Visual Basic 4.0 Technical Articles

Mapping the Schedule+ OLE Automation Server: Programming Model

Ken Lassesen
Microsoft Developer Network Technology Group

December 14, 1995

Abstract

This article consists primarily of a graphic map of a programming model of the Microsoft® Schedule+ OLE Automation server, showing its properties, methods, and child objects. The map displays the members corresponding to the Schedule+ graphical user interface (GUI) and is the first in a series of extended maps describing different views of the Schedule+ server. Any Visual Basic®-based language (Access Basic, Visual Basic, and Visual Basic for Applications) and Visual C++ ® can access the Schedule+ OLE Automation server.

The Microsoft Schedule+ OLE Automation Server Programming Model

The map of the programming model of the Microsoft® Schedule+ OLE Automation server displays the members corresponding to the Schedule+ graphical user interface (GUI). The Schedule+ OLE Automation server can be viewed in three different ways. The view corresponding to the GUI is described in this article; the other views are described in my adjunct articles, "Mapping the Schedule+ 7.0 Object Library: SPL 7.0" and "Mapping the Schedule+ OLE Automation Server: Internal Objects," which are available in the MSDN Library.

Schedule+ is OLE Automation-centric and contains many members that are not accessible or visible from the GUI. This programming model represents a subset of the Schedule+ OLE Automation server members, a subset that corresponds to the GUI for Schedule+. Although a type library of this view could be created, such a type library would limit the developer psychologically and technically to this subset alone. The map included in this article represents a traditional OLE Automation server view of Schedule+ that conceals many powerful features discussed in my other articles.

Figure 1 shows the relationship between objects as described by the Microsoft Schedule+ Programmer's Guide in the Microsoft Exchange Server Software Development Kit (SDK). A map is a good learning aid and a quick reference when developing an application. After producing a map, I found that it was easy to work with Schedule+: I just post the three appropriate extended maps for Schedule+ on my wall for quick reference, which is a lot faster than clicking topics in a Help file, one by one, to discover their properties and methods.

Because Figure 1 is difficult to read online, I have included the two most common graphics formats—encapsulated PostScript™ (.EPS) and Microsoft Windows® metafile (.WMF)—as well as a copy of my original Shapeware® Visio™ version 4.0 file (.VSD). The first two formats can be printed across multiple pages using any of the commercial graphics applications (such as Adobe™ PageMaker™, CorelDRAW®, or Microsoft Publisher) or using Microsoft Excel. The original Visio file is included for those who have a copy of Visio and want to modify the diagram easily.

