Actuate One™

One Design
One Server
One User Experience

Programming with Actuate Foundation Classes

Contains information proprietary to:
Actuate Corporation, 2207 Bridgepointe Parkway, San Mateo, CA 94404

www.actuate.com
www.birt-exchange.com

Document No. 111021-2-130321 October 14, 2011

Contents

Part 2
Actuate Foundation Class Reference

About *Programming with Actuate Foundation Classes*

Programming with Actuate Foundation Classes provides information about using the Actuate Foundation Classes, their variables, properties, and methods.

Programming with Actuate Foundation Classes includes the following chapters:

- *About Programming with Actuate Foundation Classes.* This chapter provides an overview of this guide.
- *Part 1. Working with Actuate Foundation Classes.* This part describes and provides information about Actuate Foundation Classes, working with the inheritance hierarchy, and document generation.
- *Chapter 1. Understanding Actuate Foundation Classes.* This chapter describes the Actuate Foundation Class architecture and provides an overview of Actuate Foundation Classes by functional category.
- *Chapter 2. Working with a class.* This chapter provides information about declaring and working with Actuate Foundation Classes.
- *Chapter 3. Working with an object.* This chapter provides information about working with objects and object reference variables.
- *Chapter 4. Actuate Foundation Class library.* This chapter shows an overview of the inheritance hierarchy and a summary of classes and methods.
- *Chapter 5. Understanding report generation.* This chapter provides information about document generation and the content-creation process.
- *Part 2. Actuate Foundation Class Reference.* This part provides lists of the AFC data types and the AFC classes.
- *Chapter 6. AFC data types.* This chapter lists the AFC data types.

- *Chapter 7. AFC classes.* This chapter contains an alphabetical listing of the classes. Each class description includes a variables and properties summary followed by an alphabetical listing of the methods.

Part One

Working with Actuate Foundation Classes

Chapter

1

Understanding Actuate Foundation Classes

This chapter contains the following topics:

- About the Actuate Foundation Class architecture
- Understanding the AFC by functional category

About the Actuate Foundation Class architecture

The Actuate Foundation Class (AFC) library, afc.rol, contains the classes that form the framework on which report developers build Actuate reports. Each foundation class serves a distinct purpose, such as creating a data source connection, building a currency control, or creating a page layout.

The foundation classes are written in Actuate Basic, an object-oriented programming language modeled after Microsoft Visual Basic Version 3. Actuate Basic is the programming language used to access the AFC library. For information about the data types, statements, and functions that Actuate Basic uses for report development tasks, see *Programming with Actuate Basic.* Actuate Basic is an interpreted language. The set of processes that interprets the language is called the Factory. For more information about the Factory, see Chapter 5, "Understanding report generation."

About the core protocol

Despite differences in their functionality, the foundation classes share a core protocol, which provides the logic for the tasks that are common to all report components. This core protocol gives the foundation classes a uniformity that makes the AFC library easy to understand and use. The core protocol consists of methods that come from two foundation classes. Accompaniment is the base class from which all other classes in the library descend. AcReportComponent is the base class for all reports, sections, frames, controls, page lists, flows, and pages. Figure 1-1 shows the methods that compose the core protocol.

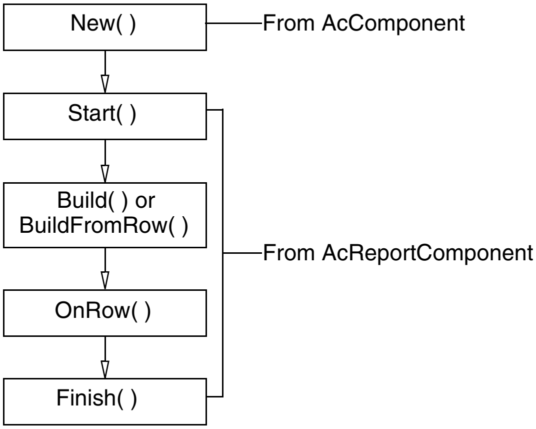


Figure 1-1 AFC core protocol

Table 1-1 describes the core methods. For more information about these methods, see "AcComponent" and "AcReportComponent" in Chapter 7, "AFC classes."

Table 1-1 AFC core methods

Core method	Description
New( )	Provides the logic for constructing a new object.
Start( )	Provides the logic for preparing an object for the build process.

Build( )
and
BuildFromRow( )
	Provide the logic for creating the contents of container objects such as reports or frames. Use Build( ) for components that do not process data rows. Use BuildFromRow( ) for components that process data rows.

OnRow( )	Assigns to a data control the value from the expression in the control's ValueExp property.
Finish( )	Provides the logic for completing an object.

About class protocols

Each class adapts the core protocol to meet the needs of the class. The higher a class is in the AFC hierarchy, the more general is its class protocol. You build the features that differentiate classes on top of the core protocol. Each successive generation of classes contains increasingly specialized versions of the core protocol. The class protocol builds on methods from the core protocol, adding methods that address the main task of the class and dropping core methods the class protocol does not need.

Figure 1-2 shows the protocols for three classes, AcSection, AcDataAdapter, and AcFlow. Although these classes support very different functionality, each class protocol derives from the core protocol.

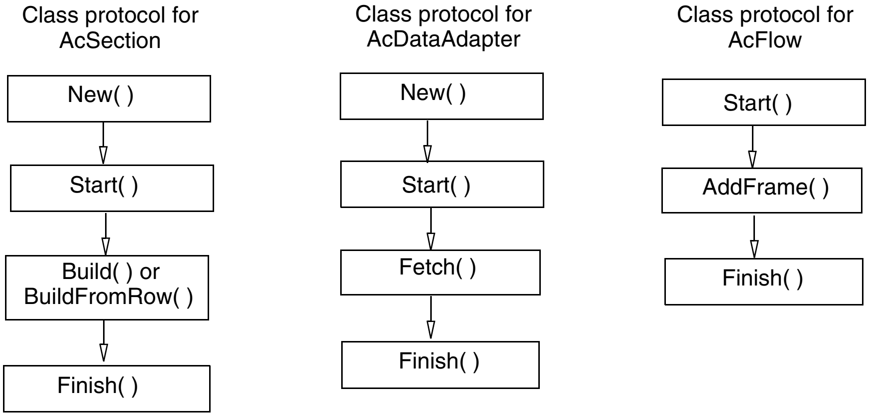

Class protocol for AcSection
New( )
Start( )
Build( ) or BuildFromRow( )
Finish( )
Class protocol for AcDataAdapter
New( )
Start( )
Fetch( )
Finish( )
Class protocol for AcFlow
Start( )
AddFrame( )
Finish( )

Figure 1-2 Protocols for AcSection, AcDataAdapter, and AcFlow

e.Report Designer Professional supports the creation of reports without the need to understand the class protocols. To create custom components that require programming or to change or extend the AFC architecture, you must understand class protocols.

About abstract base classes

An abstract base class defines the protocol governing the behavior of its subclasses. Subclasses refine and build on this protocol. Many methods in an abstract base class are empty. By subclassing a base class, you add the necessary implementation details.

Never instantiate an abstract base class. In general, derive a class only from a subclass of an abstract base class. If you subclass directly from an abstract class, you must add functionality to your subclass. Most often, however, the functionality you want to add already exists in a subclass.

About concrete classes

A concrete class defines the specific implementations of an abstract base class. A concrete class inherits and extends behavior from an abstract base class. Components in e.Report Designer Professional's Toolbox correspond to concrete classes.

You can instantiate a concrete class. You also can subclass a concrete class to modify or extend the functionality of the original class.

Understanding the AFC by functional category

The Actuate Foundation Classes are divided into the following functional categories:

- Report structure classes
- Page layout classes
- Control classes
- Connection classes
- Collection classes
- Data stream classes
- Excel classes
- A visitor class

The following sections provide an overview of the abstract and concrete classes in each category, their purpose, and their position in the class hierarchy.

About report structure classes

Report structure classes form the backbone of a report. They define the general structural characteristics of objects, the logic for creating objects, and the way objects work together.

When you create a new report in e.Report Designer Professional, you create a subclass of AcReport. AcReport, in turn, is a subclass of AcReportComponent, as shown in Figure 1-3.

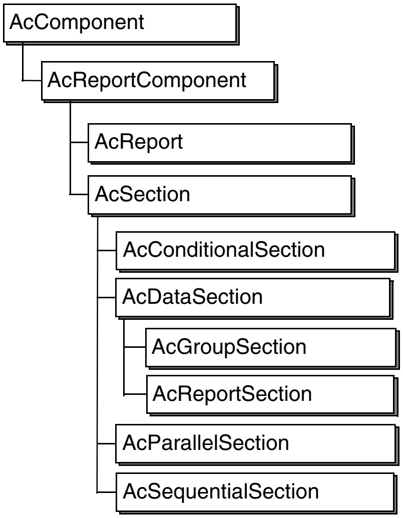


Figure 1-3 Class hierarchy for report structure classes

About report structure abstract base classes

AcComponent is the root class from which all other reporting objects descend. AcReportComponent is the base class for all components of a report. AcReport is the container for all other objects in a report. AcSection is the abstract base class for all sections. AcDataSection is the abstract base class that defines the logic a section uses to process a group of data rows. Do not derive from these classes.

About report structure concrete classes

Use the AcConditionalSection, AcGroupSection, AcParallelSection, AcReportSection, and AcSequentialSection concrete classes to organize data in a report.

About page layout classes

The page layout classes manage the creation and display of a report page. The classes that control page layout are shown in Figure 1-4.

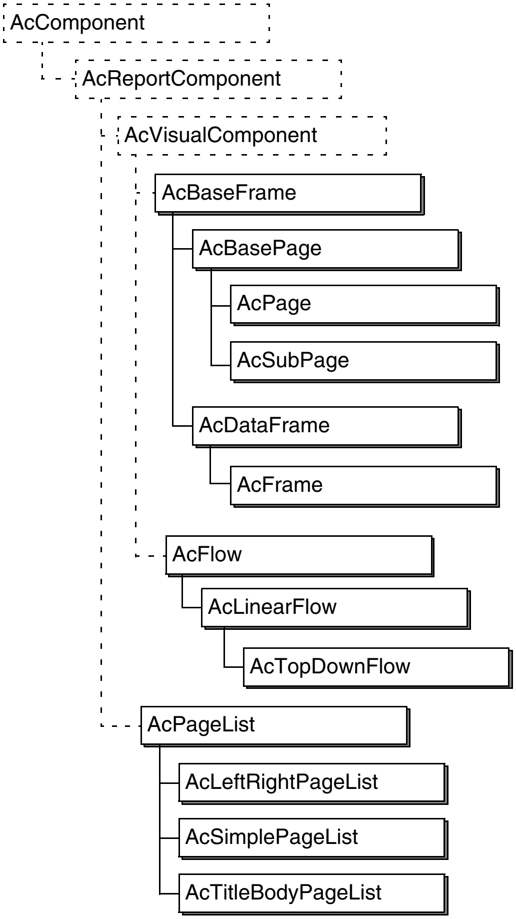


Figure 1-4 Classes that control page layout

About page layout abstract base classes

AcBaseFrame is the abstract base class that defines the core logic common to pages and frames. AcBasePage is the abstract base class that defines the logic for instantiating the contents of pages. AcDataFrame is the abstract base class that defines the logic for how frames work with data rows. AcFlow is the abstract base class that defines the logic for placing frames in a flow, the printable area of a page. AcPageList is an abstract base class that instantiates and holds the pages for a report. Do not derive from these classes.

About page layout concrete classes

Use AcSubPage and AcPage to design page styles. Use AcTopDownFlow to determine the placement of report objects on the page. The AcSimplePageList, AcLeftRightPageList, and AcTitleBodyPageList classes represent specific page designs.

AcFrame is a container for controls. In a report design, a frame and its contents are typically associated with one or more data rows. For example, if a data row contains name, address, and telephone data, the report design includes a frame that contains three data controls for the data. In e.Report Designer Professional, each time you drag a frame from a toolbox and drop it in the report design, you instantiate a subclass of AcFrame.

About control classes

Control classes include data controls, cross tabulations, charts, and static graphical objects, as shown in Figure 1-5.

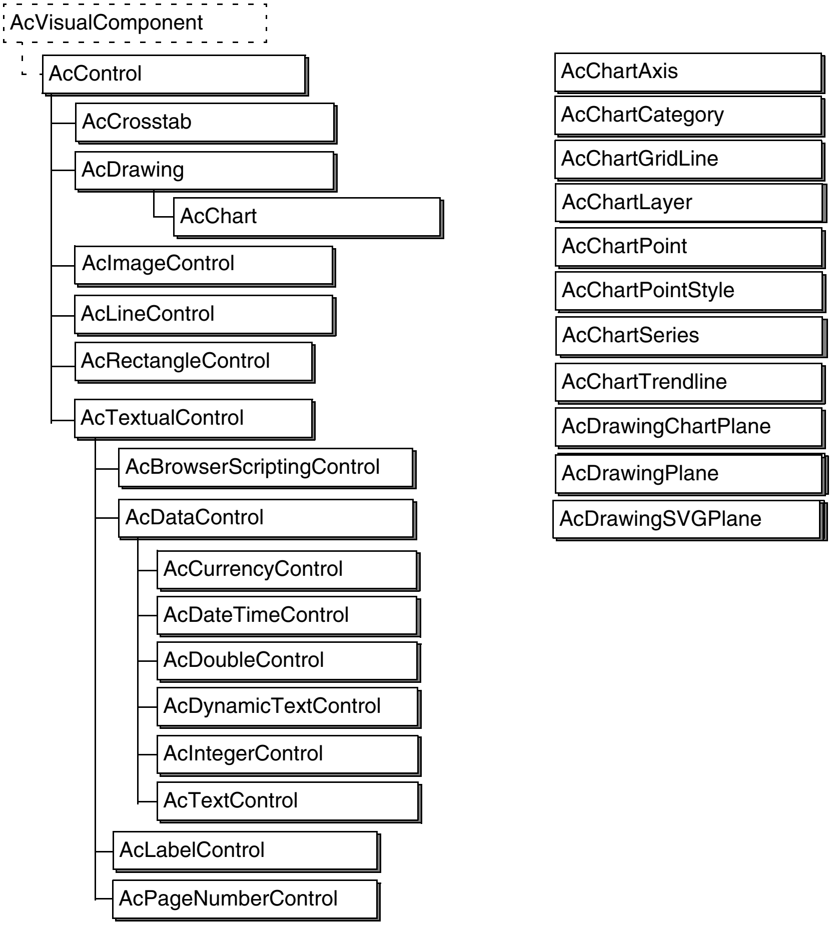


Figure 1-5 Control classes

About control abstract base classes

AcControl is the abstract base class that defines the core characteristics of all controls. AcCrosstab is the class you use in a report design to display data in rows and columns. AcDataControl is the base class for controls that display data from data rows. Derive from these classes only to create a custom control.

About control concrete classes

Use the AcCurrencyControl, AcDateTimeControl, AcDoubleControl, AcDynamicTextControl, AcIntegerControl, and AcTextControl classes to display various types of data from a data row. Use AcLabelControl to display static text. Use AcPageNumberControl to display a page number.

UseAcImageControl, AcLineControl, and AcRectangleControl as drawing elements that give a report visual interest.

Use AcChart, AcChartAxis, AcChartCategory, AcChartLayer, AcChartPoint, AcChartPointStyle, and AcChartSeries to display data in various standard chart formats, such as pie charts and bar charts. Use AcDrawingChartPlane, AcDrawingPlane, and AcDrawingSVGPlane to work with chart drawings.

Use AcDrawing to create a custom illustration using SVG code.

Use AcBrowserScriptingControl to add web functionality to a report. AcBrowserScriptingControl appears only in a DHTML report.

About connection classes

Connection classes provide communication links for an Actuate report. Figure 1-6 shows the types of connections a report can use. Each connection type is a subclass of AcDBConnection.

About connection abstract base classes

AcConnection is the abstract base class that defines the core protocol for all connection components. AcDBConnection is the base class that defines the basic protocol for establishing database connections. Do not derive directly from these classes.

About connection concrete classes

To connect to a data source, use the AcDB2Connection, AcMSSQLConnection, AcOdaConnection, AcODBCConnection, AcOracleConnection, and AcProgressSQL92Connection concrete classes.

AcDBStatement and AcDBCursor provide the Actuate Basic interface for working with a SQL statement or cursor. The AFC framework creates and uses instances of these classes when a report accesses a SQL database.

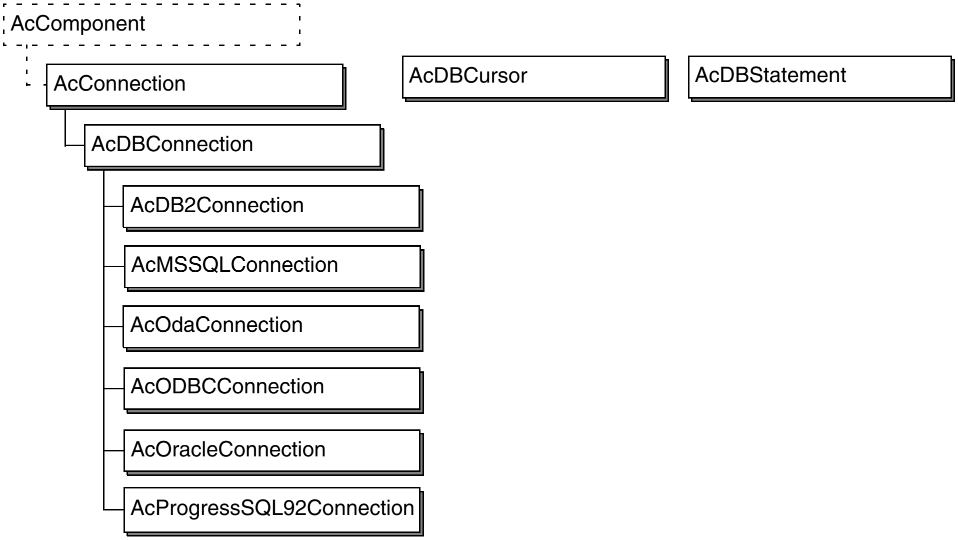


Figure 1-6 Connection classes

About collection classes

Collection classes, shown in Figure 1-7, define the way e.Report Designer Professional stores objects and accesses them in a linked list. For example, a frame uses lists to manage the controls within the frame. To work with report content in a list, such as controls in a frame or flows on a page, create a collection class and an iterator class to access the contents.

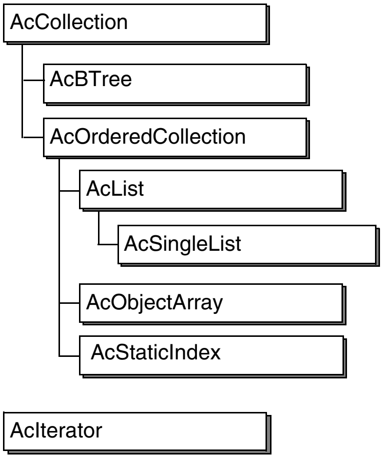


Figure 1-7 Collection classes

About collection abstract base classes

AcCollection is the abstract base class for the Actuate collection classes. AcOrderedCollection is the abstract base class for the Actuate ordered collection classes. AcList class is an abstract class that defines the list interface. AcIterator is the base class for all iterators.

About collection concrete classes

Use AcSingleList to process ordered lists, stacks, and queues. Use AcObjectArray to create a resizable array of objects. Use AcStaticIndex to implement a multi-layer, n-way tree to provide fast indexing into a large collection of data. Use AcBTree to create a list of objects sorted by one object's attributes.

About data stream classes

The data stream classes get and process data, create data rows, and send data rows to the report. Figure 1-8 shows the principal data stream classes.

About data stream abstract base classes

AcDataRow is the abstract base class for user-defined data rows.

AcDataAdapter is the abstract base class that defines the logic of classes that form a data stream. Do not derive from AcDataAdapter.

AcDataSource is a base class that defines how data sources retrieve data from an input source and create data rows. AcDatabaseSource is an abstract base class for data sources that retrieve data from databases. AcExternalDataSource is an abstract base class for generic data source objects that use a command to retrieve a single result set through an associated connection.

AcQuerySource is an abstract base class for query data sources. AcQuerySource uses a Select statement to retrieve data from a database.

AcDataFilter is the abstract base class for all data filter classes.

About data stream concrete classes

AcDataRowBuffer, AcDataRowSorter, AcSingleInputFilter, and AcMultipleInputFilter are data filters.

AcSqlQuerySource is a data source that you use to retrieve data from a SQL database. AcStoredProcedureSource creates a data source for a stored procedure that uses data rows. AcTextQuerySource creates a data source for a SELECT statement that a report developer provides.

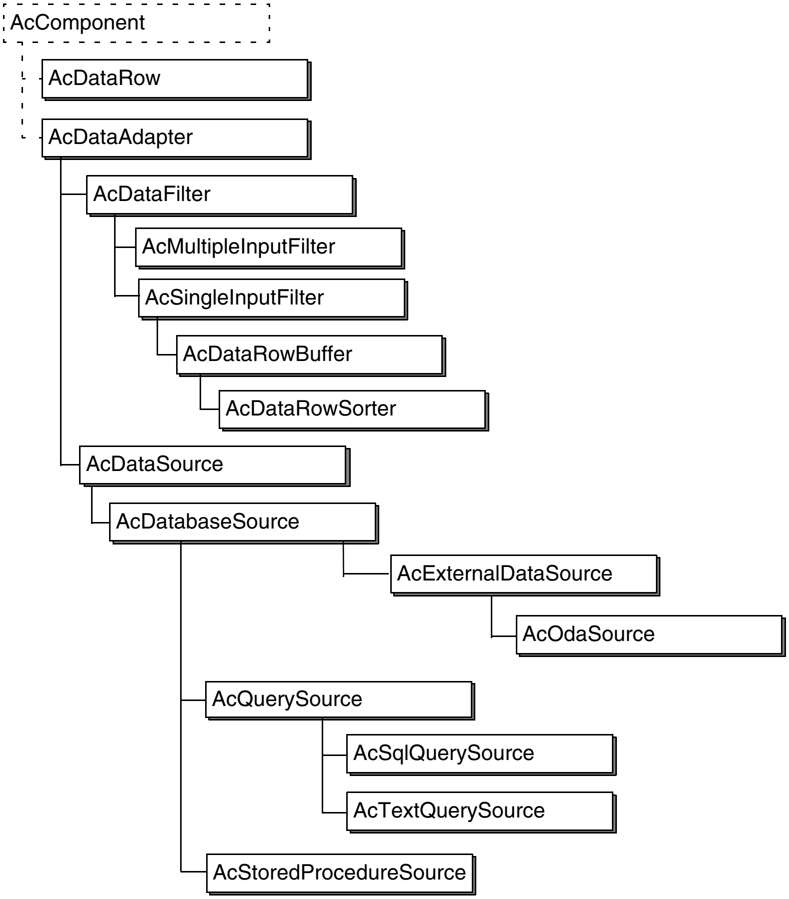


Figure 1-8 Data stream classes

About Excel classes

Actuate's Excel classes, shown in Figure 1-9, support creating and managing the Excel workbooks, worksheets, ranges, rows, columns, and cells.

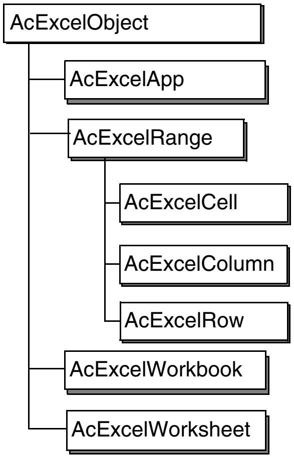


Figure 1-9 Excel classes

About Excel abstract base classes

AcExcelObject is the abstract base class from which all other Excel classes descend. AcExcelApp is the base class for all instances of classes you use to generate and work with Excel files. AcExcelRange class is the base class for AcExcelCell, AcExcelColumn, and AcExcelRow. Do not derive from these classes.

About Excel concrete classes

AcExcelCell, AcExcelColumn, AcExcelRow, AcExcelWorkbook, and AcExcelWorksheet are concrete classes for working with Excel files.

About the visitor class

Use AcVisitor to create a utility to visit and perform an action on a report object. AcVisitor provides the means to visit each type of report component. Figure 1-10 shows the visitor class.

