove "Attribute", oAttribute3
```

The object can also be removed by specifying the name or the index of the object instead of providing the function with the object itself. For example:

```
oClass.remove "Attribute", "Attribute3"
```

or

```
oClass.remove "Attribute", 3
```

To remove all objects of a specified type, for example if it was necessary to remove all attributes from a class, the code would look like:

```
oClass.remove "Attribute"
```

Note:

Remove functionality is not available via the Component Manager Automation Services in the 4.2 release.

Examples

Component Architect

The following example code prints out the fully scoped name of each of the public classes contained within "Package1".

```
Dim oModels as Object
Dim oModel as Object
Dim oDictionary as Object
Dim oPackage as Object
Dim oClasses as Object
Dim oClass as Object
Set oModels = CreateObject("SCM.Models")
Set oModel = oModels.item("Active Project")
Set oDictionary = oModel.item("Dictionary")
Set oPackage = oDictionary.item("Scoped Package", "Package1")
Set oClasses = oPackage.items("Class")
For Each oClass in oClasses
  If (oClass.Property("Access") = "Public") then
    Debug.Print oClass.Property("Full Scoped Name")
  endif
Next
```

Component Manager

The following example code prints out the name of each of the Component Specs in a specified Catalog that have an Return On Investment that is higher than their "Reuse Count".

```
Dim oRepositories as Object
Dim oRepository as Object
Dim oCatalog as Object
Dim oCompSpecs as Object
Dim oCompSpec as Object
Set oRepositories = CreateObject("SCMAS.Repositories")
Set oRepository = oRepositories.item("Active Repository")
Set oCatalog = oRepository.item("CatalogItem", "New Catalog")
Set oCompSpecs = oCatalog.items("Component Spec", "SORTN:ReuseCount")
For Each oCompSpec in oCompSpecs
  If (oCompSpec.Property("ROI") > oCompSpec.Property("ReuseCount")) then
    Debug.Print oCompSpec.Property("Name")
  Endif
Next
```