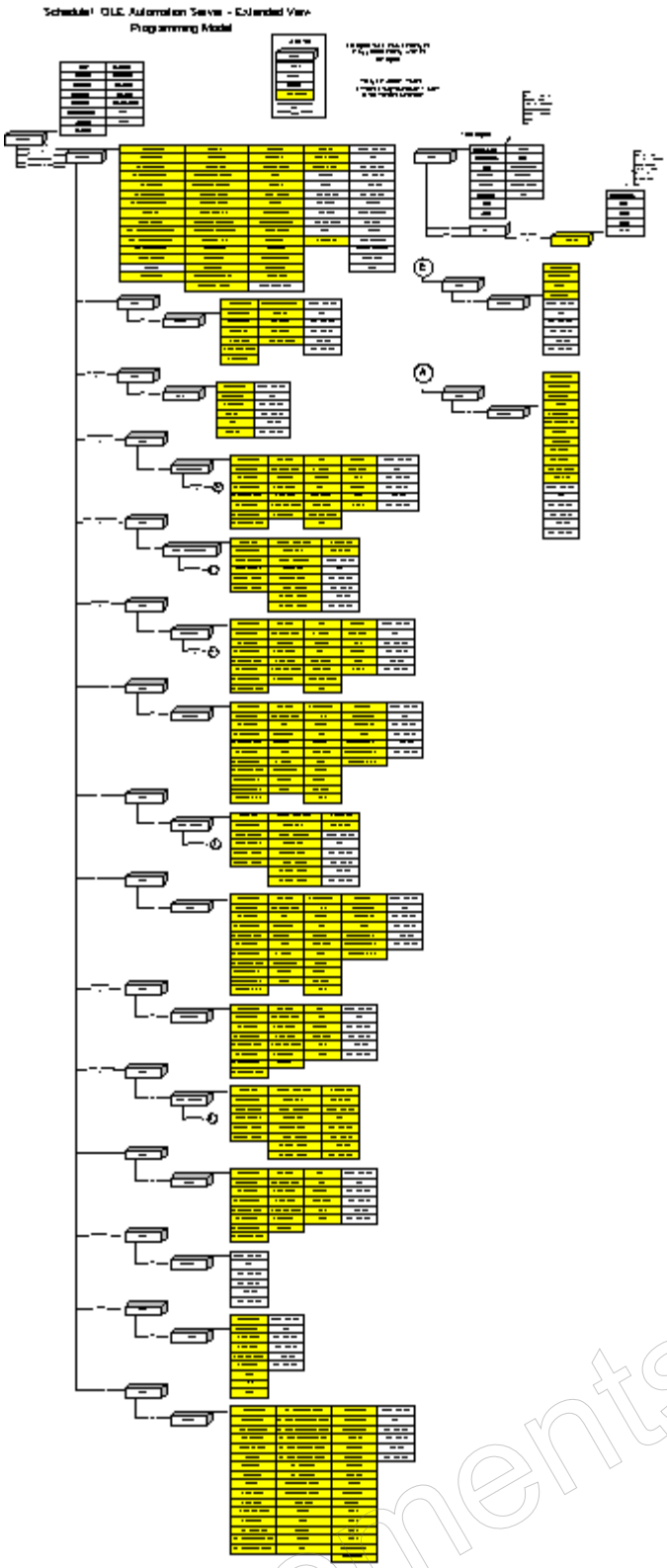

Figure 1. The programming model of the Microsoft Schedule+ OLE Automation server

Some characteristics of the Schedule+ OLE Automation server are different from common OLE Automation servers. Furthermore, the Schedule+ documentation uses terms in a different way from the documentation of other OLE Automation servers. For example, Schedule+ has an oxymoronic method called Properties. In the programming model, the following characteristics are significant.

- Most of the properties are Property objects. This is not intuitive, so a simple example may better explain this concept. For most OLE Automation servers, you will get a standard data type when you apply TypeName to a property, as shown below:

```
Set X = new stdtype.Stdfont
? TypeName (X.Size)
Currency
```

  With Schedule+, you do not get a standard data type; instead, you get a Property object whose default property is its Value property, as shown below:

```
? Typename(Item.Billing)
Property
? Typename(Item.Billing.Value)
String
```

- Some properties are not Property objects but standard data types. These properties make up the properties part of the Property, Table, Application, or Schedule objects. The extended map indicates which properties are Property objects and which properties are standard data types.

```
? Typename(Schedule.CanUndo)
Boolean
```

- Some properties can be either a Property object or a standard data type, depending on the context. See the extended map for clarification.

```
? TypeName(Item.ChangeNumber)
Property
? TypeName(Item.ChangeNumber.ChangeNumber)
Double
```

- The Properties method (without an argument) returns a long number that is the count of not empty Property objects (that is, IsEmpty returns False). Properties that are standard data types are not included in the Property objects returned by the Properties method (with an argument). For example, if Schedule.Properties returns 10, then Schedule actually has 19 properties: 10 Property objects and 9 properties that are standard data types.
- A Property object Value property is reported as Empty by the TypeName function if no value has been assigned. You cannot use Class of the Property object to determine which type of value can be assigned, because it is set to 255, signifying no value.

```
? IsEmpty(Item.CountryHome), _
  IsEmpty(Item.CountryHome.Value), _
  Item.CountryHome.Class
True          True          255
Item.CountryHome="NeverNeverLand"
? IsEmpty(Item.CountryHome), _
```

```
  IsEmpty(Item.CountryHome.Value), _
  Item.CountryHome.Class
False         False          8
```