Figure 1-10 The visitor class

Chapter

2

Working with a class

This chapter contains the following topics:

- About classes
- About class declaration
- Understanding class relationships
- Getting information about classes in a report
- Working with a class variable
- Working with a method

About classes

A class is a specification, or template, for creating an object in a report design. A class contains variables and methods that define the attributes and behavior for objects of the class. Report components, such as report sections, frames, and controls, are instances of classes. This chapter introduces the concepts that you use to declare and work with Actuate Foundation Classes.

Actuate Foundation Classes are written in Actuate Basic. You instantiate a class in Actuate Basic differently from Java or C++. To instantiate a declared class in Actuate Basic, use an object reference variable with a statement or function such as Set, NewInstance, or NewPersistentInstance. An object reference variable allocates memory for an object. You can instantiate the object using the Actuate Foundation Classes or your own code.

In e.Report Designer Professional, the design environment accomplishes many programming tasks for you, such as generating class declaration code for each component in a report design.

About class declaration

Actuate Basic code defines the structure and behavior of the Actuate Foundation Classes. e.Report Designer Professional creates Actuate Basic code for the classes that correspond to components of a report design. To write a custom class for a report design, declare the class using the Class statement. The Class statement uses the following syntax:

```
Class <subclass name> Subclass Of <superclass name>
   [<variable declarations>]
   [<nested class declarations>]
   [<method declarations>]
End Class
```

where

- <variable declarations> declare variables associated with the class.
- <nested class declarations> declare classes nested in the current class.
- <method declarations> consist of subroutines and methods associated with the class.

The following example shows a class declaration in an Actuate Basic source (.bas) file. e.Report Designer Professional generates this file when you compile a report object design (.rod) file. The example creates a class called ReportTitle. It is a subclass of AcFrame. The class defines the label and a line control that are located within the frame.

```
Class ReportTitle Subclass Of AcFrame

   Class LabelControl Subclass Of AcLabelControl
      Sub SetProperties( )
         Super::SetProperties( )
         Font.Bold = True
         Font.Color = Navy
         Font.Italic = True
         Font.Size = 18
         Position.X = 3551
         Position.Y = 120
         Size.Height = 459
         Size.Width = 2257
         Text = "Customers"
         TextPlacement.Horizontal = TextAlignCenter
      End Sub
   End Class

   Class LineControl Subclass Of AcLineControl
      Sub SetProperties( )
         Super::SetProperties( )
         EndPosition.X = 9360
         EndPosition.Y = 60
         LineStyle.Color = Navy
         LineStyle.Width = 60
         Position.X = 0
         Position.Y = 60
      End Sub
   End Class
End Class
```

Understanding class relationships

AFC classes co-exist to perform a variety of tasks. You must understand class relationships to:

- Create, modify, or delete a class, a variable, or a method.
- Refer to a class, a variable, or a method.
- Manage class modifications to avoid unexpected effects in related classes or classes that refer to the modified class.

Table 2-1 summarizes the principal concepts that govern class relationships.

Table 2-1 Terminology for expressing class relationships

Relationship	Description
Inheritance	A mechanism whereby one class is defined as a special case of a more general class. The special case is called a derived class or subclass. The general class is called a base or superclass.
Reference	A means of accessing an object directly from another object. A reference is not a subclass or a new instance of another class. For example, when ObjectA of ClassA refers to ObjectB of ClassB in code, ObjectA has access to the public components of ObjectB, including methods and variables.
Scope	Determines the visibility of classes, static variables, and methods and how you refer to those items in code.

About inheritance

In the AFC class hierarchy, inheritance supports maintaining a standard interface for a report. It also supports code reusability. The classes at the top of the hierarchy typically contain empty method declarations or methods with a few lines of general instructions. These methods enforce a protocol for creating a report. A class lower in the hierarchy adds implementation details to the higher-level method. When you derive a class from an Actuate Foundation Class, the subclass inherits the protocol.

The superclass serves as the baseline for the appearance and behavior of the subclass. Typically, a subclass augments or redefines the behavior and structure of its superclass. When you customize the subclass, the customization does not affect the superclass.

Figure 2-1 shows the AcComponent abstract base class and the principal subclasses that inherit from AcComponent.

About references

A method or object in one class can refer to a method or object in another class. When you refer to an object, you make that object available to the calling class. A reference is a link to another class, not a subclass or a new instance of another class. For example, in class AcExcelWorksheet, the following method returns a reference to an object of the class AcExcelRow:

```
Function GetRow( row As Integer ) As AcExcelRow
```

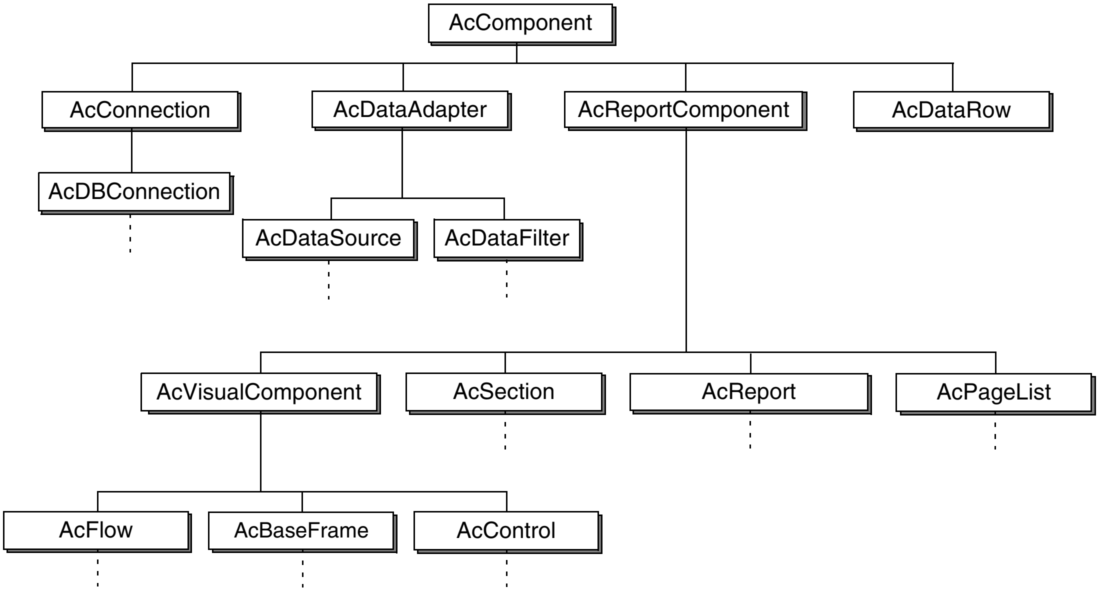


Figure 2-1 Principal subclasses that inherit from AcComponent

References support creating an object once and using it in multiple contexts. A calling class cannot modify the original class.

In e.Report Designer Professional, you create a reference when you drag a component from a library and drop it in a report design. If you change the component in the report design, you create a subclass of the component and the original component in the library does not change. Conversely, if you change the original component, all references to that component inherit the changes.

About scope

Scope defines the part of a report in which a symbol exists or is visible. A symbol is the name of a class, a method, a variable, or a constant. Scope determines how you access a class, how you create a reference to a class, when you instantiate a class, when you initialize a variable, and so on.

You can declare a class in either global or class scope. A class has global scope if you do not declare it within another class. A component in a library has global scope. A class has class scope if you declare it within another class. A class with class scope is called a nested class. For example, a control in a frame usually is a nested class, scoped to the frame that contains the control. You cannot move the base class into the scope of a nested class.

The following example shows the scope and inheritance of the SalesDetail class, which is a subclass of AcReport. In addition, the example shows two nested classes that have class scope, OfficeTitleFrame and CustomerTitleFrame. Both classes are subclasses of AcFrame.

```
Class SalesDetail Subclass Of AcReport
   …
   Class OfficeTitleFrame Subclass Of AcFrame
   …
   End Class

   Class CustomerTitleFrame Subclass Of AcFrame
   …
   End Class

End Class
```

Understanding class scope naming conventions

Using the scope-resolution operator (::), you can refer to any name of a class or static variable, even if it is not visible in the current scope, and build a path to the innermost scope. For example, the following class names refer to nested classes:

```
CustomerFrame::AddressControl
SalesRepFrame::AddressControl
```

This naming convention is similar to specifying a path in a URL using a slash (/). This convention uses the following rules:

- The class names in one scope are independent of class names in another scope. Just as you can have two files with the same name if they are in different directories, you can have two classes with the same name if they are in different scopes.
- To refer to a class in a different scope, use a qualified name. This convention is analogous to accessing a file in a different directory. For example, to write code for CustomerFrame that references AddressControl in SalesRepFrame, use the full name, SalesRepFrame::AddressControl.
- To refer to a class in the same scope, use only the class name. This convention is similar to specifying a file in the current directory. For example, if CustomerFrame contains two nested controls, CustomerNameControl and CustomerAddressControl, use the class name, CustomerAddressControl, to write code for a method in CustomerNameControl that refers to CustomerAddressControl. The qualified name is not necessary because both controls are in the same frame.

About the default scope of a class in a report design

In Actuate Basic, every class introduces its own default scope. Actuate Basic applies the following rules to set default class scope when you place a component in a report design:

- The report has global scope.
- All other classes except controls take the report's scope.
- A control takes the scope of its container, typically a frame.

Setting a default scope provides two key benefits:

- Simplified naming conventions

  Nesting a control within a frame supports managing control names. Because a control has class scope, it does not require a unique name. Actuate Basic supports an unlimited number of nesting levels.
- Reusability

  Nesting a class within the report object makes it possible to reuse the class in another report design without a naming conflict. For example, you can place a frame called CustomerFrame in a report design. In the same report, you can use another frame called CustomerFrame from a different report design or a library without changing the name of either frame. e.Report Designer Professional recognizes one frame as <Report1>::CustomerFrame and the other as <Report2>::CustomerFrame.

Table 2-2 summarizes the default class scope for several types of report classes.

Table 2-2 Default scopes for report classes

Type of class	**Default scope**
Report	Global
Section	Report
Page list	Report
Page	Report
Flow	Report
Connection	Report
Data source	Report
Data filter	Report
Frame	Report
Control	Frame class or page class

About the default scope of a class in a library

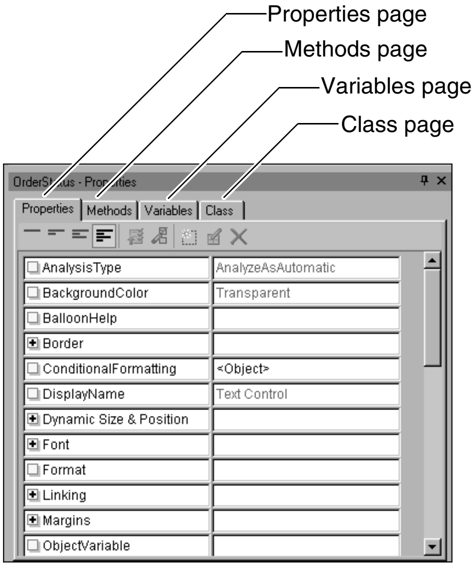
Because a class in a library is available for any report design, the class has global scope. If you publish a class that has report scope to a library, e.Report Designer Professional changes the scope to global scope in both the library and the report design.

Getting information about classes in a report

e.Report Designer Professional provides various ways to get information about a class. For example, you can use Project Browser to view all the classes available to a report. You also can use the Properties window to view information about a specific class. For more information about using these tools, see *Developing Reports using e.Report Designer Professional.*

Getting information about a specific class

To view information about a class, use the Properties window, which is shown in Figure 2-2.



□ AnalysisType	AnalyzeAsAutomatic
□ BackgroundColor	Transparent
□ BalloonHelp	
⊞ Border	
□ ConditionalFormatting	<Object>
□ DisplayName	Text Control
⊞ Dynamic Size & Position	
⊞ Font	
□ Format	
⊞ Linking	
⊞ Margins	
□ ObjectVariable	

Figure 2-2 Properties window

In the Properties window, you can access the following class information:

- Properties

  The Properties page displays properties of the class and their values. To view the properties of a class, click the class in the report design. The Properties page for the class appears. On this page, you can modify property settings and designate some properties as parameters.

  The properties that are visible on the page depend on the filtering option you select. You can view expert properties, only advanced properties, only the most commonly used properties, or only the properties you have overridden or defined.

- Methods

  The Methods page displays methods of the class. On this page, you can create new methods, edit or override methods, and delete methods to which you have access.

  The methods that are visible on the page depend on the filtering option you select. You can view the callable methods, only the overridable methods, only the most commonly used methods, or only the methods you have overridden or defined.

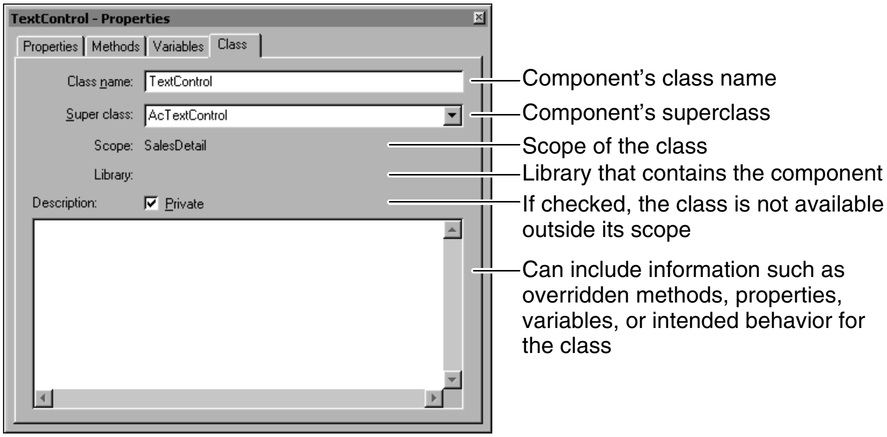
- Variables

  The Variables page displays the variables of the class. Using this page, you can create and edit variables, view the data type of a variable, filter variables, and delete variables.

  The variables that appear on the page depend on the selected filtering option. You can view local variables, local and public variables, or all variables.

- Class

  The Class page displays general information about the class, such as its superclass and scope, a path to the library that contains the class, a description of the class, and whether the class is public or private. Figure 2-3 shows the Class page.



Class name: TextControl

Super class: AcTextControl

Scope: SalesDetail

Library:

Description:

☑ Private

Component's class name

Component's superclass

Scope of the class

Library that contains the component

If checked, the class is not available outside its scope

Can include information such as overridden methods, properties, variables, or intended behavior for the class

Figure 2-3 Class page

The check box labeled Private indicates whether a class remains private, which means that the class is not available outside its current scope. A private class does not appear in the Libraries window. The Private check box applies only to classes with global scope.

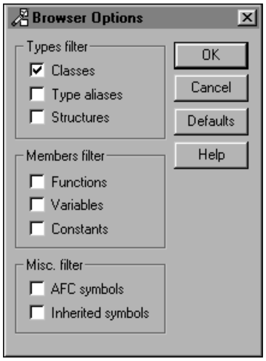
The default setting of a component you add to a frame, a page, or a section is private. The default setting of other classes is public.

Getting information about all classes in a report

For a list of all classes available to a report or to confirm a change you make using the Class page, open Project Browser.

How to view all classes available to a report

1. In e.Report Designer Professional, choose View➛Project Browser. Project appears.
2. Choose Filter. Browser Options appears, as shown in Figure 2-4.



Browser Options

Types filter
☑ Classes
□ Type aliases
□ Structures

Members filter
□ Functions
□ Variables
□ Constants

Misc. filter
□ AFC symbols
□ Inherited symbols

Figure 2-4 Browser Options dialog box

3. Select Classes. Choose OK. Project appears, as shown in Figure 2-5.

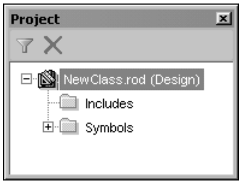
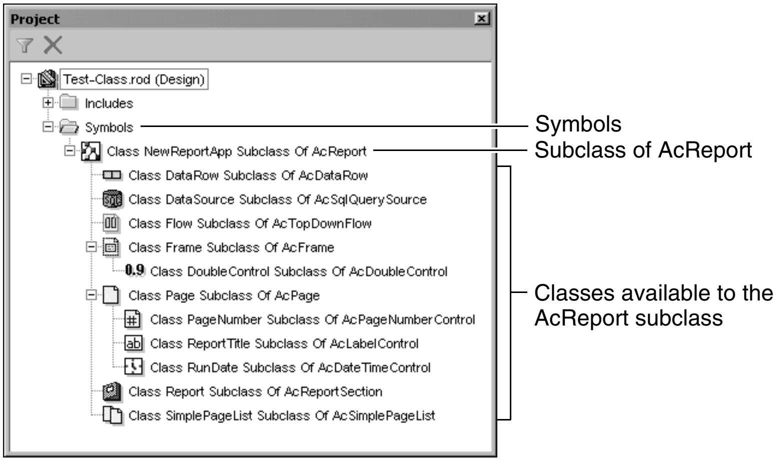


Figure 2-5 Project window

4. Expand Includes and Symbols.
5. Expand the subclass of AcReport. Project displays the Actuate Foundation Class subclasses available to the report, as shown in Figure 2-6. If the report includes libraries, the libraries and their classes also appear in Project.

Working with a class variable

A class variable defines the state and the unique attributes of an object. Most properties, such as background color, font, position, and size, are variables. The scope of a variable is within the class in which you declare the variable. The variable type determines how Actuate Basic stores the variable.



Project

Test-Class.rod (Design)
- Includes
- Symbols
  - Class NewReportApp Subclass Of AcReport
    - Class DataRow Subclass Of AcDataRow
    - Class DataSource Subclass Of AcSqlQuerySource
    - Class Flow Subclass Of AcTopDownFlow
    - Class Frame Subclass Of AcFrame
      - Class DoubleControl Subclass Of AcDoubleControl
    - Class Page Subclass Of AcPage
      - Class PageNumber Subclass Of AcPageNumberControl
      - Class ReportTitle Subclass Of AcLabelControl
      - Class RunDate Subclass Of AcDateTimeControl
    - Class Report Subclass Of AcReportSection
    - Class SimplePageList Subclass Of AcSimplePageList

Symbols

Subclass of AcReport

Classes available to the AcReport subclass

Figure 2-6 Project window listing the available classes

Actuate Basic supports two types of class variables:

- Instance

  An instance variable applies to a particular instance of a class. Actuate Basic stores one copy of the variable in each instance. Use an instance variable if each instance of a class must have a different value, such as when several text controls each must be of a different size.

  Declare an instance variable using the Dim statement, as shown in the following example:

  ```
  Dim Size As AcSize
  ```

- Static

  A static variable applies to all instances of a class and its subclasses. Actuate Basic stores one copy of the variable for all instances of the class and its subclasses. Use a static variable if all instances of a class must share the same data, such as when you create a counter to track the number of times report developers instantiate a particular class.

  Declare a static variable using the Static statement, as shown in the following example:

  ```
  Static InstanceCount As Integer
  ```

About the functional categories of variables

The AFC framework organizes variables into the functional categories that are described in Table 2-3.

Table 2-3 Functional categories of variables

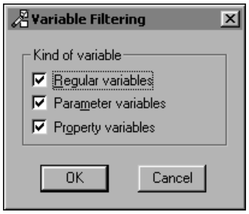
Variable type	Description
Parameter	Stores values the user supplies to specify the data to display in the report. Requester prompts for these values before report generation begins. A parameter is a static variable.
Property	Defines the attributes of an object. For example, a control has property variables such as BackgroundColor, Font, Size, and Position that define its appearance. You supply initial values for a property variable at design time. Only an instance variable can be a property. The properties of a subclass reflect property changes you make to the superclass.
Regular	Stores values that a report requires. For example, the PageNumber variable of any subclass of AcPage stores the current page number, which e.Report Designer Professional updates continuously as the report generates. This type of variable ensures that each page displays the correct page number.

How to filter the functional types of variables to display

1 In any view that displays classes, select the class for which you want to see the variable. Properties appears.

2 Choose Variables. Variables appears.

3 Choose Filter. Variable Filtering appears, as shown in Figure 2-7.



Kind of variable

☑ Regular variables
☑ Parameter variables
☑ Property variables

Figure 2-7 Variable Filtering dialog box

4 Select the types of variables you want to see. Choose OK.

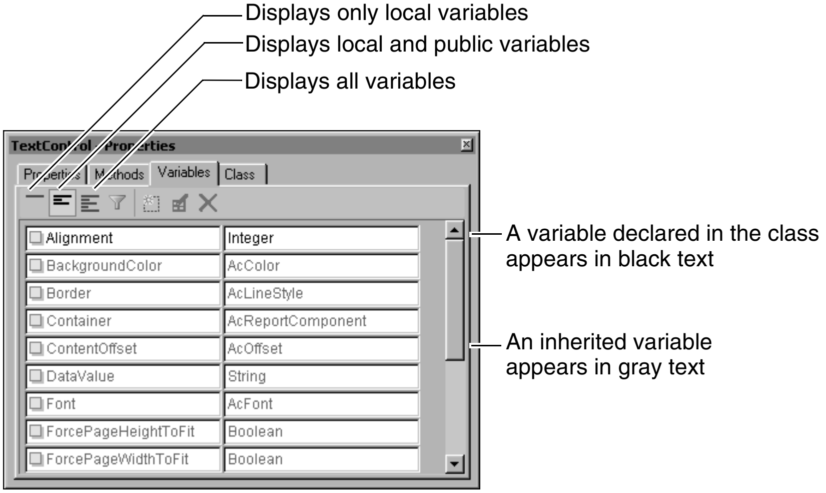
How to display sets of variables

The Variables page displays the class variables. The default setting for Variables is to display all public, inherited, and locally-declared variables in all functional types.

1 In any view that displays classes, select the class for which you want to see the variables. The Properties window appears.

2 Choose Variables. The Variables page appears.

3 Select one of the options that are shown in Figure 2-8.



□ Alignment	Integer
□ BackgroundColor	AcColor
□ Border	AcLineStyle
□ Container	AcReportComponent
□ ContentOffset	AcOffset
□ DataValue	String
□ Font	AcFont
□ ForcePageHeightToFit	Boolean
□ ForcePageWidthToFit	Boolean

Displays only local variables

Displays local and public variables

Displays all variables

A variable declared in the class appears in black text

An inherited variable appears in gray text

Figure 2-8 Display options

Defining properties

The properties of an object uniquely identify the object’s appearance and position in the report, as well as the data the object displays and other information. The AFC framework defines the following three types of properties:

- Property variables

  Most properties are variables, so you can work with a property variable in the same way as you work with any other variable. In e.Report Designer Professional, a property variable appears on both the Properties page and the Variables page. The name of the property variable is the same on both pages. For example, the BackgroundColor property for a component corresponds to the BackgroundColor variable for the same component. The Position property maps to the Position variable for the same component.

- Function properties

  The framework designates certain properties as function properties. Function properties are not variables. During the build process, the framework uses a property value that the report developer sets on the Properties page to generate a method. The code in the generated method sets the value of an associated variable, if there is one. Every function property has a generated method. Not all function properties have an associated variable.

- Miscellaneous properties

  A third group of properties, known as miscellaneous properties, do not have a generated method. Miscellaneous properties can have an associated variable. The mapping between a miscellaneous property and its associated variable is not necessarily intuitive. For example, the Key property of a group section maps to the KeyColumnName variable.

About function properties

Table 2-4 lists function properties by class. The table also shows the generated method for each property and the associated variable, if any.

Table 2-4 Function properties

Class	Function property	Generated method	Variable
AcBasePage	BalanceFlows	BalanceFlows( )	Not applicable
	CanIncreaseWidth	CanIncreaseWidth( )	Not applicable
AcBrowserScripting Control	BrowserCode	BrowserCode( )	Not applicable
	Selectable	Selectable( )	Not applicable
AcControl	Format	Format( )	Not applicable
	Searchable	Searchable( )	Not applicable
	Selectable	Selectable( )	Not applicable
	ValueType	ValueType( )	Not applicable
AcCurrencyControl	Format	Format( )	Not applicable
	Searchable	Searchable( )	Not applicable
	SearchTag	SetSearchTag( )	SearchTag
	Selectable	Selectable( )	Not applicable
	ValueType	ValueType( )	Not applicable
AcDataControl	Format	Format( )	Not applicable
	Searchable	Searchable( )	Not applicable
	Selectable	Selectable( )	Not applicable
AcDateTimeControl	Format	Format( )	Not applicable
	Searchable	Searchable( )	Not applicable
	SearchTag	SetSearchTag( )	SearchTag
	Selectable	Selectable( )	Not applicable
	ValueType	ValueType( )	Not applicable
AcFrame	AutoSplitVertical	AutoSplitVertical( )	Not applicable

Table 2-4 Function properties (continued)

Class	Function property	Generated method	Variable
AcFrame *(continued)*	CanIncreaseHeight[1]	CanIncreaseHeight( )	Not applicable
	CanIncreaseWidth	CanIncreaseWidth( )	Not applicable
	CanMoveLeft	CanMoveLeft( )	Not applicable
	CanMoveUp	CanMoveUp( )	Not applicable
	CanReduceHeight	CanReduceHeight( )	Not applicable
	CanReduceWidth	CanReduceWidth( )	Not applicable
	CustomDHTML Footer	CustomDHTML Footer( )	Not applicable
	CustomDHTML Header	CustomDHTML Header( )	Not applicable
	MaximumHeight	MaximumHeight( )	Not applicable
	MaximumWidth	MaximumWidth( )	Not applicable
	MinimumHeight	MinimumHeight( )	Not applicable
	MinimumWidth	MinimumWidth( )	Not applicable
	NoSplitBottom	NoSplitBottom( )	Not applicable
	NoSplitTop	NoSplitTop( )	Not applicable
	PageBreakAfter	PageBreakAfter( )	Not applicable
	PageBreakBefore	PageBreakBefore( )	Not applicable
	SplitMarginBottom	SplitMarginBottom( )	Not applicable
	SplitMarginTop	SplitMarginTop( )	Not applicable
	VerticalPosition	VerticalPosition( )	Not applicable
	VerticalSize	VerticalSize( )	Not applicable
AcGroupSection	GroupOn	GroupOn( )	Not applicable
	GroupInterval	GroupInterval( )	Not applicable
AcImageControl	Searchable	Searchable( )	Not applicable
	SearchTag	SetSearchTag( )	SearchTag
	Selectable	Selectable( )	Not applicable
AcLineControl	IsFrameDecoration	IsFrameDecoration( )	Not applicable
	Selectable	Selectable( )	Not applicable
	VerticalSize	VerticalSize( )	Not applicable
AcPage	CanIncreaseHeight	CanIncreaseHeight( )	Not applicable

(continues)

Table 2-4 Function properties (continued)

Class	Function property	Generated method	Variable
AcPage *(continued)*	CanIncreaseWidth	CanIncreaseWidth( )	Not applicable
	CanReduceHeight	CanReduceHeight( )	Not applicable
	CanReduceWidth	CanReduceWidth( )	Not applicable
	MaximumHeight	MaximumHeight( )	Not applicable
	MaximumWidth	MaximumWidth( )	Not applicable
	MinimumHeight	MinimumHeight( )	Not applicable
	MinimumWidth	MinimumWidth( )	Not applicable
	SplitMarginBottom	SplitMarginBottom( )	Not applicable
	SplitMarginLeft	SplitMarginLeft( )	Not applicable
	SplitMarginRight	SplitMarginRight( )	Not applicable
	SplitMarginTop	SplitMarginTop( )	Not applicable
AcPageNumber Control	Format	Format( )	Not applicable
	PageNumberType	PageNumberType( )	Not applicable
	Searchable	Searchable( )	Not applicable
	SearchTag	SetSearchTag( )	SearchTag
	Selectable	Selectable( )	Not applicable
AcRectangleControl	HorizontalSize	HorizontalSize( )	Not applicable
AcRectangleControl	IsFrameDecoration	IsFrameDecoration( )	Not applicable
	Selectable	Selectable( )	Not applicable
	VerticalSize	VerticalSize( )	Not applicable
AcSection	SearchValueExp	SetSearchValue( )	SearchValue
	PageBreakAfter	PageBreakAfter( )	Not applicable
	PageBreakBefore	PageBreakBefore( )	Not applicable
	PageBreakBetween	PageBreakBetween( )	Not applicable
AcVisualComponent	AnalysisType	AnalysisType( )	Not applicable
	CanIncreaseHeight	CanIncreaseHeight( )	Not applicable
	CanIncreaseWidth	CanIncreaseWidth( )	Not applicable
	CanMoveLeft	CanMoveLeft( )	Not applicable
	CanMoveUp	CanMoveUp( )	Not applicable
	CanReduceHeight	CanReduceHeight( )	Not applicable
	CanReduceWidth	CanReduceWidth( )	Not applicable

Table 2-4 Function properties (continued)

Class	Function property	Generated method	Variable
AcVisualComponent *(continued)*	HorizontalPosition	HorizontalPosition( )	Not applicable
	HorizontalSize	HorizontalSize( )	Not applicable
	MaximumHeight	MaximumHeight( )	Not applicable
	MaximumWidth	MaximumWidth( )	Not applicable
	MinimumHeight	MinimumHeight( )	Not applicable
	MinimumWidth	MinimumWidth( )	Not applicable
	TocValueExp	SetTocEntry( )	TocEntry
	Searchable	Searchable( )	Not applicable
	SearchAlias	SearchAlias( )	Not applicable
	Selectable	Selectable( )	Not applicable
	VerticalPosition	VerticalPosition( )	Not applicable
	VerticalSize	VerticalSize( )	Not applicable

1. Properties in the Dynamic Size and Position group, such as CanIncreaseHeight, CanIncreaseWidth, CanMoveUp, and so on, apply to frames, pages, and data controls.

About miscellaneous properties

Table 2-5 shows miscellaneous properties by class, along with the variable associated with the property, if any.

Table 2-5 Miscellaneous properties

Class	Miscellaneous property	Variable
AcBaseFrame	SearchValueExp	SearchValue
AcConditional Section	IfExp	Not applicable
AcCrosstab	LabelMultipleValues	Not applicable
	ValuePlacement	Not applicable
AcDataControl	SampleValue[1]	Not applicable
	ValueExp	Not applicable
AcGroupSection	Key	KeyColumnName
AcImageControl	FileNameExp	Not applicable

(continues)

Table 2-5 Miscellaneous properties (continued)

Class	Miscellaneous property	Variable
AcOdaSource	DriverName	Not applicable
	OdaInterfaceName	Not applicable
AcReport Component	TocValueExp	Not applicable
AcReportSection	OrderBy	Not applicable
AcSection	GrantExp	Not applicable
	SearchValueExp	SearchValue
AcVisual Component	LinkExp	Not applicable
	ObjectVariable	Not applicable

1. SampleValue also applies to the AcBrowserScriptingControl and AcPageNumberControl classes

Using a parameter

To gather values when a report runs, specify a variable as a parameter. A report typically uses parameters to filter the data to retrieve and display. For example, a query can retrieve all customer records from a Customer table. Parameters support specifying additional filter conditions when a report user runs the report, such as retrieving only records for customers in Japan or only records in a specific date range. You can also create a parameter to set properties such as the font and color of an object when a report user runs the report.

You can create a parameter using the Properties page, Query Editor or Textual Query Editor, or Parameter Editor.

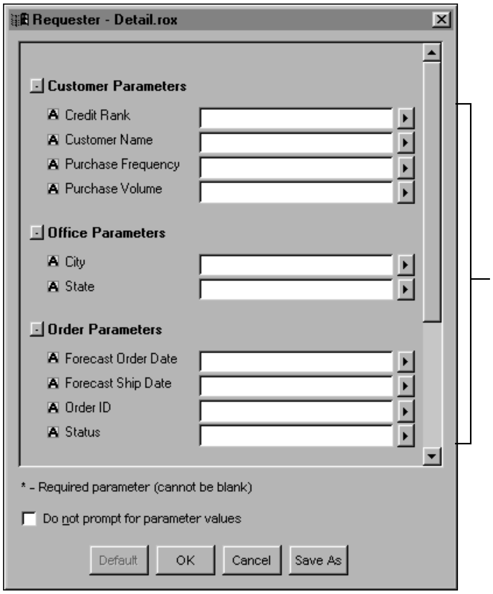
A variable you specify as a parameter appears on the Variables page. It also appears in Requester when a report user runs a report that uses parameters, as shown in Figure 2-9.

For more information about parameters and how to create them, see *Developing Reports using e.Report Designer Professional.*

Using a regular variable

To store values that e.Report Designer Professional uses when generating a report, use a regular variable. For example, a frame uses the Container variable to store a reference to its container object. You typically create a regular variable to store values that methods need. For example, if you write a method that sets alternating rows to different colors, you can use a regular variable to store the current row number.

To create a regular variable, choose New on the Variables page.



* - Required parameter (cannot be blank)

□ Do not prompt for parameter values

If a report uses parameters, Requester prompts for the parameters during report generation

Figure 2-9 Requester showing parameters

About variable visibility

The visibility of a variable determines how, when, and where you can view and use the variable. A property appears on the Properties page and its associated variable, if any, on the Variables pages. A private variable appears only on the Variables page. The visibility you can assign to a variable depends on whether it is an instance variable or a static variable. For example, only a static variable can be a parameter.

Table 2-6 describes the variable visibility settings and shows the type of variable to which each setting applies.

Table 2-6 Variable visibility settings

Visibility setting	Description	Applies to
Private	Appears on the Variables page for the class in which it is declared. Private visibility affects only where the	Instance and static variables

(continues)

Table 2-6 Variable visibility settings (continued)

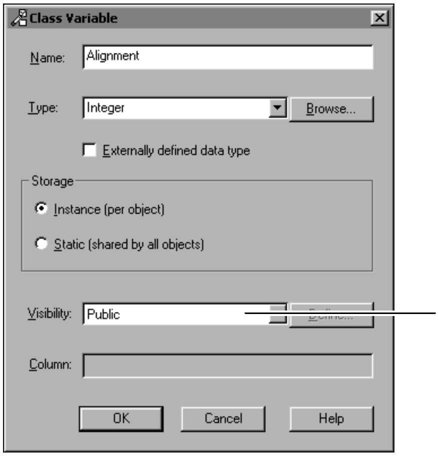
Visibility setting	Description	Applies to
Private *(continued)*	variable is visible. A private variable is available to the class and its subclasses. In Actuate Basic code, you access a Private variable in the same way as a Public variable.	Instance and static variables
Parameter	Appears in Requester when a user runs a report that uses parameters. The user can type or select a value to set additional run-specific filter conditions.	Static variable of scalar data type
Public	Appears on the Variables page for the class in which it is declared and its subclasses.	Instance and static variables

How to view the visibility setting for a variable

1 In any view that displays classes, such as Report Structure or Layout, select the class for which you want to view a variable. The Properties window appears.

2 Choose Variables. The Variables page appears.

3 Choose Edit. Class Variable appears, as shown in Figure 2-10.


Class Variable

Name: Alignment

Type: Integer

□ Externally defined data type

Storage

● Instance (per object)

○ Static (shared by all objects)

Visibility: Public

Column:

Visibility setting

Figure 2-10 Class Variable dialog box showing the visibility setting

Class Variable displays information about the variable. The visibility setting appears in the lower portion of the window.

Creating a variable

To create a variable in a report design, use the Variables page of a component. On the Variables page, you can set the data type of the variable and indicate whether the variable is private to its class or public. You can also show whether the variable uses an externally defined data type, meaning a data type from outside the Actuate Foundation Classes, and whether the variable is an instance variable, available only to a specific object, or a static variable, available to all classes in the report.

Use the Properties page to create an instance variable that is a property. When you create a property, its name appears on both the Variables page and the Properties page.

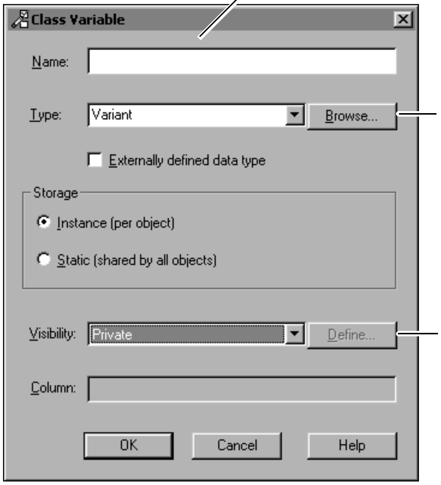
How to create a variable

1 In any view that displays classes, such as Report Structure or Layout, select the class to which you want to add a variable. The Properties window appears.

2 Choose Variables. The Variables page appears.

3 Choose New. Class Variable appears, as shown in Figure 2-11.

Type the variable or array name



Name:

Type: Variant

☐ Externally defined data type

Storage

● Instance (per object)

○ Static (shared by all objects)

Visibility: Private

Column:

Select the data type of the variable

Select the visibility of the variable

Figure 2-11 Class Variable dialog box for creating a class variable

4 Type the variable name.

You can also type the name of an array, such as ProductArray(10) or MultiArray(1 To 3, 1 To 3, 1 To 3).

5 Select or type the data type of the variable. The default value is Variant. You can use any of the following as data types:

- An Actuate Basic data type, such as Integer, Boolean, Double, or String
- A custom data type
- An Actuate Foundation Class data type, such as AcColor or AcFont
- The name of any declared class

6. If this data type is a custom data type, select Externally Defined Data Type.
7. Select either Instance or Static.
8. Select the visibility of the variable. Choose OK.

   The variable appears on the Variables page.

How to set the value of a property variable programmatically

You can programmatically set the values of a property variable at design time in an overridable method, as shown in the following example:

```
Sub Start( )
   Super::Start( )
   Font.Size = 22
   Font.FaceName = "Arial"
   …
End Sub
```

Editing a variable

Use the Variables page to modify the data type of the variable or access the Class Variable page to make other modifications. You can modify only a variable scoped to a class. You cannot modify an inherited variable. Inherited variables appear in gray on the Variables page.

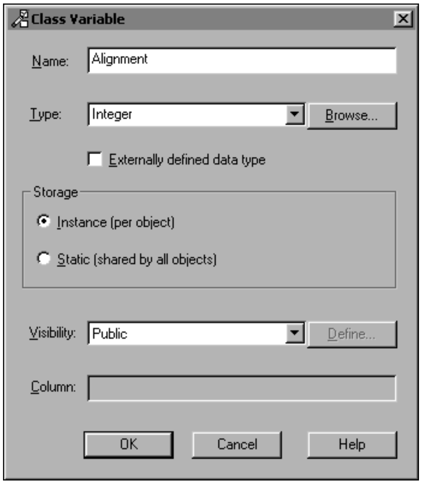
How to edit a variable

1. In any view that displays classes, such as Report Structure or Layout, select the class that contains the variable to edit. The Properties window appears.
2. Choose Variables. The Variables page appears.
3. Select the variable to edit. Choose Edit. Class Variable appears, as shown in Figure 2-12.
4. Modify the variable. Choose OK.

To revert a variable to its previous definition immediately after making a change, choose Edit➛Undo. This command works only if you do not perform another task after editing the variable.

Deleting a variable

Use the Variables page to delete a variable. You can delete only a variable scoped to a class. You cannot delete an inherited variable. Inherited variables appear in gray on the Variables page. For a property variable, if you delete the variable, you also delete the property.


Class Variable

Name: Alignment

Type: Integer

☐ Externally defined data type

Storage

● Instance (per object)

○ Static (shared by all objects)

Visibility: Public

Column:

Figure 2-12 Class Variable dialog box for editing a class variable

How to delete a variable

1. In any view that displays a class, select the class that contains the variable you want to delete. The Properties window appears.
2. Choose Variables. The Variables page appears.
3. Select the variable to delete. Choose Delete.

To recover a variable immediately after you delete it, choose Edit➛Undo.

Working with a method

A method specifies the actions an object performs. A method is a procedure you define within a class declaration. Most predefined methods in the Actuate Foundation Classes support generating a report.

You can create a new method to add functionality to a class. You can also create a method if the functionality you need does not exist in a predefined method. If the functionality you require is an extension or a version of an existing method, you can override the method.

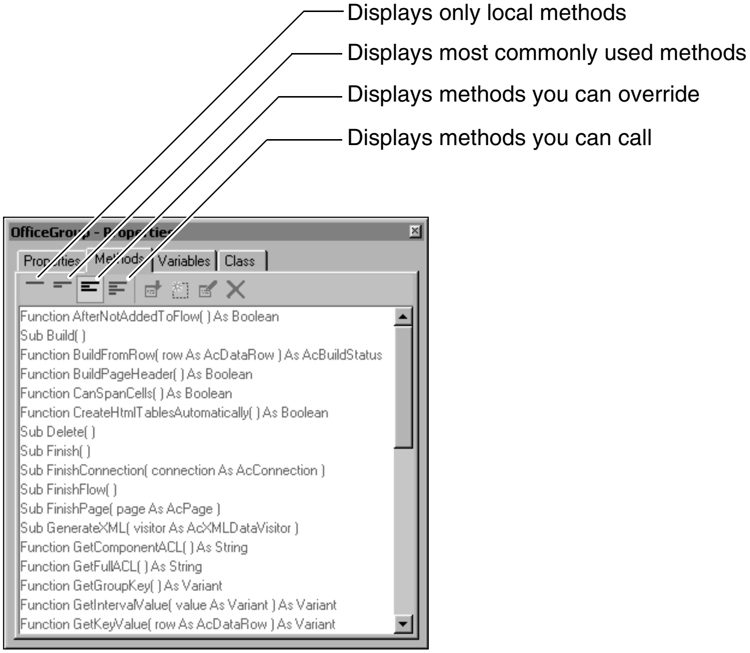
Actuate Foundation Classes support the following categories of predefined methods:

- Methods you can override
- Methods you can call
- User-defined methods

How to select the set of methods to display

In e.Report Designer Professional, the Methods page provides filters that support viewing methods in each category. The default display setting for Methods shows inherited and locally-declared overridable methods.

1. In any view that displays classes, select the class for which you want to display methods. The Properties window appears.
2. Choose Methods. The Methods page appears.
3. Select one of the options that are shown in Figure 2-13.


Displays only local methods
Displays most commonly used methods
Displays methods you can override
Displays methods you can call

```
Function AfterNotAddedToFlow( ) As Boolean
Sub Build( )
Function BuildFromRow( row As AcDataRow ) As AcBuildStatus
Function BuildPageHeader( ) As Boolean
Function CanSpanCells( ) As Boolean
Function CreateHtmlTablesAutomatically( ) As Boolean
Sub Delete( )
Sub Finish( )
Sub FinishConnection( connection As AcConnection )
Sub FinishFlow( )
Sub FinishPage( page As AcPage )
Sub GenerateXML( visitor As AcXMLDataVisitor )
Function GetComponentACL( ) As String
Function GetFullACL( ) As String
Function GetGroupKey( ) As Variant
Function GetIntervalValue( value As Variant ) As Variant
Function GetKeyValue( row As AcDataRow ) As Variant
```

Figure 2-13 Methods page

About methods you can override

An overridable method supports customizing parts of the report generation or report viewing process. For example, the methods that are part of a class protocol, such as New( ), Start( ), Build( ), Fetch( ), Finish( ), are overridable.

When overriding a method, use the following guidelines:

- Overriding a callable method can adversely impact the report generation process.
- Understand how the method works and the context in which it runs.
- Decide whether you are replacing or extending the inherited method.
    - To extend the code, you must call the original method in the superclass, as shown in the following example:

      ```
      Function Start( ) As Boolean
         Start = Super::Start( )

            ' Your new code

         End Function
      ```

      Depending on the method and what you want to accomplish, you can call the superclass method before, within, or after your code.
    - To replace the code, do not call the method in the ancestor class. You must ensure that the replacement code performs all the necessary tasks that the original method performs.

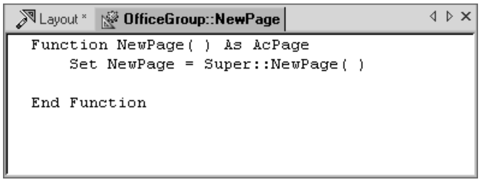
How to override a method

1 In any view that displays classes, such as Report Structure or Layout, select the class containing the method you want to override. The Properties window appears.

2 Choose Methods. The Methods page appears.

3 Select the method to override.

4 Choose Override. The method editor appears in Layout, as shown in Figure 2-14.



```
Function NewPage( ) As AcPage
    Set NewPage = Super::NewPage( )

End Function
```

Figure 2-14 The method editor

5 Add code to the method.

To retain and augment the method’s default behavior, keep the Super statement. To replace the method’s default behavior, remove the Super statement.

6 Close the method editor by choosing the X in the upper right corner of Layout.

About methods you can call

A callable method typically provides a defined service or information about an object. You should not override these methods. For example, a data adapter class provides methods such as SeekTo( ), SeekBy( ), SeekToEnd( ), and Rewind( ) that you can call to access and navigate through data. Report component classes provide methods such as IsContainer( ), IsLeaf( ), IsVisual( ), and HasContents( ) that you can call to get information about an object. Page list classes define methods such as GetPageCount( ), GetContents( ), GetCurrentPage( ), and GetFirstPage( ) that you can call to get a value your code requires.

If you cannot find a predefined method for a task, create a new method.

About private methods

The AFC framework calls private methods to perform internal tasks. Do not override a private method. Actuate does not support overridden private methods. If your report design contains an overridden private method, e.Report Designer Professional displays a warning message when you compile or run the report.

About user-defined methods

To add functionality that does not exist in a predefined method, create a new method for a class. If the functionality to add is an extension of an existing method, consider overriding the existing method instead.

Creating a method

Actuate Basic imposes no restrictions on what you can do with a method you create. A method can significantly affect the behavior of an object. Design, code, and test methods carefully. When creating a method, use the following guidelines:

- Confirm that creating a method is a better choice than overriding an existing method. If you plan to use the method in a variety of contexts, creating a method is the better choice.
- Minimize conditions you impose on other programmers who use the method. For example, be aware of the complexities that arise from creating the following kinds of methods:
  - Methods a user must call to use the component

- Methods that must execute in a strict order
- Methods that put the component into a state that could invalidate another method or event

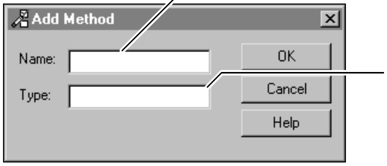
If you cannot avoid such conditions, write code that manages incorrect use of your methods. For example, if calling a method puts the component into a state that renders another method invalid, program the other method to test the state before executing its main code. At a minimum, display a warning message and a cancellation option if an error occurs. Use code comments to describe any special requirements or preconditions.

How to create a method

1 In any view that displays classes, select the class to which to add a new method. The Properties window appears.

2 Choose Methods. The Methods page appears.

3 Choose New. Add Method appears, as shown in Figure 2-15.



Specify the method name

Specify the method's return type, if needed

Figure 2-15 Add Method dialog box

4 Type a name for the method.

5 Specify a return data type for the method, if necessary. Choose OK.

The method name appears on the Methods page as a locally-defined method. At the same time, the method editor appears in Layout, displaying the method declaration.

6 Write code for the method.

7 Close the method editor by choosing the X in the upper right corner of Layout.

Naming a method

The name of a method must follow the naming conventions for any other object in Actuate Basic, such as a class or a variable.

When naming a method, use the following guidelines:

- Begin a method name with a verb.

  For example, GetHorizontalPosition is clearer than XPosition, which sounds like a property.

- Use unambiguous descriptive names that reflect the method’s purpose.

  For example, a name such as ReadDataRow is more informative than DoDataRow.

For more information about Actuate Basic naming conventions, see *Programming with Actuate Basic.*

You can use duplicate method names within a report if the methods are overloaded or in different scopes.

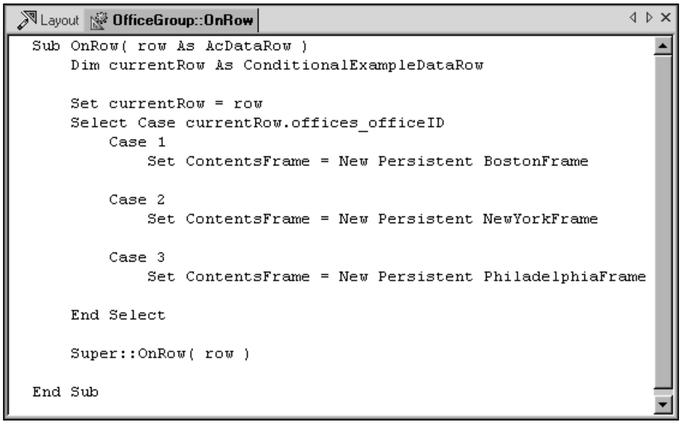
Editing a method

You can edit only a method you create or an overridden method. You cannot edit an inherited method.