- The GetProperty and GetProperties methods do not return Property objects; rather, they return the Value properties of the Property object in a two-dimensional variant array, as shown below:

```
? Typename(Item.GetProperties("CityLand"))
Variant()
? propData = Item.GetProperties("CityLand")
? UBound(propData,1),UBound(propData,2)
 0             0
```

If the Property object is empty, the value returned is "Error 438."

```
? propData(Xq(0,0)), propData(0,0)
Error         Error 438
```

- The GetProperty and GetProperties methods do not distinguish between an empty Property object, a standard data type property, and a nonexistent property.

```
PropData=Schedule.GetProperties("CanUndo")
? PropData(0,0), Schedule.CanUndo
Error 438     True

PropData=Schedule.GetProperties("SoundLevel")
? PropData(0,0), IsEmpty(Schedule.SoundLevel)
Error 438     True

PropData=Schedule.GetProperties("BillGates"):
? PropData(0,0)
Error 438
```

A good understanding of the Property object, the Properties method, and object properties is essential for doing OLE Automation with Schedule+. For additional information and caveats, read my adjunct articles, "Mapping the Schedule+ 7.0 Object Library: SPL 7.0" and "Mapping the Schedule+ OLE Automation Server: Internal Objects."

Object Definitions

The objects and collections in Figure 1 are defined in the table below. The objects are listed in the same sequence as they appear in the map.

Table 1. Object Definitions

Object	Definition
Application object	This object reports information about the single-document interface (SDI) application.
Schedule object	This object represents one schedule in the Application object. This is also an Item object.
Table object	This object represents a table of Item records that store information. Although it resembles a collection, it is not a collection object.
Item object	This object represents an object that may be a Property object or a Table object.
Property object	This object represents information about a characteristic of an object. There may be multiple values stored in one Property object. (See Count to determine the number.)

Access object	This object stores access control information for a user's schedule file. This object is an Item object.
Alarm object	This object represents a single alarm in the AlarmsToRing table. This object is an Item object.
SingleAppointment object	This object represents an appointment that occurs only once. This object is an Item object.
RecurringAppointment object	This object represents recurring appointments recorded as a pattern described by the properties. This object is an Item object.
Appointment object	This object represents an appointment. This object is an Item object.
SingleTask object	This object represents a task that occurs only once. This object is an Item object.
RecurringTask object	This object represents recurring tasks recorded as a pattern described by the properties. This object is an Item object.
Task object	This object represents a task. This object is an Item object.
SingleEvent object	This object represents an event that occurs only once. This object is an Item object.
RecurringEvent object	This object represents recurring events recorded as a pattern described by the properties. This object is an Item object.
Event object	This object represents an event. This object is an Item object.
DeletedItem object	This object contains the path to a deleted object. This object is an Item object.
Project object	This object represents a project. This object is an Item object.
Contact object	This object represents a single contact. This object is an Item object.

Bibliography

Lassesen, Ken. "An Extended Introduction to Schedule+ OLE Automation Programming." (MSDN Library, Technical Articles)

Lassesen, Ken. "Mapping the Schedule+ OLE Automation Server: Internal Objects." (MSDN Library, Technical Articles)

Lassesen, Ken. "Mapping the Schedule+ 7.0 Object Library: SPL 7.0." (MSDN Library, Technical Articles)

Lassesen, Ken. "Schedule+ OLE Automation." Developer Network News, January/February 1996, Number 1. (MSDN Library, Periodicals, Microsoft Developer Network News)

Lassesen, Ken. "Using Microsoft OLE Automation Servers to Develop Solutions." (MSDN Library, Technical Articles)

Microsoft Exchange Server SDK. Microsoft Schedule+ Programmer's Guide. (MSDN Library)