How to edit a method

1. In any view that displays classes, such as Report Structure or Layout, select the class containing the method to edit. The Properties window appears.
2. Choose Methods. The Methods page appears.

   Methods displays inherited and locally-defined methods. To narrow or expand the list of methods that appears, change the filtering options.
3. Select the method to edit. Choose Edit. The method editor appears in Layout, as shown in Figure 2-16.



```
Sub OnRow( row As AcDataRow )
    Dim currentRow As ConditionalExampleDataRow

    Set currentRow = row
    Select Case currentRow.offices_officeID
        Case 1
            Set ContentsFrame = New Persistent BostonFrame

        Case 2
            Set ContentsFrame = New Persistent NewYorkFrame

        Case 3
            Set ContentsFrame = New Persistent PhiladelphiaFrame

    End Select

    Super::OnRow( row )

End Sub
```

Figure 2-16 The method editor

4. Modify the code as necessary.
5. To close the method editor, choose the X in the upper right corner of Layout.

Deleting a method

You can delete only a method you create or a method that you have overridden. You cannot delete an inherited method.

How to delete a method

1 In any view that displays classes, select the class containing the method to delete. The Properties window appears.

2 Choose Methods. The Methods page appears.

Methods displays inherited and locally-defined methods. To narrow or expand the list of methods that appears, change the filtering options.

3 Select the method you want to delete. Choose Delete.

To recover a method immediately after deleting it, choose Edit➛Undo. This command works only if you do not perform another task after deleting the method.

Overloading a method

Overloading a method means creating multiple methods in the same class, with the same name but different argument lists. The compiler selects the appropriate version of the method based on the arguments you use to call the method. Overloading supports varying the number and data types of a method's arguments. In the following example, StrConcat( ) is an overloaded method:

```
Function StrConcat( str1 As String, str2 As String ) As String
Function StrConcat( str1 As String, str2 As String, concatenator
   As String ) As String
```

The first method is the standard call that concatenates strings using a comma character. To get the comma character, this function calls the second function using the same str1 and str2 arguments, along with a comma (",") as the third argument.

```
myFirstString = StrConcat( myName, myProperty, ", " )
```

The output of this method is similar to the following example:

```
CustomerName, Address
```

You can use the second function if you want to concatenate strings using a different character or set of characters, such as two hyphens to simulate an em-dash, as shown in the following example:

```
mySecondString = StrConcat( myName, myProperty, "- - " )
```

The output of this method is similar to the following example:

```
DataSource - - DriverName
```

Chapter

3

Working with an object

This chapter contains the following topics:

- About objects and object reference variables
- Creating an object
- Using an object reference variable
- About object lifetime

About objects and object reference variables

An object is an instance of a class. Every component in a report design is an object, including frames, controls, and sections. In e.Report Designer Professional, you set the properties of an object when you design the object. Later, you can modify an object's properties for a specific report. For example, you can display negative numbers in red and positive numbers in black. To do so, declare and use variables that refer to objects.

A variable that is a pointer to an object is called an object reference variable. An object reference variable refers to an object that can have different property values from the original class definition.

This chapter describes how to create an object and use an object reference variable. For more information about variables and all other information about Actuate Basic, see *Programming with Actuate Basic.*

Creating an object

To create an object, first declare an object reference variable. Then, take one of the following steps:

- Create the object using the New or New Persistent keywords.
- Access an existing object by calling a method that returns an object of the appropriate class.

The following sections describe these steps in detail.

Declaring an object reference variable

You declare an object reference variable the same way you declare other variables, except that you assign the class or AnyClass type as the variable type. Declare an object reference variable using one of the following statements:

- Dim
- ReDim
- Static
- Global

The object reference declaration uses the following syntax:

```
{Dim | ReDim | Static | Global} <variable name> As {<class> |
    AnyClass}
```

The following sections describe how to declare an object reference variable as a specific class and as type AnyClass.

Declaring an object reference variable as a specific class

You typically declare an object reference variable as a specific class. You can specify an Actuate Foundation Class, its subclass, or a custom class. To declare an object as a specific class, use the Class statement. For example, to create an object reference variable of type AcLabelControl, use a declaration similar to the following statement. This object reference variable can refer to any object of the AcLabelControl class or its subclasses.

```
Dim MyLabelControl As AcLabelControl
```

Declaring an object reference variable as AnyClass type

If you do not know an object's class, declare the object reference variable for that object as AnyClass using the following syntax:

```
Dim handle As AnyClass
```

Using Actuate Basic to create an object

Declaring an object reference variable does not create the object. The object does not exist in memory until you instantiate the class. To create the object, use the New or New Persistent keyword using the Set statement. Set…New and Set…New Persistent use the following syntax for creating an object:

```
Set <variable name> = New [Persistent] <class> [(<argument list>)]
```

Set…New and Set…New Persistent create a new object of <class> and store the reference to the object in <variable name>. The following example creates a label and stores the reference to the label in the MyLabel object reference variable:

```
Set MyLabel = New AcLabelControl
```

Use Set...New Persistent to keep the object until the user deletes the report. When e.Report Designer Professional generates the report objects that users view and use, it creates persistent objects by default.

Using an object reference variable

After you create an object or obtain the handle to an existing object, you can work with it using an object reference variable. More than one object reference variable can refer to the same object. You can call the object's methods or access the object's member variables. You can also work with the object reference variable itself. For example, you can pass an object reference variable to a procedure, make the variable refer to another object, compare an object reference variable, and test it. The following sections describe how to perform these tasks.

When working with an object, it is important to understand the difference between a simple variable, such as an integer or string variable, and an object reference variable. When you use a simple variable, you manipulate a value directly. If you assign the value of one simple variable to another, you copy the value. Subsequent changes to the original variable do not affect the copy. When you use an object reference variable, a change to the original object affects all references to the object. The following sections describe how changes to variable values affect simple and object reference variables.

Working with a simple variable

A simple variable contains a value. For example, the SearchTag variable for a report section contains a character string. The RowNumber variable for a data row contains an integer. When you assign one variable to another, you copy the contents of the first variable to the second. Subsequent changes to the contents of the original variable have no effect on the second variable, as shown in the following example:

```
Dim Variable1, Variable2 As Integer
Variable1 = 7
Variable2 = Variable1   'Variable2 contains the value 7
Variable1 = 77
Print Variable1         'Prints 77
Print Variable2         'Prints 7
```

Working with an object reference variable

As with a simple variable, an object reference variable contains a value. The value of an object reference variable is the reference to, or address of, an object. The object reference variable does not contain the object itself. You can assign an object reference variable to an object or to another object reference variable. When you assign one object reference variable to another, you do not copy the object. Instead, you create a second reference to the same object.

The following example creates a label and sets its text property. The object reference variable LabelControl1 refers to the label.

```
' Declare an object reference variable
Dim LabelControl1 As AcLabelControl
' Create the object
Set LabelControl1 = New AcLabelControl
' Set the Text property of the label
LabelControl1.Text = "Annual Sales Report"
```

Figure 3-1 shows the result of the preceding example.

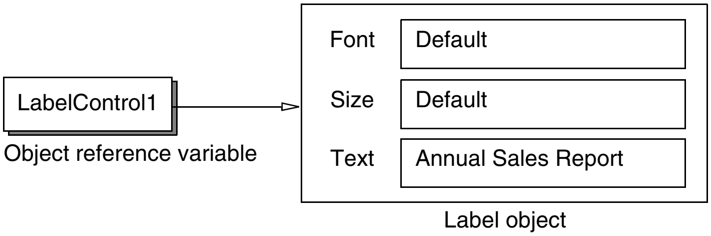


Figure 3-1 Setting the text property for a label using an object reference variable

The following example assigns another object reference variable, LabelControl2, to the first object reference variable, LabelControl1:

```
Dim LabelControl2 As AcLabelControl
Set LabelControl2 = LabelControl1
LabelControl2.Text = "Monthly Sales Report"

Print LabelControl2.Text      'Prints "Monthly Sales Report"
Print LabelControl1.Text      'Prints "Monthly Sales Report"
```

Figure 3-2 shows the result of the preceding example.

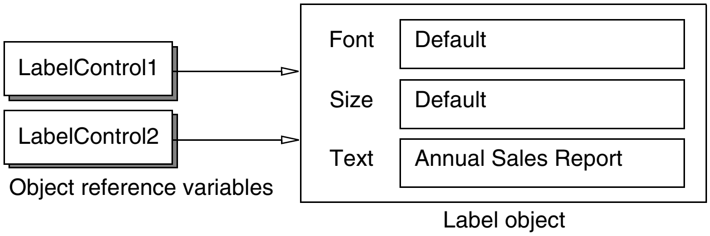


Figure 3-2 Using multiple object reference variables

Referring to an object's variables and methods

To change, store, or retrieve an object's values, you refer to its instance variables and methods using dot notation, as shown in the following example:

```
<object reference variable>.<variable>
<object reference variable>.<method>
```

The dot instructs Actuate Basic to access an instance variable or method in an object. For example, to refer to a variable or method in a label control, specify the object reference variable, followed by a dot, followed by the variable or method name, as shown in the following example:

```
MyLabel.BackgroundColor
MyLabel.Build( )
```

To change the background color of the label, assign a value to one of its variables, as shown in the following example:

```
MyLabel.BackgroundColor = Yellow
```

If an object contains an object reference variable that points to another object, as shown in Figure 3-3, you can use dot notation to build a path of references.

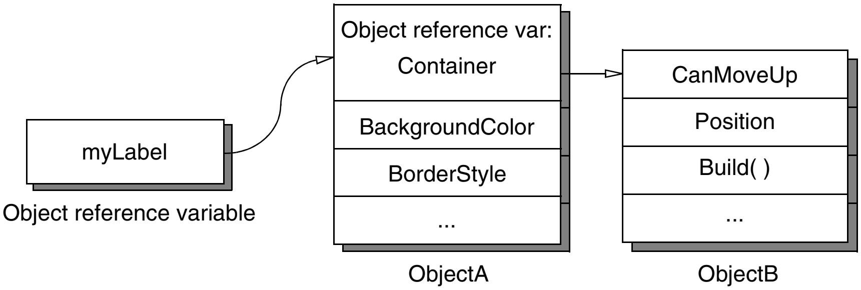


Figure 3-3 Using dot notation to build a path of references for an object reference variable

Using Figure 3-3, you can build the following path:

```
myLabel.Container.CanMoveUp
```

Referencing a method of a class

You typically reference an object's methods to execute a task on the object. Sometimes, however, you must reference a method defined in a superclass. For example, if you override a method but must still perform its original task, you can call the method in the superclass.

Referencing a method in a superclass

When you reference a method in a superclass, e.Report Designer Professional first searches the superclass of the current class, then continues up the hierarchy until it finds the method. You typically use this technique to augment the functionality of an overridden method. Referencing a method in a superclass executes the original code and the code you add. Referencing a method in a superclass has the following advantages:

- Because you do not hard code a class name, your code is more reusable.
- You do not have to know the name of the superclass.

To call a method in a superclass, use the following syntax:

```
Super::<method>
```

Referencing a method using a class name

You can specify the class containing the method you want to call. If you specify a class name, Actuate Basic searches only the class you specify. Specify a class name if you modified the method in each successively derived class and you must call a specific version of the class. To specify a class containing the method, use the following syntax:

```
<class name>::<method>
```

For example, ClassC derives from ClassB and ClassB derives from ClassA. Each class has its own version of the Build( ) method. To write code for MyLabel, a subclass of ClassC, and use ClassA's Build( ) method, use the following statement:

```
MyLabel.ClassA::Build
```

To write code for MyLabel without using ClassA's Build( ) method, use the following statement:

```
MyLabel.Build
```

In the preceding example, if MyLabel's Build( ) method does not contain overridden code, the Build( ) method calls Super::Build( ), which is the Build( ) method of ClassC.

Resolving an ambiguous method call

Inheritance can result in two methods with the same name that execute different tasks because they are in different scopes. When a report contains duplicate method names, you must qualify the method name when you call the method. Otherwise, Actuate Basic resolves an ambiguous method call by searching within the current instance first, then searching within the global scope. In the following examples, DerivedClass derives from BaseClass. BaseClass defines methods X and Y. DerivedClass defines its own version of method Y. When you call Y( ) from X( ) within MyObject, an instance of DerivedClass, Actuate Basic calls the DerivedClass version of Y( ).

```
Class BaseClass
   …
   Sub X
      Y( )
      …
   End Sub
   Sub Y
      Beep
   End Sub
End Class

Class DerivedClass Subclass of BaseClass
'DerivedClass inherits method X and redefines method Y
```

```
   …
   Sub Y
      Super::Y
      MsgBox "This operation is invalid"
   End Sub
End Class
…
Dim MyObject As DerivedClass
Set MyObject = New DerivedClass

X( )                 'X( ) calls DerivedClass' version of Y( )

MyObject.Y           'Refers to Y( ) in DerivedClass
MyObject.BaseClass::Y 'Refers explicitly to Y( ) in BaseClass
```

Assigning an object to an object reference variable

To assign an object reference variable to an object, use the Set statement, as shown in the following example:

```
Set <object reference variable> = <object expression>
```

Do not use Let to assign an object reference variable to an object. Let, which takes the form x = y, assigns one simple variable to another. Because object reference variables do not contain actual values, using Let as shown in the following example results in an error:

```
Dim x As AcLabelControl
Dim y As AcLabelControl

Let x = New AcLabelControl
y = x      'Compilation error—Invalid assignment
```

You can assign an object to an object reference variable if the object is of the same type as the object reference variable or of a type that derives from the type of the object reference variable.

As shown in the following example, you can assign Control1 to Control2 because you declare both variables as AcControl:

```
Dim Control1 As AcControl
Dim Control2 As AcControl

Set Control1 = New AcControl
Set Control2 = Control1
```

As shown in the following example, although you declare Control1 and Control2 as different types, you can assign Control1 to Control2 because AcTextControl derives from class AcControl:

```
Dim Control1 As AcTextControl
Dim Control2 As AcControl
```

```
Set Control1 = New AcTextControl
Set Control2 = Control1
```

You cannot assign an object to an object reference variable of an unrelated class or a parent class. For example, you cannot assign a report object to a control object reference variable.

The following example results in a run-time error because AcControl does not derive from AcTextControl:

```
Dim Control1 As AcTextControl
Dim Control2 As AcControl
SetControl2 = New AcControl
SetControl1 = Control2 'Runtime error—Illegal handle conversion
```

Setting an object reference variable to Nothing

When an object reference variable does not refer to an object, it has the special value, Nothing. This value has a similar purpose as the special value Null has for a simple variable. An object reference variable cannot hold the value Null. When you declare an object reference variable, it is initially set to Nothing. You can assign Nothing to any object reference variable using Set, as shown in the following example:

```
Set MyControl = Nothing
```

Passing an object reference to a procedure

As with other variables, you can pass an object reference to a procedure as a parameter and return it as a return value. The following examples show when to pass an object reference to a procedure as a parameter. The procedure in the following example receives a reference to an object, AnyControl, as a parameter and sizes it:

```
Sub SizeObject(AnyControl As AcControl)
   AnyControl.Size.Width = 5000 'Twips
   AnyControl.Size.Height = 1000 'Twips
End Sub
```

The function in the following example creates a label and returns a reference to it:

```
Function NewLabel( ) As MyLabelControl
   Set NewLabel = New MyLabelControl
End Function
```

Getting information about an object

Table 3-1 lists the Actuate Basic functions that you can use to get information about an object.

Table 3-1 Actuate Basic functions for getting object information

Function	Description
GetClassID( )	Returns the unique number that e.Report Designer Professional automatically assigns to all objects. Objects of the same class have the same ID number. Use GetClassID( ) to determine whether two objects are of the same class without the overhead of a string comparison.
GetClass Name( )	Returns the name of the object's class. Use GetClassName( ) when you need an object's class before performing an action.
IsKindOf( )	Tests whether an object is of a specified class or is derived from a specified class. Returns True if the object is an instance of the specified class or is an instance of a subclass of the specified class. Otherwise, this function returns False. Use IsKindOf( ) to test whether an object is of a particular class before performing an action.

Testing an object reference using the Is operator

Use the Is operator to perform the following tasks:

- Test whether an object reference variable does not refer to an object (Is Nothing).
- Compare two object reference variables.

Testing for Nothing

Use Is with Nothing to see if an object reference variable does not refer to an object. The procedure in the following example displays different messages depending on whether an object reference variable is empty:

```
Sub TestContent( element As AcVisualComponent)
   If element Is Nothing Then
      MsgBox "The object reference variable is empty"
   Else
      MsgBox "The object reference variable is set"
   End If
End Sub
```

Comparing object reference variables

Use Is to compare two object reference variables and determine whether they both refer to the same object. The function in the following example determines whether AcControl is in a list of controls that make up the contents of a frame:

```
Function IsInFrame (frame As AcFrame, control As AcControl) As
   Boolean
   Dim element As AcVisualComponent
   Dim iter As AcListIterator
   Set iter = frame.ContentList.NewIterator( )
   Do While iter.HasMore( )
      Set element = iter.GetNext( )
      If element Is control Then
         IsInFrame = True
         Exit Function
      End If
   Loop
   IsInFrame = False
End Function
```

About object lifetime

The lifetime of an object depends on whether the object is transient or persistent. The following sections describe transient, persistent, and pinned objects.

About transient objects

e.Report Designer Professional creates transient, or temporary, objects to perform specialized tasks during report generation. e.Report Designer Professional releases these objects from memory once the specialized tasks finish. Examples of transient objects include data streams and connections.

e.Report Designer Professional releases a transient object from memory when the last reference variable that refers to it is destroyed or is set to refer to another object. e.Report Designer Professional keeps track of the reference count, which increases each time a new object reference variable refers to the object. The reference count decreases each time an object reference variable is:

- Set to Nothing
- Set to refer to another variable
- Destroyed because it is out of scope or because it is a variable of an object that is destroyed

When the reference count is zero, e.Report Designer Professional deletes the object.

About persistent objects

The persistent objects that e.Report Designer Professional creates exist until you delete the report file. All objects that appear in the report at view time are persistent, including data controls, graphical elements, sections, and page layout components. Because the report object instance (.roi) file saves the report data and structure, users can view the report at any time.

About pinned objects

During e.Report generation, Actuate Basic objects are locked into memory, or pinned, until they are completely processed. Normally, pinned objects are released from memory once they have been finalized. Incorrectly written code in overridden methods can cause objects to remain pinned in memory indefinitely. If large numbers of pinned objects accumulate in memory during generation of an e.Report, that report will run more slowly and consume more system resources than it should. In extreme cases, this can cause significant system performance degradation.

When a report object instance (.roi) file is closed, all the objects in that file should have been finalized and released from memory. If any objects are still pinned when the ROI file is closed, there is a problem in the report design. e.Report Designer Professional and iServer now provide information about objects that remain pinned when an ROI file is closed. This information is in the form of a warning message: "Warning <Number> objects were still pinned when the ROI file was closed." A list of the pinned objects follows the message. This message does not indicate a defect or behavior change in the e.Reports runtime or Actuate Foundation Classes; it has been added to warn you of faulty overridden method code in your report designs.

If you do not correct the faulty code in an existing report which now causes the new warning message to appear, that report will continue to run exactly as before. However, Actuate strongly recommends that you correct the defective code. Changes in usage patterns or data could cause the number of excess pinned objects created by the report to rise to a level that causes problems. You should not put a new report into production if it produces this warning message.

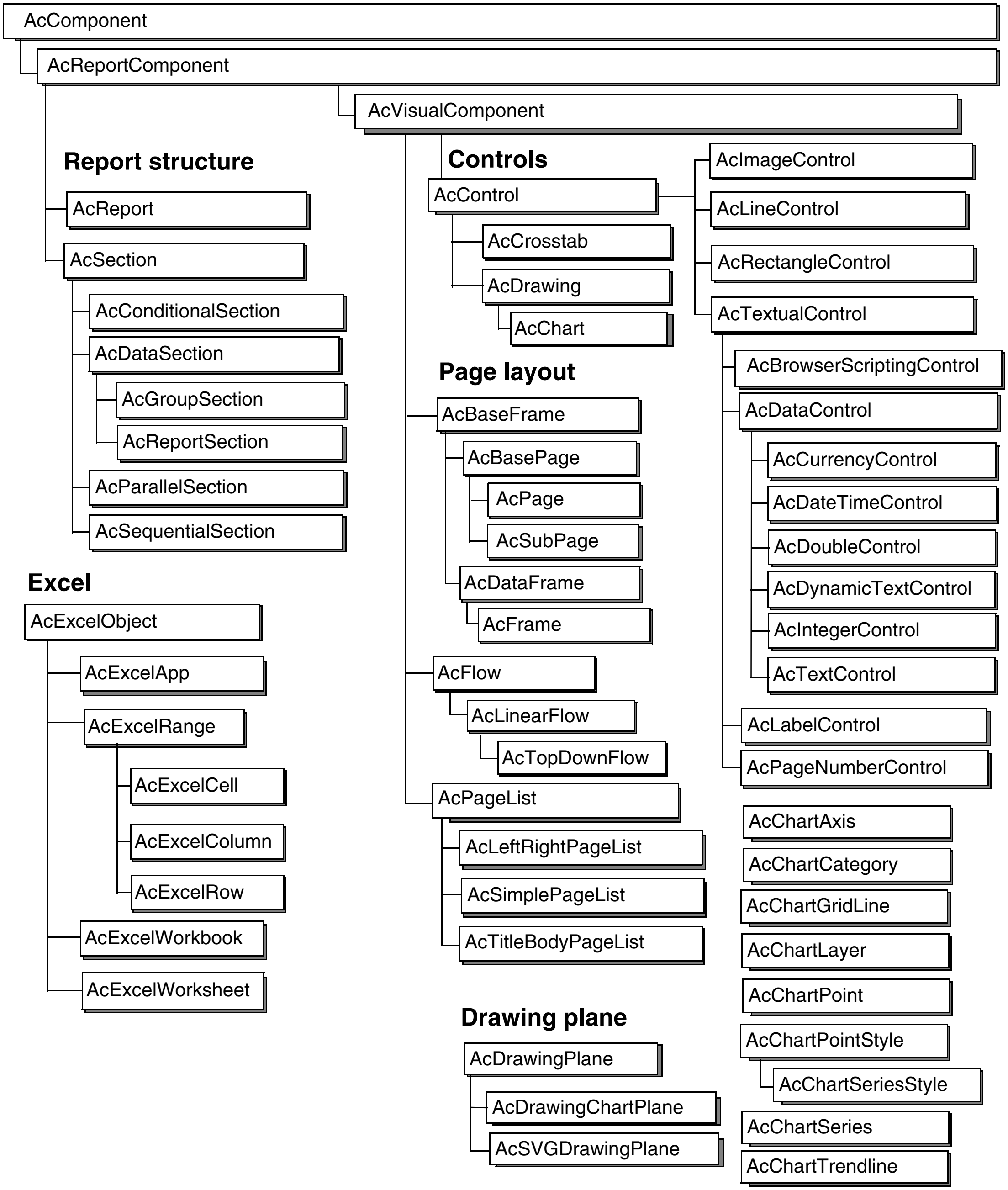
Understanding and resolving the causes of excess pinned objects requires in-depth analysis of report designs, and could require substantial modifications to those designs. This work is beyond the scope of services covered by your Actuate Support agreement. If you are uncertain about how to resolve pinned object issues, Actuate's Professional Services team can review your report designs and provide advice.

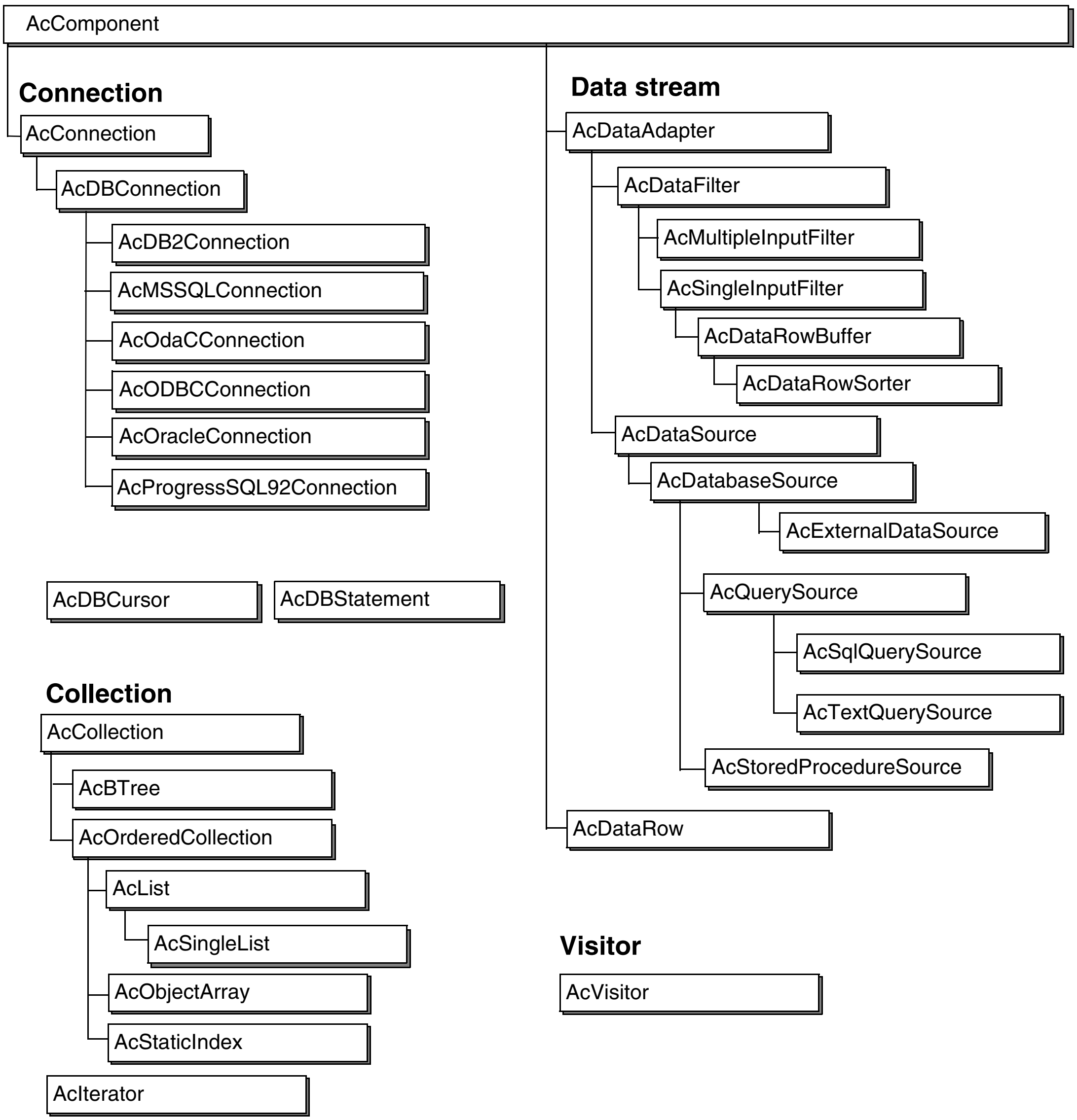
Chapter

4

Actuate Foundation Class library

This chapter covers the topic "Summary of classes and methods."


AcComponent
AcReportComponent
AcVisualComponent
Report structure
AcReport
AcSection
AcConditionalSection
AcDataSection
AcGroupSection
AcReportSection
AcParallelSection
AcSequentialSection
Excel
AcExcelObject
AcExcelApp
AcExcelRange
AcExcelCell
AcExcelColumn
AcExcelRow
AcExcelWorkbook
AcExcelWorksheet
Controls
AcControl
AcCrosstab
AcDrawing
AcChart
AcImageControl
AcLineControl
AcRectangleControl
AcTextualControl
AcBrowserScriptingControl
AcDataControl
AcCurrencyControl
AcDateTimeControl
AcDoubleControl
AcDynamicTextControl
AcIntegerControl
AcTextControl
AcLabelControl
AcPageNumberControl
Page layout
AcBaseFrame
AcBasePage
AcPage
AcSubPage
AcDataFrame
AcFrame
AcFlow
AcLinearFlow
AcTopDownFlow
AcPageList
AcLeftRightPageList
AcSimplePageList
AcTitleBodyPageList
Drawing plane
AcDrawingPlane
AcDrawingChartPlane
AcSVGDrawingPlane
AcChartAxis
AcChartCategory
AcChartGridLine
AcChartLayer
AcChartPoint
AcChartPointStyle
AcChartSeriesStyle
AcChartSeries
AcChartTrendline


AcComponent
Connection
AcConnection
AcDBConnection
AcDB2Connection
AcMSSQLConnection
AcOdaCConnection
AcODBCConnection
AcOracleConnection
AcProgressSQL92Connection
AcDBCursor
AcDBStatement
Collection
AcCollection
AcBTree
AcOrderedCollection
AcList
AcSingleList
AcObjectArray
AcStaticIndex
AcIterator
Data stream
AcDataAdapter
AcDataFilter
AcMultipleInputFilter
AcSingleInputFilter
AcDataRowBuffer
AcDataRowSorter
AcDataSource
AcDatabaseSource
AcExternalDataSource
AcQuerySource
AcSqlQuerySource
AcTextQuerySource
AcStoredProcedureSource
AcDataRow
Visitor
AcVisitor

Summary of classes and methods

The Actuate Foundation Class (AFC) library, afc.rol, contains classes and methods that support building a wide range of custom reports. This chapter provides an overview of the AFC library class by class, along with the methods for each class. The classes and methods change from time to time as the product architecture changes to meet customer needs.

This chapter groups classes into the following categories:

- Report structure
- Page layout
- Control
- Connection
- Collection
- Data stream
- Excel
- Visitor

Within each category, classes are arranged in tables according to the class hierarchy. For example, the report structure class table begins with the root class, AcComponent, and with AcReportComponent, the class that derives from AcComponent.

The methods for a class appear in alphabetical order in each class table. These methods are either callable or overridable. Callable methods provide useful functionality, such as returning a reference to a component or identifying a component as transient or persistent. You should not attempt to override callable methods. Overridable methods are those you can modify to change class functionality.

Some classes are appropriate for customizing a report and other classes are not. For example, you should never instantiate the abstract base classes that define the core protocol, the rules governing the use of a class.

In the tables in this chapter, inherited methods and methods that define the core protocol typically appear only in a single class table. For example, ApplyVisitor( ), which applies to many subclasses of AcComponent, is defined only in the AcComponent class. New( ), which is part of the core protocol, appears in the AcComponent class but not in classes that inherit from AcComponent. Exceptions to this convention are methods whose functionality substantially changes in a subclass. For example, the core method BuildFromRow( ) is defined once in the AcReportComponent class and again in AcChart, where BuildFromRow( ) supports specialized functionality for charts.

Report structure classes and methods

Use report structure classes and methods to define the structural components of a report, including the topmost container report object and the report sections.

AcComponent

AcComponent is the root class for report components. All structural components derive from AcComponent. This class defines the mechanism for creating objects within container objects. AcComponent methods are listed in Table 4-1.

Table 4-1 AcComponent methods

Method	Classification	Type	Description
ApplyVisitor( )	Callable	AcVisitor	Starts visitor functions for a component.
Delete( )	Overridable	N/A	Destructor.
IsPersistent( )	Callable	Boolean	Returns True if the component is persistent. Returns False if the component is transient.
New( )	Overridable	N/A	Constructor method for this class.

AcReportComponent

A subclass of AcComponent, AcReportComponent is the base class for all sections, pages, frames, and controls. AcReportComponent defines the general structural characteristics of all classes in which a report stores persistent objects. AcReportComponent methods are listed in Table 4-2.

Table 4-2 AcReportComponent methods

Method	Classification	Type	Description
Abandon( )	Callable	N/A	Removes a component that the report no longer needs.
AddContent( )	Callable	N/A	Adds a new content component to the current component.
Build( )	Overridable	N/A	Builds components that do not use data rows. Container components override this method.
BuildFromRow( )	Overridable	AcBuild Status	Builds components that use data rows. Data container objects override this method.

(continues)

Table 4-2 AcReportComponent methods (continued)

Method	Classification	Type	Description
DetachContent( )	Callable	N/A	A container object, such as a frame, calls this method to detach one of its content objects, such as a control.
DetachFromContainer( )	Callable	N/A	A content object, such as a control, calls this method to detach the content object from its container, such as a frame.
FindContainerByClass( )	Callable	AcReport Component	Returns a reference to this class in the structure hierarchy.
FindContentByClass( )	Callable	AcVisual Component	Returns a content component by the component's class name. Derived classes override this method to implement a specific search method.
Finish( )	Overridable		Prepares the component to be written to the report object instance (.roi) file. Called when a component is finished building.
GenerateXML( )	Overridable	N/A	Generates XML for components with custom XML.
GetComponentACL( )	Overridable	String	Returns the access control list (ACL) for the component.
GetConnection( )	Callable	Ac Connection	Returns the connection associated with this component.
GetContainer( )	Callable	AcReport	Returns a reference to the container object for this component.
GetContentCount( )	Callable	Integer	Returns the number of content items in a component.
GetContentIterator( )	Callable	AcIterator	Returns an iterator over the contents of this component.
GetContents( )	Callable	AcOrdered Collection	Returns a handle to the collection of contents for this component.
GetDataStream( )	Callable	AcData Adapter	Returns the data stream associated with this component.
GetFirstContent( )	Callable	AcReport Component	Gets the first content component.

Table 4-2 AcReportComponent methods (continued)

Method	Classification	Type	Description
GetFirstContent Frame( )	Callable	AcFrame	Gets the first content frame for a component.
GetFlow( )	Callable	AcFlow	Returns a handle to the flow of this component.
GetFullACL( )	Overridable	String	Returns the access control list (ACL) for the component and each of its containers in the report structure hierarchy.
GetPage( )	Callable	AcPage	Returns the page that contains the object.
GetPageIndex( )	Callable	Integer	Returns the page index of the page that contains the object. The page index identifies the position of the page within the report, starting with 1.
GetPageList( )	Callable	AcPageList	Returns the page list associated with the report that contains this component.
GetReport( )	Callable	AcReport	Returns the report that contains this component.
GetRowCount( )	Callable	Integer	Returns the number of rows that this component has processed.
GetSearchTag( )	Overridable	String	Returns the value of the SearchTag property.
GetTocEntry( )	Overridable	String	Returns the text of the component's table of contents entry.
GetVisiblePageIndex( )	Callable	Integer	Returns the visible page number of the page that contains the object.
GetXMLText( )	Overridable	String	Returns the value of a control that has the XMLType property set to XMLText.
HasContents( )	Callable	Boolean	Returns True if this component contains at least one content component.
IsContainer( )	Callable	Boolean	Returns True if this component can hold content components.

(continues)

Table 4-2 AcReportComponent methods (continued)

Method	Classification	Type	Description
IsFlow( )	Callable	Boolean	Checks whether this component is a flow.
IsFrame( )	Callable	Boolean	Checks whether the component is a frame.
IsLeaf( )	Callable	Boolean	Returns True if this component cannot contain a content component.
IsPage( )	Callable	Boolean	Returns True if this component is a page.
IsSubpage( )	Callable	Boolean	Returns True if this component is a subpage.
IsVisual( )	Callable	Boolean	Checks whether the component is visual.
OnRow( )	Overridable	N/A	Displays values from a single row. Called for each new row.
SetSearchTag( )	Overridable	N/A	Sets the value of the SearchTag property. SearchTag uniquely identifies a component when a report design contains multiple instances of the same component.
SetTocEntry( )	Overridable	N/A	Sets the name of the table of contents entry for a component.
Start( )	Overridable	N/A	Prepares a component for build operations. Called when a component begins building.

AcReport

A subclass of AcReportComponent, AcReport is the root object in a report object instance (.roi) file. Methods for this class determine whether the ROI is temporary, how the report interacts with the viewing or printing environment, whether the report uses page-level security, and how to set privileges for a burst report. Other methods provide information about the report, such as the layout orientation, the page list, and the locale to use for report generation, viewing, or printing. AcReport methods are listed in Table 4-3.

Table 4-3 AcReport methods

Method	Classification	Type	Description
GetContent( )	Callable	AcReport Component	Returns the component in the Content slot of the root report component.
GetCustomFormat( )	Overridable	N/A	Retrieves the generated Excel file from the View process.
GetFactoryLocale( )	Overridable	String	Specifies the locale to use for report generation.
GetGlobalDHTML Code( )	Overridable	String	Returns the custom code from a browser scripting control and makes it available to every DHTML page the DHTML converter generates.
GetLanguage( )	Callable	String	Returns the report's language.
GetLayoutOrientation( )	Overridable	AcLayout Orientation	Returns the report's layout orientation, either right-to-left or left-to-right.
GetPrintLocale( )	Overridable	String	Returns the locale to use for printing a report on iServer.
GetReport( )	Overridable	AcReport	Returns a reference to the root report component.
GetUserACL( )	Overridable	String	Returns the access control list (ACL) for the current user.
GetViewLocale( )	Overridable	String	Returns the locale to use for report viewing.
HasPageSecurity( )	Callable	Boolean	Returns True if the report uses page-level security.
NewContent( )	Overridable	AcReport Component	Creates a component in the top-level Content slot.
NewPageList( )	Overridable	AcPageList	Creates the page list for the report.
OnFinishPrint( )	Overridable	N/A	Override this method to perform tasks after printing, such as logging or sending a notification.
OnStartPrint( )	Overridable	N/A	Called at the start of a print operation to perform custom tasks.

(continues)

Table 4-3 AcReport methods (continued)

Method	Classification	Type	Description
RoiIsTemporary( )	Overridable	Boolean	Specifies whether to delete the report object instance (.roi) file after the report runs. The default setting is True.
SetBurstReport Privileges( )	Overridable	N/A	Sets privileges for a burst report. By default, the burst report has the same privileges as the original report.
SetGlobalDHTML Code( )	Callable	String	Sets the custom code in a browser scripting control.
SetLayoutOrientation( )	Callable	N/A	Sets the report layout to either right-to-left or left-to-right orientation.
SetROIAging Properties( )	Overridable	N/A	Sets autoarchive rules for an ROI file.
SuggestRoiName( )	Overridable	String	Called to suggest the ROI name for this report. Useful for naming the output of batch reports, for example.
TocAddComponent( )	Callable	AcTocNode Type	Adds the report to the table of contents.
XMLDataProlog( )	Overridable	N/A	Creates the XML prolog.

AcSection

A subclass of AcReportComponent, AcSection defines the characteristics of all non-visual structural classes, including report sections, group sections, and parallel and sequential sections. Derived classes represent different ways of grouping data. AcSection methods are listed in Table 4-4.

Table 4-4 AcSection methods

Method	Classification	Type	Description
CommittedToFlow( )	Overridable	N/A	Called by the page list for each section assigned to a flow. After the section is committed to a flow, you can override this method to perform custom processing.

Table 4-4 AcSection methods (continued)

Method	Classification	Type	Description
DeletePageFrame( )	Callable	AcFrame	Called by a section to delete a frame.
FinishConnection( )	Overridable	N/A	Closes the connection for this section. Override this method if you do not want the section to close the connection.
FinishFlow( )	Overridable	N/A	Called at the end of each flow to support cleanup tasks and other custom functionality.
FinishPage( )	Overridable	N/A	Called at the end of a new page to support custom functionality.
GetCurrentRow( )	Callable	AcDataRow	Returns the current data row.
GetSearchValue( )	Overridable	String	Returns the value of the SearchValueExp property for the section.
NewPage( )	Overridable	AcPage	Determines which page type to use in this section. Page types include Letter, Legal, A4, A5, B4, B5, and custom types.
ObtainConnection( )	Overridable	Ac Connection	Creates a connection for this section. Override this method if you want to use other than the default mechanism to get a connection.
PageBreakAfter( )	Overridable	Boolean	Returns True if the PageBreakAfter property is set.
PageBreakBefore( )	Overridable	Boolean	Returns True if the property, PageBreakBefore, is set.
SetSearchValue( )	Overridable	AcDataRow	Sets the search value for use in searching a report, activating a hyperlink, or generating a reportlet from a report.
SetSecurity( )	Overridable	AcDataRow	Sets the ACL for the section. Override this method to build a custom ACL for the section.

(continues)

Table 4-4 AcSection methods (continued)

Method	Classification	Type	Description
StartFlow( )	Overridable	Boolean	Called at the beginning of each new flow to support custom functionality.
StartPage( )	Overridable	N/A	Called at the beginning of each new page to support custom functionality.
StopAfterCurrent Frame( )	Callable	N/A	Stop processing after the current frame is added to the page.
StopAfterCurrentRow( )	Callable	N/A	Stops processing after the current data row is complete.
StopNow( )	Callable	N/A	Stops processing a data row immediately.
TocAddComponent( )	Callable	AcTocNode Type	Adds the section to the table of contents.
TocAddContents( )	Callable	Boolean	If True, adds the contents of the section to the table of contents.

AcConditionalSection

AcConditionalSection is a subclass of AcSection. AcConditionalSection defines a section that displays content based on a condition the report developer sets. For example, a conditional section can display values greater than 500 in red text. ConditionIsTrue( ) indicates whether the condition exists. The AcConditionalSection method is described in Table 4-5.

Table 4-5 AcConditionalSection methods

Method	Classification	Type	Description
ConditionIsTrue( )	Overridable	Boolean	Returns the Boolean value that indicates whether a condition for displaying the section’s contents is True

AcDataSection

AcDataSection is a subclass of AcSection. A data section is either a report section or a group section. You can use the methods for AcDataSection to retrieve components in the section, such as the page header or footer, or to instantiate a component in the section's Before, After, Content, PageFooter, or PageHeader slot. AcDataSection methods are listed in Table 4-6.

Table 4-6 AcDataSection methods

Method	Classification	Type	Description
GetAfter( )	Callable	AcReport Component	Returns the After component, if any, for this data section.
GetBefore( )	Callable	AcReport Component	Returns the Before component, if any, for this data section.
GetFirstPageFooter( )	Overridable	AcBase Frame	Returns the first page footer, if any, for this data section.
GetFirstPageHeader( )	Overridable	AcBase Frame	Returns the first page header, if any, for this data section.
GetPageFooter( )	Callable	AcBase Frame	Returns the current page footer, if any, for this page.
GetPageHeader( )	Callable	AcBase Frame	Returns the current page header, if any, for this page.
NewAfter( )	Overridable	AcReport Component	Instantiates a component in the After slot.
NewBefore( )	Overridable	AcReport Component	Instantiates a component in the Before slot.
NewContent( )	Overridable	AcReport Component	Instantiates a component in the Content slot.
NewPageFooter( )	Overridable	AcBase Frame	Instantiates a component in the PageFooter slot.
NewPageHeader( )	Overridable	AcBase Frame	Instantiates a component in the PageHeader slot.
OnEmptyGroup( )	Overridable	N/A	Called if the section processes no rows. Passes the information that no row exists. Also supports logging and other custom functionality.

AcGroupSection

AcGroupSection is a subclass of AcDataSection. A group section defines a group as a set of data rows that have the same key value, such as data rows with a state field value of CA. A group section is an organizational tool that does not process data rows. Use the public methods for this class to retrieve the group key value, check whether the key value is still valid, and create a table of contents label for the group section. AcGroupSection methods are listed in Table 4-7.

Table 4-7 AcGroupSection methods

Method	Classification	Type	Description
GetKeyString( )	Callable	String	Returns the key value for a group
IsSameKey( )	Overridable	Boolean	Checks whether the group section key has changed

AcParallelSection

AcParallelSection is a subclass of AcSection. A parallel section contains two or more reports that appear side by side on a page in separate flows. When you create a parallel section, you add multiple report sections to it and assign each section to a separate flow. The AcParallelSection method is described in Table 4-8.

Table 4-8 AcParallelSection methods

Method	Classification	Type	Description
AddReport( )	Callable	N/A	Adds a subreport to the Reports slot of a parallel section

AcReportSection

AcReportSection is a subclass of AcDataSection. A report section opens a data connection and retrieves rows from a data source. Use the public methods for AcReportSection to locate, open, close, or create the data stream for the section. You can also set the sorting key and create a label for the table of contents entry for a report section. AcReportSection methods are listed in Table 4-9.

Table 4-9 AcReportSection methods

Method	Classification	Type	Description
FinishDataStream( )	Overridable	N/A	Closes the data stream for this report section.
NewDataStream( )	Overridable	AcData Adapter	Instantiates the component in the DataStream slot of the report section. Override this method to customize the data adapter that the report instantiates and opens.
ObtainDataStream( )	Overridable	AcData Adapter	Creates the data stream to use for this report section. Override this method to reuse an existing data stream. This method does not also open the data stream.

Table 4-9 AcReportSection methods

Method	Classification	Type	Description
SetSortKey( )	Overridable	N/A	Sets the sort key for the data adapter. The default behavior for this method sets the sort key to the column specified in the Key property for any group sections in this report.
StartDataStream( )	Overridable		Opens the data stream.

AcSequentialSection

AcSequentialSection is a subclass of AcSection. A sequential section contains two or more reports that run or print one after the other. The reports appear in the same flow. AcSequentialSection methods are listed in Table 4-10.

Table 4-10 AcSequentialSection methods

Method	Classification	Type	Description
NewContent( )	Overridable	AcReport Component	Instantiates one of the list of contents for this section
SelectContent( )	Overridable	Boolean	Indicates whether to use a content component as report output
StopAfterCurrent Section( )	Callable	N/A	Stops the current section after the current nested section terminates

Page layout classes and methods

Use page layout classes and methods to customize building frames, flows, pages, and page lists.

AcBaseFrame

AcBaseFrame is a subclass of AcVisualComponent. AcBaseFrame defines the general characteristics of frames and pages and the logic for instantiating the content in frames and pages. AcBaseFrame methods are listed in Table 4-11.

Table 4-11 AcBaseFrame methods

Method	Classification	Type	Description
AddToAdjustSizeList( )	Overridable	N/A	Adds a component to its container's list of components to resize.

(continues)

Table 4-11 AcBaseFrame methods (continued)

Method	Classification	Type	Description
BindToFlow( )	Overridable	N/A	Called when the framework adds a frame to a flow on a page.
FindContentByClass ID( )	Callable	AcVisual Component	Locates one of the frame's content components using the class ID of that component.
GetControl( )	Callable	AcControl	Locates a named control.
GetControlValue( )	Callable	Variant	Returns the value of a data control within the frame.
GetPageNumber( )	Callable	String	Returns the formatted page number for a page.
GetSearchValue( )	Overridable	String	Differentiates between subclasses of a parent class when a user is searching for values, activating a hyperlink, or generating reportlet content from a report.
IsDataFrame( )	Callable	Boolean	Indicates whether the component is a data frame.
IsFooter( )	Callable	Boolean	Indicates whether the component is a footer.
IsHeader( )	Callable	Boolean	Indicates whether the component is a header.
MakeContents( )	Overridable	N/A	Creates the frame contents dynamically when specific conditions are present.
RebindToFlow( )	Overridable	N/A	The framework calls this method for controls in a subpage when the subpage's BalanceFlows property is True.
SearchAttributeName( )	Overridable	String	The name of an attribute on which to base a search.

AcBasePage

A subclass of AcBaseFrame, AcBasePage is an abstract base class that defines the logic for instantiating the contents of a page. AcBasePage methods are listed in Table 4-12.

Table 4-12 AcBasePage methods

Method	Classification	Type	Description
BalanceFlows( )	Overridable	Boolean	Implements the BalanceFlows property. This property specifies whether to redistribute the contents of the page to make all flows on the page the same height. The default value is False.
GetFirstDataFrame( )	Callable	AcFrame	Retrieves the first data frame on a page.
GetLastDataFrame( )	Callable	AcFrame	Retrieves the last data frame on a page.

AcPage

AcPage is a subclass of AcBasePage that represents pages in a report. Use AcPage methods to get information about a page, such as the page number, or to indicate whether the page uses dynamic geometry. You also can indicate how to position data from a dynamic text control when the data splits across multiple pages. AcPage methods are listed in Table 4-13.

Table 4-13 AcPage methods

Method	Classification	Type	Description
FormatPageNumber( )	Overridable	String	Returns the formatted page number. Override this method if your formatting requires writing code.
GetVisiblePageIndex( )	Callable	Integer	Returns the index for visible pages.
SplitMarginBottom( )	Overridable	AcTwips	Implements the SplitMarginBottom property. When a dynamic text control can split to fit on multiple pages, SplitMarginBottom sets a blank space between the bottom edge of a page and its contents.
SplitMarginLeft( )	Overridable	AcTwips	Implements the SplitMarginLeft property. When a dynamic text control can split to fit on multiple pages, SplitMarginLeft sets a blank space between the left edge of a page and its contents.

(continues)

Table 4-13 AcPage methods (continued)

Method	Classification	Type	Description
SplitMarginRight( )	Overridable	AcTwips	Implements the SplitMarginRight property. When a dynamic text control can split to fit on multiple pages, SplitMarginRight sets a blank space between the right edge of a page and its contents.
SplitMarginTop( )	Overridable	AcTwips	Implements the SplitMarginTop property. When a dynamic text control can split to fit on multiple pages, SplitMarginTop sets a blank space between the top edge of a page and its contents.

AcSubPage

AcSubpage is a subclass of AcBasePage. AcSubpage supports dynamically switching from one column to two columns on the same page. There are no public methods defined specifically for this class.

AcDataFrame

AcDataFrame is a subclass of AcBaseFrame. AcDataFrame is an abstract base class that defines the logic for how frames work with data rows. There are no public classes defined specifically for this class.

AcFrame

A subclass of AcDataFrame, AcFrame is the base class for frames in a report design. The methods in this class support changing the size of the frame, creating custom code for a web page, setting the relationship between a frame and a page, splitting a frame across multiple pages, and so on. AcFrame methods are listed in Table 4-14.

Table 4-14 AcFrame methods

Method	Classification	Type	Description
AutoSplitVertical( )	Overridable	AcAutoSplit	Returns the value of the AutoSplitVertical property. AutoSplitVertical specifies how the Factory splits a frame or a dynamic text control.
CustomDHTML Footer( )	Overridable	String	Supports custom browser code as a footer in an HTML form.

Table 4-14 AcFrame methods (continued)

Method	Classification	Type	Description
CustomDHTMLHeader( )	Overridable	String	Supports custom browser code as a header in an HTML form.
GetBorderOrigin( )	Callable	AcPoint	Returns the origin, or upper left coordinates, of the border.
GetBorderRect( )	Callable	AcRectangle	Returns the the rectangle that defines the border.
GetBorderSize( )	Callable	AcSize	Returns the size of the border.
NoSplitBottom( )	Overridable	AcTwips	Returns the value of the NoSplitBottom property. NoSplitBottom specifies the height of the area that must not be split at the bottom of the frame, or the minimum height of the last segment.
NoSplitTop( )	Overridable	AcTwips	Returns the value of the NoSplitTop property. NoSplitTop specifies the height of the area that must not be split at the top of the frame, or the minimum height of the first segment. Applies only to a frame that contains at least one dynamic text control that splits across multiple pages.
PageBreakAfter( )	Callable	Boolean	Returns the value of a frame's PageBreakAfter property. If PageBreakAfter is True, a new page begins immediately after the frame. Applies only to frames in Before, Content, or After slots.
PageBreakBefore( )	Callable	Boolean	Returns the value of a frame's PageBreakBefore property. If PageBreakBefore is True, the frame appears at the top of a new page. Applies only to frames in Before, Content, or After slots.

(continues)

Table 4-14 AcFrame methods (continued)

Method	Classification	Type	Description
SplitMarginBottom( )	Overridable	AcTwips	Returns the value of the SplitMarginBottom property. SplitMarginBottom specifies a blank area between the bottom edge of each segment, except the last, and its contents. Applies only to a frame that contains at least one dynamic text control that splits across multiple pages.
SplitMarginTop( )	Overridable	AcTwips	Returns the value of the SplitMarginTop property. SplitMarginTop specifies a blank area between the top edge of each segment, except the first, and its contents. Applies only to a frame that contains at least one dynamic text control that splits across multiple pages.

AcFlow

AcFlow is a subclass of AcVisualComponent. AcFlow defines the logic for placing frames in a flow, adding components to a flow, making adjustments to the size of a flow, and other tasks related to flows. AcFlow methods are listed in Table 4-15.

Table 4-15 AcFlow methods

Method	Classification	Type	Description
AddFooter( )	Overridable	Boolean	Adds a footer frame to the flow.
AddFrame( )	Overridable	N/A	Adds a frame to the flow at the next available position in the report.
AddHeader( )	Overridable	Boolean	Adds a header frame to the flow.
AddSubpage( )	Overridable	Boolean	Adds a subpage to the flow.
AdjustFooter( )	Overridable	N/A	Adjusts the space available for a page footer within the flow.
CanFitFrame( )	Callable	Boolean	Checks whether the flow contains enough space to accommodate a specific frame.
CanFitHeight( )	Overridable	Boolean	Checks whether the flow can contain a specific component.

Table 4-15 AcFlow methods

Method	Classification	Type	Description
GetFirstDataFrame( )	Callable	AcFrame	Returns the first data frame associated with the current flow.
GetFreeSpace( )	Overridable	AcSize	Returns the unused space in the flow.
GetInsideSize( )	Callable	AcSize	Returns the size of the content rectangle.
GetLastDataFrame( )	Callable	AcFrame	Returns the last data frame associated with the current flow.
IsEmpty( )	Overridable	Boolean	Indicates whether the flow contains a data frame, such as a Content, Before, or After frame.
ReleaseSpace( )	Overridable	N/A	Releases reserved space back to the flow.
ReserveSpace( )	Overridable	N/A	Reserves a part of the available space within the flow.
ResetSpace( )	Overridable	N/A	Calls ResizeByConstrained( ) from AcVisualComponent. Resets the available space in a flow in response to a change in the flow’s contents.
ResizeByConstrainedBy Contents( )	Callable	N/A	Resets the amount of space in the flow to zero.
ShiftFooterUp( )	Overridable	N/A	Moves the footer up so the footer appears immediately after the last frame in the flow.

AcLinearFlow

A subclass of AcFlow, AcLinearFlow is the abstract base class for working with a flow that fills in one direction, either from top to bottom or from left to right. AcLinearFlow methods are listed in Table 4-16.

Table 4-16 AcLinearFlow methods

Method	Classification	Type	Description
GetFreeSpace( )	Callable	AcSize	Returns the unused space in the flow

(continues)

Table 4-16 AcLinearFlow methods (continued)

Method	Classification	Type	Description
GetInsideOrigin( )	Callable	AcPoint	Gets the position of the inside area of the flow, relative to the upper left corner of the frame
GetInsideRect( )	Callable	AcRectangle	Gets the rectangle that defines the inside space of the flow, relative to the upper left corner, or origin, of the frame
GetInsideSize( )	Callable	AcSize	Returns the size of the content rectangle

AcTopDownFlow

A subclass of AcLinearFlow, AcTopDownFlow defines the logic for adding frames to a flow that fills from top to bottom only. The AcTopDownFlow method is described in Table 4-17.

Table 4-17 AcTopDownFlow methods

Method	Classification	Type	Description
AdjustFooter( )	Callable	AcFrame	Adjusts the top of a page footer to allow for size changes

AcPageList

AcPageList is a subclass of AcReportComponent that instantiates and holds the pages for a report. AcPageList is an abstract class that defines the logic for building pages and managing data display. AcPageList methods are listed in Table 4-18.

Table 4-18 AcPageList methods

Method	Classification	Type	Description
AddFrame( )	Callable	N/A	Adds a frame to a page list. Places the frame in a flow on a page.
EjectPage( )	Callable	N/A	Finishes the currently active page.
GetCurrentFlow( )	Callable	AcFlow	Returns the active flow on the current page.
GetCurrentPage( )	Callable	AcPage	Returns the current page in the page list.
GetCurrentPage ACL( )	Callable	String	Returns the ACL for the current page in the page list.

Table 4-18 AcPageList methods

Method	Classification	Type	Description
GetEstimatedPageCount( )	Overridable	Integer	Provides an estimate of the number of pages a report will contain.
GetFirstPage( )	Callable	AcPage	Returns the first page in the page list.
GetLastPage( )	Callable	AcPage	Returns the last page in the page list.
GetPageCount( )	Callable	Integer	Returns the number of total pages in the page list.
HasPageSecurity( )	Callable	Boolean	Indicates whether the page uses page-level security.
NeedCheckpoint( )	Overridable	Boolean	Override this method to control how frequently to flush persistent objects to the report object instance (.roi) file.
NeedHeight( )	Callable	N/A	Ensures that a specified amount of vertical space is available in the current flow, and if not, starts a new flow.
NewPage( )	Overridable	AcPage	An empty method that derived classes override to instantiate a new page.
UseAcceleratedCheckpoints( )	Overridable	Boolean	Creates additional page checkpoints in the ROI file.

AcLeftRightPageList

AcLeftRightPageList is a subclass of AcPageList. AcLeftRightPageList provides a report format that has alternating left and right pages. There are no public methods defined specifically for this class.

AcSimplePageList

AcSimplePageList is a subclass of AcPageList. AcSimplePageList provides a report style in which all pages have the same layout. There are no public methods defined specifically for this class.

AcTitleBodyPageList

AcTitleBodyPageList is a subclass of AcPageList. AcTitleBodyPageList provides a report style in which the title page is different from the body pages.

Control classes and methods

Use control classes and methods to manipulate the position and value of visual controls.

AcVisualComponent

AcVisualComponent is a subclass of AcReportComponent. AcVisualComponent is the base class that defines the characteristics of all visual classes, such as frames, charts and other controls, pages, and flows. Derived classes display data or graphical elements. AcVisualComponent methods are listed in Table 4-19.

Table 4-19 AcVisualComponent methods

Method	Classification	Type	Description
AdjustHorizontal Geometry( )	Overridable	N/A	Adjusts the width and horizontal position of the object relative to its reference object
AdjustSize( )	Overridable	N/A	Changes the size of the component
AdjustVertical Geometry( )	Overridable	N/A	Adjusts the height and vertical position of the object relative to its reference object
CanIncreaseHeight( )	Callable	Boolean	Implements the CanIncreaseHeight property
CanIncreaseWidth( )	Callable	Boolean	Implements the CanIncreaseWidth property
CanMoveLeft( )	Callable	Boolean	Implements the CanMoveLeft property
CanMoveUp( )	Callable	Boolean	Implements the CanMoveUp property
CanReduceHeight( )	Callable	Boolean	Implements the CanReduceHeight property
CanReduceWidth( )	Callable	Boolean	Implements the CanReduceWidth property
CanSplitVertically( )	Overridable	Boolean	Determines whether an object can split across multiple pages
ComputeLowestSplit( )	Callable	Boolean	Determines the lowest point at which an object can split across multiple pages
FindLowestSplit( )	Overridable	Boolean	Establishes the vertical point at which the object can split

Table 4-19 AcVisualComponent methods (continued)

Method	Classification	Type	Description
FindPageContainer ByClass( )	Callable	AcReport Component	Returns a reference to this class in the page hierarchy
GetBottom( )	Callable	Integer	Returns the position of the bottom of the component, in twips, relative to the top of its container frame
GetFirstSlave( )	Callable	AcVisual Component	Returns the handle to the object's first slave object
GetFrame( )	Callable	AcFrame	Returns a reference to the frame containing the visual object
GetHeight( )	Callable	Integer	Returns the height of the component
GetLastSlave( )	Callable	AcVisual Component	Returns the handle to the object's last slave object
GetLeft( )	Callable	Integer	Returns the position of the left edge of the component
GetLinkTo( )	Callable	String	Returns the value of the hyperlink expression in the LinkTo variable
GetMaster( )	Callable	AcVisual Component	Returns the handle to the object's master object
GetPageContainer( )	Callable	AcVisual Component	Returns the container in the page hierarchy for the component
GetPixelSize( )	Callable	AcSize	Gets the size of the component in pixels
GetRect( )	Callable	AcRectangle	Returns the coordinates of the component relative to its frame
GetRight( )	Callable	Integer	Returns the position of the right edge of the component
GetTop( )	Callable	Integer	Returns the position of the top of the component
GetVisualComponent( )	Callable	AcVisual Component	Returns the current visual component
GetWidth( )	Callable	Integer	Returns the width of the component
HorizontalPosition( )	Callable	AcHorizontal Position	Implements the HorizontalPosition property

(continues)

Table 4-19 AcVisualComponent methods (continued)

Method	Classification	Type	Description
HorizontalSize( )	Callable	AcHorizontal Size	Implements the HorizontalSize property
IsFirstSlave( )	Callable	Boolean	Determines whether the object is the first slave of the master object
IsFrameDecoration( )	Callable	Boolean	Determines whether the object is a frame decoration
IsLastSlave( )	Callable	Boolean	Determines whether the object is the last slave of the master object
IsMaster( )	Callable	Boolean	Determines whether the object is a master object
IsNormal( )	Callable	Boolean	Returns True if the object is neither a master nor a slave object
IsSlave( )	Callable	Boolean	Determines whether the object is a slave object
IsVisible( )	Callable	Boolean	Determines whether the component is visible to the user
MaximumHeight( )	Callable	Boolean	Implements the MaximumHeight property
MaximumWidth( )	Callable	Boolean	Implements the MaximumWidth property
MinimumHeight( )	Callable	Boolean	Implements the MinimumHeight property
MinimumWidth( )	Callable	Boolean	Implements the MinimumWidth property
MoveBy( )	Callable	N/A	Moves the component by the amount given
MoveByConstrained( )	Callable	N/A	Specifies the distance by which to move the component
MoveTo( )	Callable	N/A	Moves the component to the position given
MoveToConstrained( )	Callable	N/A	Moves the component
ResizeBy( )	Callable	N/A	Resizes a component by the distances given
ResizeByConstrained( )	Callable	N/A	Specifies the amount by which to resize the component

Table 4-19 AcVisualComponent methods (continued)

Method	Classification	Type	Description
ResizeTo( )	Callable	N/A	Resizes a frame or control to the given size
ResizeToConstrained( )	Callable	N/A	Resizes the component to the given size
Searchable( )	Callable	AcSearch Type	Implements the Searchable property
SearchAlias( )	Callable	String	Implements the SearchAlias property
Selectable( )	Overridable	Boolean	Implements the Selectable property
SplitVertically( )	Overridable	N/A	Splits visual components vertically across pages
StatusText( )	Callable	String	Returns the value of GetLinkTo( ) if there is a hyperlink
TargetWindowName( )	Callable	String	Implements the TargetWindowName property
VerticalPosition( )	Callable	AcVertical Position	Returns the value of the VerticalPosition property
VerticalSize( )	Callable	AcVertical Size	Implements the VerticalSize property

AcControl

A subclass of AcVisualComponent, AcControl defines the general characteristics of all controls. AcControl methods are listed in Table 4-20.

Table 4-20 AcControl methods

Method	Classification	Type	Description
BalloonHelp( )	Overridable	String	Returns the text to display when a user hovers the mouse pointer over a control.
GetControlValue( )	Callable	Variant	Returns the value of another control within the same frame.
GetText( )	Overridable	String	Formats the value of a data control or label for display.

(continues)

Table 4-20 AcControl methods (continued)

Method	Classification	Type	Description
GetXMLText( )	Overridable	String	Returns the value of a control that has the XMLType property set to XMLText.
GetValue( )	Callable	Variant	Returns the value of the DataValue variable for a data control.
IsSummary( )	Overridable	Boolean	Use IsSummary( ) to determine whether the control processes a single row or multiple rows.
PageNo( )	Callable	Integer	Returns the position of the page in the report, starting from 1.
PageNo$( )	Callable	String	Returns the formatted page number of the control as a string. For example, to show the formatted page number such as vi, 107, or 12-5 in a control, set the value of the control to PageNo$.
SetDataValue( )	Callable	N/A	Sets the value for a data control within the same frame.

AcCrosstab

AcCrosstab is a subclass of AcControl. Use AcCrosstab to display data in spreadsheet format in an Actuate Basic report. AcCrosstab methods are listed in Table 4-21.

Table 4-21 AcCrosstab methods

Method	Classification	Type	Description
FinishBuilding( )	Overridable	N/A	Finishes building the data collector. Creates and populates the visual data structure.

AcDrawing

AcDrawing is a subclass of AcControl and the parent class of AcChart. Use this class to display a drawing. AcDrawing methods are listed in Table 4-22.

Table 4-22 AcDrawing methods

Method	Classification	Type	Description
AddDrawingPlane( )	Callable	N/A	Adds a drawing plane to he end of a drawing's list of drawing planes

Table 4-22 AcDrawing methods

Method	Classification	Type	Description
GetAntialias( )	Callable	Boolean	Determines whether a drawing will be rendered with antialiasing
GetBackgroundColor( )	Callable	AcColor	Returns the background color of a drawing
GetDrawingPlane( )	Callable	AcDrawing Plane	Returns a reference to the specified drawing plane within a drawing
GetNumberOfDrawing Planes( )	Callable	Integer	Determines the number of drawing planes in a drawing
GetRenderIn24Bit Color( )	Callable	Boolean	Determines whether a drawing will be rendered in 24-bit color
InsertDrawingPlane( )	Callable	AcDrawing Plane	Inserts a drawing plane at a specific position within a drawing's list of drawing planes
RemoveDrawing Plane( )	Callable	N/A	Removes a drawing plane from a drawing
RenderToFile( )	Callable	N/A	Renders a drawing into a file
SetAntialias( )	Callable	N/A	Specifies whether a drawing will be rendered with antialiasing
SetRenderIn24Bit Color( )	Callable	N/A	Specifies whether a drawing will be rendered in 24-bit color

AcChart

A subclass of AcDrawing, AcChart builds a data structure of objects that represent the various elements of a chart, such as axes, categories, and points. AcChart methods are listed in Table 4-23.

Table 4-23 AcChart methods

Method	Classification	Type	Description
AdjustChart( )	Callable	N/A	Override this method to make final adjustments to a chart after all its automatic layout has been created.
BaseAndOverlayScales AreMatched( )	Callable	Boolean	Returns True if the base and overlay *y*-axis scales of a chart are forced to be identical.

(continues)

Table 4-23 AcChart methods (continued)

Method	Classification	Type	Description
BuildFromRow( )	Overridable	N/A	Override this method to manipulate the raw data to be displayed in a chart.
BuildSampleCategory ScaleData( )	Callable	N/A	Generates sample data for a chart whose *x*-axis is based on categories.
BuildSampleValue ScaleData( )	Callable	N/A	Call this method to generate sample data for a scatter chart.
ComputeMinMaxData Values( )	Callable	N/A	Computes the minimum and maximum data values for each layer of a chart and the chart as a whole from the individual data points.
ComputeScales( )	Callable	N/A	Computes the scales for all the axes of a chart.
CustomizeAxes( )	Overridable	N/A	Override this method to change the appearance of a chart's axes.
CustomizeCategories AndSeries( )	Overridable	N/A	Override this method to adjust the data displayed in a chart.
CustomizeChart( )	Overridable	N/A	Override this method to modify the initial structure of a chart.
CustomizeLayers( )	Overridable	N/A	Override this method to modify the appearance of the individual layers of a chart.
CustomizeSeries Styles( )	Overridable	N/A	Override this method to modify the appearance of individual series or pie sectors in a chart.
DescribeLayout( )	Callable	N/A	Computes the layout of a chart without rendering it.
DisableHyperchart( )	Callable	N/A	Call this method to disable hyperchart links in a chart.
DisableOverlayLayer( )	Callable	N/A	Call this method to disable the overlay layer of a chart.
DisableStudyLayers( )	Callable	N/A	Call this method to disable all study layers of a chart.
DrawOnChart( )	Callable	N/A	Call this method to add drawing elements to a chart.

Table 4-23 AcChart methods (continued)

Method	Classification	Type	Description
EnableHyperchart( )	Callable	N/A	Call this method to enable hyperchart links in a chart.
EnableOverlayLayer( )	Callable	N/A	Call this method to enable the overlay layer of a chart.
EnableStudyLayers( )	Callable	N/A	Call this method to enable a specified number of study layers in a chart.
FlipAxes( )	Callable	Boolean	Returns True if a chart's *x*-axis displays vertically.
GetBaseLayer( )	Callable	AcChart Layer	Returns a reference to the base layer of a chart.
GetBorderStyle( )	Callable	AcDrawing BorderStyle	Returns the style of the border around a chart.
GetChartDrawing Plane( )	Callable	AcDrawing ChartPlane	Returns a reference to the drawing plane of a chart.
GetFillStyle( )	Callable	AcDrawing FillStyle	Returns the background fill style for a chart.
GetHyperchartLink( )	Overridable	String	Override this method to provide the hyperlink URL for a given layer, category, and series within a chart.
GetLayer( )	Callable	AcChart Layer	Returns a reference to a layer of a chart.
GetLegendBackground Color( )	Callable	AcColor	Returns the background color of a chart's legend.
GetLegendBorder Style( )	Callable	AcDrawing BorderStyle	Returns the style of the border around a chart's legend.
GetLegendFont( )	Callable	AcFont	Returns the font used for a chart's legend.
GetLegendPlacement( )	Callable	AcChart Legend Placement	Returns the placement of a chart's legend relative to the chart.
GetNumberOfLayers( )	Callable	Integer	Returns the number of layers in a chart.
GetNumberOfStudy Layers( )	Callable	Integer	Returns the number of study layers in a chart.

(continues)

Table 4-23 AcChart methods (continued)

Method	Classification	Type	Description
GetOverlayLayer( )	Callable	AcChart Layer	Returns a reference to the overlay layer of a chart.
GetStudyLayer( )	Callable	AcChart Layer	Returns a reference to a study layer of a chart.
GetTitleStyle( )	Callable	AcDrawing TextStyle	Returns the style of a chart's title.
GetTitleText( )	Callable	String	Returns the text of a chart's title.
HasOverlayLayer( )	Callable	Boolean	Returns True if a chart has an overlay layer.
IsHyperchart( )	Callable	Boolean	Returns True if a chart has hyperchart links.
IsThreeD( )	Callable	Boolean	Returns True if a chart will be displayed with a three-dimensional appearance.
Localize( )	Overridable	N/A	Override this method to localize a chart at view time.
MakeAxes( )	Callable	N/A	Call this method to create the axes of a chart that you are creating dynamically.
MakeLayers( )	Callable	N/A	Call this method to create the layers of a chart that you are creating dynamically.
SetBackgroundColor( )	Callable	N/A	Sets the background color of a chart.
SetBorderStyle( )	Callable	AcDrawing BorderStyle	Sets the style of the border around a chart.
SetFillStyle( )	Callable	N/A	Sets the background fill style for a chart.
SetFlipAxes( )	Callable	N/A	Specifies whether to display a chart's *x*-axis vertically.
SetLegendBackground Color( )	Callable	N/A	Sets the background color of a chart's legend.
SetLegendBorder Style( )	Callable	N/A	Sets the style of the border around a chart's legend.
SetLegendFont( )	Callable	N/A	Sets the font used for a chart's legend.

Table 4-23 AcChart methods (continued)

Method	Classification	Type	Description
SetLegendPlacement( )	Callable	N/A	Sets the placement of a chart's legend.
SetMatchBaseAndOverlayScales( )	Callable	N/A	Specifies whether to force the base and overlay *y*-axis scales of a chart to be identical.
SetStatus( )	Callable	N/A	Sets the status of a chart being created dynamically.
SetThreeD( )	Callable	N/A	Specifies whether to display a chart with a three-dimensional appearance.
SetTitleStyle( )	Callable	N/A	Sets the style of a chart's title text.
SetTitleText( )	Callable	N/A	Sets a chart's title text.
StartEmpty( )	Callable	N/A	Call this method to initialize a chart being created dynamically.
StartLayers( )	Callable	N/A	Call this method to initialize the layers of a chart being created dynamically.

AcImageControl

AcImageControl is a subclass of AcControl. Use AcImageControl to display a static image or an image based on the contents of a data column. The AcImageControl method is described in Table 4-24.

Table 4-24 AcImageControl methods

Method	Classification	Type	Description
GetFileName( )	Overridable	String	Returns the file name for the image to be displayed

AcLineControl

AcLineControl is a subclass of AcControl. AcLineControl provides the logic for using a line graphic in a report. There are no public methods defined specifically for this class.

AcRectangleControl

AcRectangleControl is a subclass of AcControl. There are no public methods defined specifically for this class.

AcTextualControl

AcTextualControl is a subclass of AcControl. There are no public methods defined specifically for this class.

AcBrowserScriptingControl

A subclass of AcTextualControl, AcBrowserScriptingControl provides web functionality for reports a user can view in DHTML. AcBrowserScriptingControl methods are listed in Table 4-25.

Table 4-25 AcBrowserScriptingControl methods

Method	Classification	Type	Description
BrowserCode( )	Callable	String	Retrieves the value of the BrowserCode property
GetText( )	Callable	String	Retrieves the value of the AlternateText property

AcDataControl

AcDataControl is a subclass of AcTextualControl. AcDataControl defines the logic for setting the values of data controls, which display data obtained from the input source. Do not derive directly from AcDataControl. AcDataControl methods are listed in Table 4-26.

Table 4-26 AcDataControl methods

Method	Classification	Type	Description
Format( )	Callable	String	Returns the format pattern specified in the control's Format property
GetGroupKey( )	Callable	Variant	Returns the key for the group section that contains the control

AcCurrencyControl

AcCurrencyControl is a subclass of AcDataControl. AcCurrencyControl stores and displays a currency value. This class provides a greater level of precision than AcDoubleControl and avoids rounding errors. There are no public methods defined specifically for this class.

AcDateTimeControl

AcDateTimeControl is a subclass of AcDataControl. AcDateTimeControl stores and displays a date or time numeric value. There are no public methods defined specifically for this class.

AcDoubleControl

AcDoubleControl is a subclass of AcDataControl. AcDoubleControl stores and displays a real number, a number that has a fractional part. There are no public methods defined specifically for this class.

AcDynamicTextControl

AcDynamicTextControl is a subclass of AcDataControl. AcDynamicTextControl provides the ability to display text blocks in which the text uses multiple formatting styles. A dynamic text control also automatically adjusts its size and the size of the frame that contains it to accommodate varying amounts of data. AcDynamicTextControl methods are listed in Table 4-27.

Table 4-27 AcDynamicTextControl methods

Method	Classification	Type	Description
AutoSplitVertical( )	Callable	AcAutoSplit	Returns the value of the AutoSplitVertical property
BuildText( )	Overridable	Boolean	Parses tagged text and populates the internal data structure of the control
GetAvailableHeight( )	Overridable	AcTwips	Returns the height of the area in which text can be placed within the control
GetAvailableWidth( )	Overridable	AcTwips	Returns the width of the area in which text can be placed within the control
GetFixedWidthFont FaceName( )	Overridable	String	Returns the name of the default fixed-width font
GetPlainText( )	Overridable	String	Returns the value of the Plaintext variable
GetTaggedText( )	Overridable	String	Returns the value of the TaggedText variable
KeepTaggedText( )	Overridable	Boolean	Returns the value of the KeepTaggedText property
LineSpacing( )	Overridable	Double	Returns the value of the LineSpacing property
LineWidthPadding( )	Overridable	AcPercentage	Returns the value of the LineWidthPadding property
MinimumLineHeight( )	Overridable	AcTwips	Returns the value of the MinimumLineHeight property

(continues)

Table 4-27 AcDynamicTextControl methods (continued)

Method	Classification	Type	Description
NoSplitBottom( )	Callable	AcTwips	Returns the value of the NoSplitBottom property
NoSplitTop( )	Callable	AcTwips	Returns the value of the NoSplitTop property
ProcessText( )	Overridable	N/A	Creates the internal data structure
SetTaggedText( )	Overridable	N/A	Sets the TaggedText value
SpaceBetweenLines( )	Overridable	AcTwips	Returns the value of the SpaceBetweenLines property
SpaceBetween Paragraphs( )	Overridable	AcTwips	Returns the value of the SpaceBetweenParagraphs property
SplitMarginBottom( )	Callable	AcTwips	Returns the value of the SplitMarginBottom property
SplitMarginTop( )	Callable	AcTwips	Returns the value of the SplitMarginTop property
TabPadding( )	Overridable	AcPercentage	Returns the value of the TabPadding property
TabSpacing( )	Overridable	AcTwips	Returns the value of the TabSpacing property
TextFormat( )	Overridable	AcText Format	Returns the value of the TextFormat property
WidowAndOrphan Control( )	Overridable	Boolean	Returns the value of the WidowAndOrphanControl property

AcIntegerControl

AcIntegerControl is a subclass of AcDataControl. AcIntegerControl stores and displays whole numbers. There are no public methods defined specifically for this class.

AcTextControl

AcTextualControl is a subclass of AcDataControl. AcTextControl displays string data. Typically, this class displays one line from a table column, such as a name or address, but it also can be used for multi-line text. There are no public methods defined specifically for this class.

AcLabelControl

AcLabelControl is a subclass of AcTextualControl. AcLabelControl displays static text labels. There are no public methods defined specifically for this class.

AcPageNumberControl

AcPageNumberControl is a subclass of AcTextualControl. AcPageNumberControl calculates, formats, and displays the current page number or the total number of pages in the report. AcPageNumberControl methods are listed in Table 4-28.

Table 4-28 AcPageNumberControl methods

Method	Classification	Type	Description
GetActualPageCount( )	Callable	Integer	Returns the total page count for the report without considering page security
GetActualPage Number( )	Callable	Integer	Returns the current page number without considering page security
GetFormattedPage Number( )	Callable	String	Returns the page number without considering page security, using the format specified in the PageNumberFormat property for the page
GetVisiblePageCount( )	Callable	Integer	Returns the total page count for the report considering page security
GetVisiblePage Number( )	Callable	Integer	Returns the current page number considering page security
PageN( )	Callable	String	Formats controls that have the page number types ActualPageN or VisiblePageN
PageNOfM( )	Callable	String	Formats controls that have the page number types ActualPageNofM or VisiblePageNofM
PageNumberType( )	Callable	AcPage NumberStyle	Returns the value of the PageNumberType property

AcChartAxis

AcChartAxis class represents a single axis within a chart layer. The methods of this class change the appearance of a chart by changing a single axis. AcChartAxis does not inherit from other classes. AcChartAxis methods are listed in Table 4-29.

Table 4-29 AcChartAxis methods

Method	Classification	Type	Description
AddGridLine( )	Callable	AcChartGrid Line	Adds a grid line to the end of a chart axis's list of grid lines
ClearMajorTick Interval( )	Callable	N/A	Resets the major tick interval of a chart axis to its default setting and causes the axis to compute the major tick interval automatically
ClearMaximumValue( )	Callable	N/A	Removes a fixed maximum value from a chart axis
ClearMinimumValue( )	Callable	N/A	Removes a fixed minimum value from a chart axis
ClearOtherAxisCrosses At( )	Callable	N/A	Removes a fixed axis crossing point from a chart axis and causes the axis to compute the axis crossing point automatically
ComputeScale( )	Callable	N/A	Computes the scale for a chart axis
ForceMajorTickCount( )	Callable	Boolean	Returns True if the number of major ticks on a chart axis is forced to be a specific value
GetAxisLetter( )	Callable	AcChartAxis Letter	Returns an axis letter value that indicates the chart axis letter
GetAxisLetterText( )	Callable	String	Returns a string that indicates the chart axis letter
GetDataType( )	Callable	AcDataType	Returns the data type of the scale of a chart axis
GetDefaultRange Ratio( )	Callable	Double	Returns the ratio used to compute the range of a chart axis when all the values plotted on the axis lie on the axis's origin
GetGridLine( )	Callable	AcChartGrid Line	Returns a reference to the specified grid line within a chart axis
GetInnerMarginRatio( )	Callable	Double	Returns the minimum ratio between the inner margin on a chart axis and the total range of that axis
GetLabelFormat( )	Callable	String	Returns the format pattern used to format labels on a chart axis

Table 4-29 AcChartAxis methods (continued)

Method	Classification	Type	Description
GetLabelPlacement( )	Callable	AcChartAxis Label Placement	Returns the placement of labels on a chart axis
GetLabelStyle( )	Callable	AcDrawing TextStyle	Returns the style for labels on a chart axis
GetLabelText( )	Callable	String	Returns the formatted text of the specified label on a chart axis
GetLabelValue( )	Callable	Variant	Returns the value of the specified label on a chart axis
GetLayer( )	Callable	AcChart Layer	Returns a reference to the parent chart layer of a chart axis
GetLineStyle( )	Callable	AcDrawing LineStyle	Returns the line style used to draw a chart axis
GetMajorGridLine Style( )	Callable	AcDrawing LineStyle	Returns the line style used to draw grid lines for the major ticks on a chart axis
GetMajorTick Calculation( )	Callable	AcChartTick Calculation	Returns the type of calculation used to compute major ticks on a chart axis
GetMajorTickCount( )	Callable	Integer	Returns the exact or maximum number of major ticks on a chart axis
GetMajorTickInterval( )	Callable	Double	Returns the exact or minimum interval between major ticks on a chart axis
GetMajorTick Placement( )	Callable	AcChartTick Placement	Returns the placement of major ticks on a chart axis
GetMaximumData Value( )	Callable	Variant	Returns the highest value plotted against a chart axis
GetMaximumTrendline Value( )	Callable	Variant	Returns the maximum y value of all the trendlines in a chart axis
GetMaximumValue( )	Callable	Variant	Returns the upper bound of a chart axis
GetMinimumData Value( )	Callable	Variant	Returns the lowest value plotted against a chart axis

(continues)

Table 4-29 AcChartAxis methods (continued)

Method	Classification	Type	Description
GetMinimumTrendlineValue( )	Callable	Variant	Returns the minimum y value of all the trendlines in a chart axis
GetMinimumValue( )	Callable	Variant	Returns the lower bound of a chart axis
GetMinorGridLineStyle( )	Callable	AcDrawingLineStyle	Returns the line style used to draw grid lines for the minor ticks on a chart axis
GetMinorTickCount( )	Callable	Integer	Returns the number of minor ticks between major ticks on a chart axis
GetMinorTickPlacement( )	Callable	AcChartTickPlacement	Returns the placement of minor ticks on a chart axis
GetNoZeroRatio( )	Callable	Double	Returns the minimum ratio between the lowest and highest values plotted on a chart axis that will cause zero to be suppressed on that axis
GetNumberOfGridlines( )	Callable	Integer	Returns the number of grid lines on the chart axis
GetNumberOfLabels( )	Callable	Integer	Returns the number of labels on a chart axis
GetOriginValue( )	Callable	Variant	Returns the origin of a chart axis
GetOtherAxisCrossesAt( )	Callable	Variant	Returns the value at which the opposite axis crosses a chart axis
GetOtherAxisPlacement( )	Callable	AcChartAxisPlacement	Returns the placement of the opposite axis relative to a chart axis
GetOuterMarginRatio( )	Callable	Double	Returns the minimum ratio between the outer margin on a chart axis and the total range of that axis
GetTitleStyle( )	Callable	AcDrawingTextStyle	Returns the style of the title of a chart axis
GetTitleText( )	Callable	String	Returns the text of the title of a chart axis
HasFixedMaximum( )	Callable	Boolean	Returns True if a chart axis has a fixed upper bound
HasFixedMinimum( )	Callable	Boolean	Returns True if a chart axis has a fixed lower bound

Table 4-29 AcChartAxis methods (continued)

Method	Classification	Type	Description
IgnoreTrendlines( )	Callable	Boolean	Determines whether trendlines will be ignored when computing the scale for a chart axis
InsertGridline( )	Callable	AcChartGrid Line	Inserts a grid line at a specific position within a chart axis's list of grid lines
IsCategoryScale( )	Callable	Boolean	Returns True if a chart axis is a category scale axis
IsValueScale( )	Callable	Boolean	Returns True if a chart axis is a value scale axis
IsXAxis( )	Callable	Boolean	Returns True if a chart axis is the *x*-axis of its parent chart layer
IsYAxis( )	Callable	Boolean	Returns True if a chart axis is the *y*-axis of its parent chart layer
IsZAxis( )	Callable	Boolean	Returns True if a chart axis is the *z*-axis of its parent chart layer
PlotCategoriesBetween Ticks( )	Callable	Boolean	Returns True if categories are plotted between the ticks on a chart axis
ResetMajorTick Interval( )	Callable	N/A	Resets the major tick interval of a chart axis to its default
SetDataType( )	Callable	N/A	Sets the data type of the scale of a chart axis
SetDefaultRangeRatio( )	Callable	N/A	Sets the default ratio used to scale a chart axis when all the values plotted on the axis lie on the axis's origin
SetForceMajorTick Count( )	Callable	N/A	Specifies whether to force the number of major ticks on a chart axis to a specific value
SetIgnoreTrendlines( )	Callable	N/A	Call this method to specify whether trendlines will be ignored when computing the scale for a chart axis
SetInnerMarginRatio( )	Callable	N/A	Sets the minimum ratio between the inner margin on a chart axis and the total range of that axis

(continues)

Table 4-29 AcChartAxis methods (continued)

Method	Classification	Type	Description
SetLabelFormat( )	Callable	N/A	Sets the format pattern used to format labels on a chart axis
SetLabelPlacement( )	Callable	N/A	Specifies the placement of labels on a chart axis
SetLabelStyle( )	Callable	N/A	Sets the style for chart axis labels
SetLabelValue( )	Callable	N/A	Sets the value of the specified label on a chart axis
SetLineStyle( )	Callable	N/A	Sets the line style used to draw a chart axis
SetMajorGridLine Style( )	Callable	N/A	Sets the line style used to draw grid lines for the major ticks on a chart axis
SetMajorTick Calculation( )	Callable	N/A	Specifies the type of calculation used to compute major ticks on a chart axis
SetMajorTickCount( )	Callable	N/A	Sets the exact or maximum number of major ticks on a chart axis
SetMajorTickInterval( )	Callable	N/A	Sets the exact or minimum interval between major ticks on a chart axis
SetMajorTick Placement( )	Callable	N/A	Specifies the placement of major ticks on a chart axis
SetMaximumData Value( )	Callable	N/A	Uses a specific value as if it were the highest value plotted against a chart axis
SetMaximumValue( )	Callable	N/A	Sets a fixed upper bound on a chart axis
SetMinimumData Value( )	Callable	N/A	Uses a specific value as if it were the lowest value plotted against a chart axis
SetMinimumValue( )	Callable	N/A	Sets a fixed lower bound on a chart axis
SetMinorGridLine Style( )	Callable	N/A	Sets the line style used to draw grid lines for the minor ticks on a chart axis
SetMinorTickCount( )	Callable	N/A	Sets the number of minor ticks between major ticks on a chart axis

Table 4-29 AcChartAxis methods (continued)

Method	Classification	Type	Description
SetMinorTickPlacement( )	Callable	N/A	Specifies the placement of minor ticks on a chart axis
SetNoZeroRatio( )	Callable	N/A	Sets the minimum ratio between the lowest and highest values plotted on a chart axis that will cause zero to be suppressed on that axis
SetOtherAxisCrossesAt( )	Callable	N/A	Sets the value at which the opposite axis crosses an axis
SetOtherAxisPlacement( )	Callable	N/A	Sets the opposite axis position relative to a chart axis
SetOuterMarginRatio( )	Callable	N/A	Sets the minimum ratio between the outer margin on a chart axis and the total range of that axis
SetPlotCategoriesBetweenTicks( )	Callable	N/A	Specifies whether to plot categories between the ticks on a chart axis
SetTitleStyle( )	Callable	N/A	Sets the style of the title of a chart axis
SetTitleText( )	Callable	N/A	Sets the text of the chart axis title

AcChartCategory

Use the AcChartCategory class to represent a single category within a chart layer. Use the methods of this class to access and modify a chart layer's categories. AcChartCategory does not inherit from other classes. AcChartCategory methods are listed in Table 4-30.

Table 4-30 AcChartCategory methods

Method	Classification	Type	Description
GetIndex( )	Callable	Integer	Returns the index of a chart category within its parent chart layer's list of categories
GetKeyValue( )	Callable	Variant	Returns the unique key value for a chart category
GetLabelText( )	Callable	String	Returns the formatted label text for a chart category

(continues)

Table 4-30 AcChartCategory methods (continued)

Method	Classification	Type	Description
GetLabelValue( )	Callable	Variant	Returns the label value for a chart category
GetLayer( )	Callable	AcChart Layer	Returns a reference to the parent chart layer of a chart category
SetKeyValue( )	Callable	N/A	Sets the unique key value for a chart category
SetLabelValue( )	Callable	N/A	Sets the label value for a chart category

AcChartGridLine

Use AcChartGridLine to represent a grid line in a chart. AcChartGridLine does not inherit from other classes. AcChartGridLine methods are listed in Table 4-31.

Table 4-31 AcChartGridLine methods

Method	Classification	Type	Description
DrawInFrontOfPoints( )	Callable	Boolean	Determines whether a grid line is drawn in front of the data points in a chart.
GetAxis( )	Callable	AcChartAxis	Returns a reference to the parent chart axis of a grid line.
GetIndex( )	Callable	Integer	Returns the index of a grid line within its parent axis's list of grid lines.
GetLabelText( )	Callable	String	Returns the label text for a grid line.
GetLineStyle( )	Callable	AcDrawing LineStyle	Returns the line style used to draw a grid line.
GetValue( )	Callable	Variant	Returns the axis value at which a grid line is drawn.
SetDrawInFrontOf Points( )	Callable	N/A	True causes a grid line to be drawn in front of the data points. False causes a grid line to be drawn behind the data points.
SetLabelText( )	Callable	N/A	Sets the label text for a grid line.
SetLineStyle( )	Callable	N/A	Sets the line style for a grid line.
SetValue( )	Callable	N/A	Sets the axis value at which a grid line is drawn.

AcChartLayer

Use the AcChartLayer class to represent a single chart layer. Use AcChartLayer's methods to access a chart's layers and modify the appearance and functionality of those layers. AcChartLayer does not inherit from other classes. AcChartLayer methods are listed in Table 4-32.

Table 4-32 AcChartLayer methods

Method	Classification	Type	Description
AddCategory( )	Callable	AcChart Category	Appends a new category to the end of a chart layer's list of categories.
AddSeries( )	Callable	AcChart Series	Appends a new series to the end of a chart layer's list of series.
ChartTypeIsStackable( )	Callable	Boolean	Returns True if a chart layer's chart type supports stacked series.
GetBarShape( )	Callable	AcChartBar Shape	Returns the shape of bars in a three-dimensional bar chart layer.
GetBubbleSize( )	Callable	Double	Returns the size of the largest bubble in the chart as a percentage of the chart canvas size.
GetCategory( )	Callable	AcChart Category	Returns a reference to the specified category in a chart layer.
GetCategoryGapRatio( )	Callable	Double	Returns the size of the gap between categories in a bar chart layer, relative to the width of a single bar.
GetCategory Grouping( )	Callable	AcData Grouping	Returns a reference to the data grouping definition used to control how data are grouped into categories in a chart.
GetCategoryLabel Format( )	Callable	String	Returns the format pattern used to format category labels in a chart layer.
GetChart( )	Callable	AcChart	Returns a reference to a chart layer's parent chart.
GetChartType( )	Callable	AcChartType	Returns the chart type of a chart layer.

(continues)

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
GetDownBarBorderStyle( )	Callable	AcDrawingBorderStyle	Returns the style of the border around a down bar in a chart layer.
GetDownBarFillStyle( )	Callable	AcDrawingFillStyle	Returns the fill style for a down bar in a chart layer.
GetDropLineStyle( )	Callable	AcDrawingLineStyle	Returns the line style used to draw drop lines in a chart layer.
GetHighLowLineStyle( )	Callable	AcDrawingLineStyle	Returns the line style used to draw high-low lines in a chart layer.
GetIndex( )	Callable	Integer	Returns the index of a chart layer within its parent chart's list of layers.
GetLayerType( )	Callable	AcChartLayerType	Returns the chart layer type of a chart layer.
GetLineWidth( )	Callable	AcTwips	Returns the default width of the lines joining the points within each series in a chart layer.
GetMarkerSize( )	Callable	AcTwips	Returns the default size of markers within a chart layer.
GetMaximumDataXValue( )	Callable	Variant	Returns the maximum x value of all the points in a chart layer.
GetMaximumDataYValue( )	Callable	Variant	Returns the maximum y value of all the points in a chart layer.
GetMaximumNumberOfPoints( )	Callable	Integer	Returns the maximum number of points permitted in a chart layer.
GetMaximumNumberOfPointsPerSeries( )	Callable	Integer	Returns the maximum number of points permitted in a single series in a chart layer.
GetMaximumNumberOfSeries( )	Callable	Integer	Returns the maximum number of series permitted in a chart layer.
GetMaximumTrendlineYValue( )	Callable	Variant	Returns the maximum y value of all the trendlines in a chart layer.
GetMinimumDataXValue( )	Callable	Variant	Returns the minimum x value of all the points in a chart layer.
GetMinimumDataYValue( )	Callable	Variant	Returns the minimum y value of all the points in a chart layer.

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
GetMinimumTrendline YValue( )	Callable	Variant	Returns the minimum y value of all the trendlines in a chart layer.
GetMissingPoints( )	Callable	AcChart Missing Points	Returns the way that missing points are plotted in a chart layer.
GetNumberOf Categories( )	Callable	Integer	Returns the number of categories in a chart layer.
GetNumberOfSeries( )	Callable	Integer	Returns the number of series in a chart layer.
GetPieCenter( )	Callable	AcPoint	Returns the position of the center of a pie chart relative to the top left corner of its parent chart's chart drawing plane.
GetPieExplosion( )	Callable	AcChartPie Explode	Returns the circumstances in which pie sectors will be exploded in a pie chart layer.
GetPieExplosion Amount( )	Callable	Double	Returns the amount that pie sectors will be exploded in a pie chart layer.
GetPieExplosion TestOperator( )	Callable	AcChart Comparison Operator	Returns the operator used to test whether a pie sector will be exploded in a pie chart layer.
GetPieExplosion TestValue( )	Callable	Variant	Returns the value used to test whether a pie sector will be exploded in a pie chart layer.
GetPieRadius( )	Callable	AcTwips	Returns the radius of a pie chart. You can use this method only for two-dimensional pie charts.
GetPlotAreaBorder Style( )	Callable	AcDrawing BorderStyle	Returns the style of the border around a chart layer's plot area.
GetPlotAreaFillStyle( )	Callable	AcDrawing FillStyle	Returns the background fill style for a chart layer's plot area.

(continues)

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
GetPlotAreaPosition( )	Callable	AcPoint	Returns the position of a chart layer's plot area relative to the top left corner of its parent chart's chart drawing plane. You can use this method only for two-dimensional charts that are not pie charts.
GetPlotAreaSize( )	Callable	AcSize	Returns the size of a chart layer's plot area. You can use this method only for two-dimensional charts that are not pie charts.
GetPointBorderStyle( )	Callable	AcDrawing BorderStyle	Returns the default style for the borders around points in a chart layer.
GetPointLabelFormat( )	Callable	String	Returns the default format pattern used to format point labels in a chart layer.
GetPointLabelLine Style( )	Callable	AcDrawing LineStyle	Returns the line style used to draw point label lines in a chart layer.
GetPointLabel Placement( )	Callable	AcChart PointLabel Placement	Returns the default placement of point labels in a chart layer.
GetPointLabelSource( )	Callable	AcChart PointLabel Source	Returns the default source for point label values in a chart layer.
GetPointLabelStyle( )	Callable	AcDrawing TextStyle	Returns the default style for point labels in a chart layer.
GetSeries( )	Callable	AcChart Series	Returns a reference to the specified series in a chart layer.
GetSeriesGrouping( )	Callable	AcData Grouping	Returns a reference to the data grouping definition used to control how data are grouped into series in a chart layer.
GetSeriesLabelFormat( )	Callable	String	Returns the format pattern used to format series labels in a chart layer.
GetSeriesOverlap Ratio( )	Callable	Double	Returns the amount that adjacent series in a bar chart layer will overlap, relative to the width of a single bar.

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
GetSeriesPlacement( )	Callable	AcChart Series Placement	Returns the relative placement of points for multiple series within a category in a chart layer.
GetSeriesStyle( )	Callable	AcChart SeriesStyle	Returns a reference to the specified series style in a chart layer.
GetStartAngle( )	Callable	AcAngle	Returns the angle at which the first sector in a pie chart layer is drawn.
GetStudyHeightRatio( )	Callable	Double	Returns the ratio of the height of a study layer to the height of its parent chart's base layer.
GetThreeDBackWall FillStyle( )	Callable	AcDrawing FillStyle	Returns the background fill style for a three-dimensional chart's back wall.
GetThreeDFloorFill Style( )	Callable	AcDrawing FillStyle	Returns the background fill style for a three-dimensional chart's floor.
GetThreeDSideWall FillStyle( )	Callable	AcDrawing FillStyle	Returns the background fill style for a three-dimensional chart's side wall.
GetUpBarBorderStyle( )	Callable	AcDrawing BorderStyle	Returns the style of the border around an up bar in a chart layer.
GetUpBarFillStyle( )	Callable	AcDrawing FillStyle	Returns the fill style for a down bar in a chart layer.
GetXAxis( )	Callable	AcChartAxis	Returns a reference to a chart layer's *x*-axis.
GetYAxis( )	Callable	AcChartAxis	Returns a reference to a chart layer's *y*-axis.
HasCategoryScale XAxis( )	Callable	Boolean	Returns True if a chart layer's *x*-axis is a category scale axis.
HasValueScaleXAxis( )	Callable	Boolean	Returns True if a chart layer's *x*-axis is a value scale axis.
HasXAxis( )	Callable	Boolean	Returns True if a chart layer has an *x*-axis.
HasYAxis( )	Callable	Boolean	Returns True if a chart layer has a *y*-axis.

(continues)

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
InsertCategory( )	Callable	AcChart Category	Inserts a new category at a specific position in a chart layer's list of categories.
InsertSeries( )	Callable	AcChart Series	Inserts a new series at a specific position in a chart layer's list of series.
IsBaseLayer( )	Callable	Boolean	Returns True if a chart layer is the base layer of its parent chart.
IsOverlayLayer( )	Callable	Boolean	Returns True if a chart layer is the overlay layer of its parent chart.
IsStacked( )	Callable	Boolean	Returns True if the series in a chart layer are stacked.
IsStudyLayer( )	Callable	Boolean	Returns True if a chart layer is a study layer of its parent chart.
PieExplosionTestValue IsPercentage( )	Callable	Boolean	Returns True if the pie explosion test value in a pie chart layer is treated as a percentage of the total pie.
PlotBarsAsLines( )	Callable	Boolean	Returns True if points in a bar chart layer will be plotted as lines instead of bars.
PlotLinesBetween Points( )	Callable	Boolean	Returns True if the default setting for series in a chart layer is that lines will be drawn between the points within each series.
PlotMarkersAtPoints( )	Callable	Boolean	Returns True if the default setting for series within a chart layer is that markers will be drawn at points.
PlotUpDownBars( )	Callable	Boolean	Returns True if up and down bars will be drawn between points within each category in a chart layer.
RemoveCategory( )	Callable	N/A	Removes a category from a chart layer.
RemoveSeries( )	Callable	N/A	Removes a series from a chart layer.

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
SetBarShape( )	Callable	N/A	Sets the shape of bars in a three-dimensional bar chart layer.
SetBubbleSize( )	Callable	N/A	Sets the size of the largest bubble in a bubble chart as a percentage of the chart canvas size. Must be in the range of MinimumBubbleSize to MaximumBubbleSize.
SetCategoryGapRatio( )	Callable	N/A	Sets the size of the gap between categories in a bar chart layer, relative to the width of a single bar.
SetCategoryLabelFormat( )	Callable	N/A	Sets the format pattern used to format category labels in a chart layer.
SetChartType( )	Callable	N/A	Sets the chart type of a chart layer.
SetDownBarBorderStyle( )	Callable	N/A	Sets the style of the border around down bars in a chart layer.
SetDownBarFillStyle( )	Callable	N/A	Sets the fill style for down bars in a chart layer.
SetDropLineStyle( )	Callable	N/A	Sets the line style used to draw drop lines in a chart layer.
SetHighLowLineStyle( )	Callable	N/A	Sets the line style used to draw high-low lines in a chart layer.
SetLineWidth( )	Callable	N/A	Sets the default width of the lines joining the points within each series in a chart layer.
SetMarkerSize( )	Callable	N/A	Sets the default size for markers within a chart layer.
SetMaximumNumberOfPoints( )	Callable	N/A	Sets the maximum number of points permitted in a chart layer.
SetMaximumNumberOfPointsPerSeries( )	Callable	N/A	Sets the maximum number of points permitted in a single series in a chart layer.
SetMaximumNumberOfSeries( )	Callable	N/A	Sets the maximum number of series permitted in a chart layer.
SetMissingPoints( )	Callable	N/A	Specifies how to plot missing points in a chart layer.

(continues)

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
SetPieExplosion( )	Callable	N/A	Specifies the circumstances in which pie sectors are exploded in a pie chart layer.
SetPieExplosion Amount( )	Callable	N/A	Sets the amount by which pie sectors are exploded in a pie chart layer.
SetPieExplosion TestOperator( )	Callable	N/A	Sets the operator used to test whether a pie sector will be exploded in a pie chart layer.
SetPieExplosion TestValue( )	Callable	N/A	Sets the value used to test whether a pie sector will be exploded in a pie chart layer.
SetPieExplosionTest ValueIsPercentage( )	Callable	N/A	Specifies whether the pie explosion test value in a pie chart layer is treated as a percentage of the total pie.
SetPlotAreaBackground Color( )	Callable	N/A	Sets the background color of a chart layer's plot area.
SetPlotAreaBorder Style( )	Callable	N/A	Sets the style of the border around a chart layer's plot area.
SetPlotAreaFillStyle( )	Callable	N/A	Sets the background fill style for a chart layer's plot area.
SetPlotBarsAsLines( )	Callable	N/A	Specifies whether to plot points in a bar chart layer as lines instead of bars.
SetPlotHighLowLines( )	Callable	N/A	Specifies whether to draw high-low lines in a chart layer.
SetPlotLinesBetween Points( )	Callable	N/A	Specifies whether the default setting for series in a chart layer is to draw lines between the points within each series.
SetPlotMarkersAt Points( )	Callable	N/A	Specifies whether the default setting for series within a chart layer is to draw markers at points.
SetPlotUpDownBars( )	Callable	N/A	Specifies whether to draw up-and-down bars between points within each category in a chart layer.

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
SetPointBorderStyle( )	Callable	N/A	Sets the default style for the borders around points in a chart layer.
SetPointLabelFormat( )	Callable	N/A	Sets the default format pattern used to format point labels in a chart layer.
SetPointLabelLine Style( )	Callable	N/A	Sets the line style used to draw point label lines in a chart layer.
SetPointLabel Placement( )	Callable	N/A	Sets the default placement of point labels in a chart layer.
SetPointLabelSource( )	Callable	N/A	Sets the default source for point label values in a chart layer.
SetPointLabelStyle( )	Callable	N/A	Sets the default style for point labels in a chart layer.
SetSeriesLabelFormat( )	Callable	N/A	Sets the default format pattern used to format series labels in a chart layer.
SetSeriesOverlap Ratio( )	Callable	N/A	Specifies the amount by which adjacent series in a bar chart layer overlap, relative to the width of a single bar.
SetSeriesPlacement( )	Callable	N/A	Sets the relative placement of points for multiple series within a category in a chart layer.
SetStartAngle( )	Callable	N/A	Sets the angle at which to draw the first sector in a pie chart layer.
SetStockHasClose( )	Callable	N/A	Specifies whether a stock chart layer has a Close series.
SetStockHasOpen( )	Callable	N/A	Specifies whether a stock chart layer has an Open series.
SetStudyHeightRatio( )	Callable	N/A	Sets the ratio of the height of a study layer to the height of its parent chart's base layer.
SetThreeDFloorFill Style( )	Callable	N/A	Sets the background fill style for a three-dimensional chart's floor.
SetThreeDWallFill Style( )	Callable	N/A	Sets the background fill style for a three-dimensional chart's walls.

(continues)

Table 4-32 AcChartLayer methods (continued)

Method	Classification	Type	Description
SetUpBarBorderStyle( )	Callable	N/A	Sets the style of the border around down bars in a chart layer.
SetUpBarFillStyle( )	Callable	N/A	Sets the fill style for down bars in a chart layer.
StockHasClose( )	Callable	Boolean	Returns True if a stock chart layer has a Close series.
StockHasOpen( )	Callable	Boolean	Returns True if a stock chart layer has an Open series.

AcChartPoint

AcChartPoint represents a single point within a chart series. Use the methods of AcChartPoint to access and modify the appearance of a chart's points. AcChartPoint does not inherit from other classes. AcChartPoint methods are listed in Table 4-33.

Table 4-33 AcChartPoint methods

Method	Classification	Type	Description
AddCustomStyle( )	Callable	AcChart PointStyle	Adds a custom style to a chart point.
ClearCustomLabel Format( )	Callable	N/A	Removes a custom label format pattern from a chart point.
ClearCustomLabel Value( )	Callable	N/A	Removes a custom label value from a chart point.
ClearValues( )	Callable	N/A	Makes a chart point into an empty point.
ExplodeSlice( )	Callable	Boolean	Returns True if a chart point is a pie chart sector that is exploded.
GetCategory( )	Callable	AcChart Category	Returns a reference to the chart category corresponding to a chart point.
GetCustomLabel Format( )	Callable	String	Returns the custom format pattern that formats a chart point's label.
GetCustomLabel Value( )	Callable	Variant	Returns the custom value of a chart point's label.
GetCustomStyle( )	Callable	AcChart PointStyle	Returns a reference to the custom style for a chart point.

Table 4-33 AcChartPoint methods

Method	Classification	Type	Description
GetIndex( )	Callable	Integer	Returns the index of a chart point within its parent chart series' list of points.
GetLabelText( )	Callable	String	Returns the formatted text of a chart point's label.
GetSeries( )	Callable	AcChart Series	Returns a reference to the parent chart series of a chart point.
GetXValue( )	Callable	Variant	Returns the x value of a chart point.
GetYValue( )	Callable	Variant	Returns the y value of a chart point.
GetZValue( )	Callable	Variant	Returns the z value of a chart point. Currently, only bubble charts support z values.
HasCustomLabel Format( )	Callable	Boolean	Returns True if a chart point has a custom label format pattern.
HasCustomLabel Value( )	Callable	Boolean	Returns True if a chart point has a custom label value.
HasCustomStyle( )	Callable	Boolean	Returns True if a chart point has a custom style.
IsMissing( )	Callable	Boolean	Returns True if a chart point is empty.
SetCustomLabel Format( )	Callable	N/A	Adds a custom label format pattern to a chart point.
SetCustomLabelValue( )	Callable	N/A	Adds a custom label value to a chart point.
SetExplodeSlice( )	Callable	N/A	Specifies whether a chart point that is a pie sector is exploded.
SetValues( )	Callable	N/A	Sets the values of a chart point.
SetXValue( )	Callable	N/A	Sets the x value of a chart point.
SetYValue( )	Callable	N/A	Sets the y value of a chart point.
SetZValue( )	Callable	Variant	Sets the z value of a chart point. Currently, only bubble charts support *z* values.

AcChartPointStyle

AcChartPointStyle represents a custom style for a single point within a chart series. Use AcChartPointStyle's methods to create, access, and modify a chart point's custom style. AcChartPointStyle does not inherit from other classes. AcChartPointStyle methods are listed in Table 4-34.

Table 4-34 AcChartPointStyle methods

Method	Classification	Type	Description
GetBorderStyle( )	Callable	AcDrawing BorderStyle	Returns the style of the border around a chart point
GetFillStyle( )	Callable	AcDrawing FillStyle	Returns the background fill style for a chart point
GetMarkerFillColor( )	Callable	AcColor	Returns the fill color of the marker for a chart point
GetMarkerLineColor( )	Callable	AcColor	Returns the line color of the marker for a chart point
GetMarkerShape( )	Callable	AcChart MarkerShape	Returns the shape of the marker for a chart point
GetMarkerSize( )	Callable	AcTwips	Returns the size of the marker for a chart point
GetPieExplosion Amount( )	Callable	Double	Returns the amount that a pie sector chart point is exploded in a pie chart layer
GetPointLabel Placement( )	Callable	AcChart PointLabel Placement	Returns the placement of the point label for a chart point
GetPointLabelStyle( )	Callable	AcDrawing TextStyle	Returns the style of the point label for a chart point
SetBackgroundColor( )	Callable	N/A	Sets the background color for a chart point
SetBorderStyle( )	Callable	N/A	Sets the style of the border around a chart point
SetFillStyle( )	Callable	N/A	Sets the background fill style for a chart point
SetMarkerFillColor( )	Callable	N/A	Sets the fill color of the marker for a chart point
SetMarkerLineColor( )	Callable	N/A	Sets the line color of the marker for a chart point

Table 4-34 AcChartPointStyle methods

Method	Classification	Type	Description
SetMarkerShape( )	Callable	N/A	Sets the shape of the marker for a chart point
SetMarkerSize( )	Callable	N/A	Sets the size of the marker for a chart point.
SetPieExplosionAmount( )	Callable	N/A	Sets the amount that a pie sector chart point is exploded in a pie chart layer
SetPointLabelPlacement( )	Callable	N/A	Sets the placement of the point label for a chart point
SetPointLabelStyle( )	Callable	N/A	Sets the style of the point label for a chart point

AcChartSeriesStyle

A subclass of AcChartPointStyle, AcChartSeriesStyle represents a custom style for a chart series. AcChartSeriesStyle methods are listed in Table 4-35.

Table 4-35 AcChartSeriesStyle methods

Method	Classification	Type	Description
GetLineStyle( )	Callable	AcDrawingLineStyle	Returns the style of lines between points in a chart series
GetPointLabelFormat( )	Callable	String	Returns the format pattern used to format point labels in a chart series or a pie chart category
GetPointLabelSource( )	Callable	AcChartPointLabelSource	Returns the source for point label values in a chart series or a pie chart category
PlotBarsAsLines( )	Callable	Boolean	Returns True if points are plotted as lines in a chart series
PlotLinesBetweenPoints( )	Callable	Boolean	Returns True if lines are plotted between points in a chart series
PlotMarkersAtPoints( )	Callable	Boolean	Returns True if markers are drawn by default at points in a chart series
SetLineStyle( )	Callable	AcDrawingLineStyle	Sets the style of lines between points in a chart series

(continues)

Table 4-35 AcChartSeriesStyle methods (continued)

Method	Classification	Type	Description
SetPlotBarsAsLines( )	Callable	N/A	Determines whether to plot points as lines in a bar chart series
SetPlotLinesBetween Points( )	Callable	N/A	Determines whether to plot lines between points in a chart series
SetPlotMarkersAt Points( )	Callable	N/A	Determines whether to draw markers by default at points in a chart series
SetPointLabelFormat( )	Callable	N/A	Sets the format pattern used to format point labels in a chart series or a pie chart category
SetPointLabelSource( )	Callable	N/A	Sets the source for point label values in a chart series or a pie chart category

AcChartSeries

AcChartSeries represents a single series within a chart layer. AcChartSeries methods support accessing and modifying a chart layer's series. AcChartSeries does not inherit from other classes. AcChartSeries methods are listed in Table 4-36.

Table 4-36 AcChartSeries methods

Method	Classification	Type	Description
AddEmptyPoint( )	Callable	AcChart Point	Appends a new empty point to the end of a chart series' list of points
AddPoint( )	Callable	AcChart Point	Appends a new point to the end of a chart series' list of points
AddTrendline( )	Callable	AcChart Trendline	Adds a trendline to the end of a chart series' list of trendlines
GetIndex( )	Callable	Integer	Returns the index of a chart series within its parent chart layer's list of series
GetKeyValue( )	Callable	Variant	Returns the unique key value for a chart series
GetLabelText( )	Callable	String	Returns the formatted label text for a chart series
GetLabelValue( )	Callable	Variant	Returns the label value for a chart series

Table 4-36 AcChartSeries methods

Method	Classification	Type	Description
GetLayer( )	Callable	AcChart Layer	Returns a reference to the parent layer of a chart series
GetNumberOfPoints( )	Callable	Integer	Returns the number of points in a chart series
GetNumberOf Trendlines( )	Callable	Integer	Returns the number of trendlines in a chart series
GetPoint( )	Callable	AcChart Point	Returns a reference to a point in a chart series
GetStyle( )	Callable	AcChart SeriesStyle	Returns a reference to the series style corresponding to a chart series
GetSumOfSliceValues( )	Callable	Variant	Returns the sum of the values of all the sectors in a pie chart series
GetTrendline( )	Callable	AcChart Trendline	Returns a reference to a trendline in a chart
InsertEmptyPoint( )	Callable	AcChart Point	Inserts a new empty point at a specific position in a chart series' list of points
InsertPoint( )	Callable	AcChart Point	Inserts a new point at a specific position in a chart series' list of points
InsertTrendline( )	Callable	AcChart Point	Returns a reference to the specified trendline for a chart series
RemovePoint( )	Callable	N/A	Removes a point from a chart series
RemoveTrendline( )	Callable	N/A	Removes a trendline at a specific position within a chart series' list of trendlines
SetKeyValue( )	Callable	N/A	Sets the unique key value for a chart series.
SetLabelValue( )	Callable	N/A	Sets the chart series' label value

AcChartTrendline

Use AcChartTrendline to represent a trendline in a chart. AcChartTrendline does not inherit from other classes. AcChartTrendline methods are listed in Table 4-37.

Table 4-37 AcChartTrendline methods

Method	Classification	Type	Description
ClearIntercept( )	Callable	N/A	Clears the intercept value for a trendline
GetEndYValue( )	Callable	Variant	Returns the y value of the end of a trendline
GetIndex( )	Callable	Integer	Returns the index of a trendline within its parent chart series' list of trendlines
GetIntercept( )	Callable	Variant	Returns the intercept value for a trendline
GetLabelText( )	Callable	String	Returns the trendline's label text
GetLineStyle( )	Callable	AcDrawing LineStyle	Returns the line style used to draw a trendline
GetMaximumYValue( )	Callable	Variant	Returns the maximum y value of a trendline
GetMinimumYValue( )	Callable	Variant	Returns the minimum y value of a trendline
GetOrder( )	Callable	Integer	Returns the order of a polynomial trendline
GetPeriod( )	Callable	Integer	Returns the period of a moving average trendline
GetStartYValue( )	Callable	Variant	Returns the y value of the start of a trendline
GetTrendlineType( )	Callable	AcChart Trendline Type	Returns a value that indicates how a trendline is fitted to the points in its parent series
HasIntercept( )	Callable	Boolean	Determines whether a trendline has an intercept value
SetIntercept( )	Callable	N/A	Sets the intercept value for a trendline
SetLabelText( )	Callable	N/A	Sets the label text for a trendline
SetLineStyle( )	Callable	N/A	Sets the line style used to draw a trendline
SetOrder( )	Callable	N/A	Sets the order of a polynomial trendline
SetPeriod( )	Callable	N/A	Sets the period for a moving average trendline

Table 4-37 AcChartTrendline methods

Method	Classification	Type	Description
SetTrendlineType( )	Callable	N/A	Defines how a trendline is fitted to the points in its parent series

AcDrawingPlane

AcDrawingPlane represents a single drawing plane within a drawing. AcDrawingPlane does not inherit from other classes. AcDrawingPlane methods are listed in Table 4-38.

Table 4-38 AcDrawingPlane methods

Method	Classification	Type	Description
GetDrawingPlaneType( )	Callable	AcDrawingPlaneType	Returns the type of a drawing plane
IsHidden( )	Callable	Boolean	Determines whether a drawing plane is hidden
SetHidden( )	Callable	N/A	Specifies whether a drawing plane is hidden
SetPosition( )	Callable	N/A	Sets the position of a drawing plane within its parent drawing
SetSize( )	Callable	N/A	Sets the size of a drawing plane

AcDrawingChartPlane

AcDrawingChartPlane is a subclass of AcDrawingPlane. Use this class to represent a drawing plane for a chart within a drawing. There are no public methods defined specifically for AcDrawingChartPlane.

AcDrawingSVGPlane

AcDrawingSVGPlane is a subclass of AcDrawingPlane. Use this class to represent a drawing plane whose contents are defined using Scalable Vector Graphics (SVG). AcDrawingSVGPlane methods are listed in Table 4-39.

Table 4-39 AcDrawingSVGPlane methods

Method	Classification	Type	Description
GetSVG( )	Callable	String	Returns the SVG code for an SVG drawing plane
SetSVG( )	Callable	N/A	Sets the SVG code for an SVG drawing plane

Connection classes and methods

Use connection classes and methods to connect to a data source.

AcConnection

AcConnection is a subclass of AcComponent. AcConnection defines the protocol for connecting to and disconnecting from an input source. AcConnection methods are listed in Table 4-40.

Table 4-40 AcConnection methods

Method	Classification	Type	Description
Connect( )	Overridable	Boolean	Sets run-time properties and establishes a connection. An empty method that derived classes override to connect to a data source.
Disconnect( )	Callable	N/A	Disconnects from a data source. An empty method that derived classes override.
IsConnected( )	Callable	Boolean	Determines whether a data source connection exists.
RaiseError( )	Callable	N/A	Produces an error message.

AcDBConnection

AcDBConnection is a subclass of AcConnection. AcDBConnection provides methods for connecting to and disconnecting from a database and defining error-handling methods when a connection fails. This class also provides the logic for creating the database statement object required to execute a SQL statement. AcDBConnection methods are listed in Table 4-41.

Table 4-41 AcDBConnection methods

Method	Classification	Type	Description
GetGeneralError( )	Callable	Integer	Returns the general error code
GetGeneralErrorText( )	Callable	String	Returns the text of the general error
GetSpecificError( )	Callable	Integer	Returns the specific error code
GetSpecificErrorText( )	Callable	String	Returns the text of the specific error
Prepare( )	Callable	AcDB Statement	Creates a database statement object for a SQL statement

AcDB2Connection

AcDB2Connection is a subclass of AcDBConnection. AcDB2Connection establishes a connection to a DB2 database. There are no public methods defined specifically for this class.

AcMSSQLConnection

AcMSSQLConnection is a subclass of AcDBConnection. AcMSSQLConnection establishes a connection to a Microsoft SQL database. There are no public methods defined specifically for this class.

AcOdaConnection

AcOdaConnection is a subclass of AcDBConnection. AcOdaConnection establishes a connection to an Open Data Access (ODA) driver. AcOdaConnection methods are listed in Table 4-42.

Table 4-42 AcOdaConnection methods

Method	Classification	Type	Description
SetProperties( )	Overridable	N/A	Sets the value of a property variable to the value the user sets
SetRuntimeProperties( )	Overridable	N/A	Calls SetConnectionProperty( ) to assign a value to each run-time property of the connection

AcODBCConnection

AcODBCConnection is a subclass of AcDBConnection. AcODBCConnection establishes a connection to an ODBC database. There are no public methods defined specifically for this class.

AcOracleConnection

AcOracleConnection is a subclass of AcDBConnection. AcOracleConnection establishes a connection to an Oracle database. There are no public methods defined specifically for this class.

AcDBCursor

AcDBCursor provides an Actuate Basic interface to a database cursor for a SQL statement. A database cursor is an identifier associated with a set of rows. SELECT statements that return more than one row of data require a database cursor. Use this class to create a cursor when you write custom code to handle data retrieval from, for example, a stored procedure. AcDBCursor does not inherit from other classes. AcDBCursor methods are listed in Table 4-43.

Table 4-43 AcDBCursor methods

Method	Classification	Type	Description
BindColumn( )	Callable	N/A	Binds a database column to a data row variable
BindParameter( )	Callable	Boolean	Assigns the specified value to a cursor parameter
CloseCursor( )	Callable	N/A	Closes the cursor
DefineProcedureInput Parameter( )	Callable	Boolean	Defines an input parameter used by a stored procedure
DefineProcedureOutput Parameter( )	Callable	Boolean	Defines an input and output parameter or an output only parameter used by a stored procedure
DefineProcedureReturn Parameter( )	Callable	Boolean	Specifies the data type of a return value from a stored procedure
Delete( )	Overridable	N/A	Deletes the cursor object
Fetch( )	Callable	Boolean	Reads one row from the cursor
GetConnection( )	Callable	AcDB Statement	Returns the connection against which the cursor operates
GetOutputParameter( )	Callable	Variant	Returns the value of a stored procedure’s output parameter
GetProcedureStatus( )	Callable	Integer	Returns a value that indicates the status of a stored procedure
GetStatement( )	Callable	AcDB Statement	Returns the statement from which the cursor was created
IsOpen( )	Callable	Boolean	Determines whether the cursor is open
New( )	Overridable	N/A	The class’ constructor method
OpenCursor( )	Callable	Boolean	Opens the cursor
SetProperty( )	Callable	Boolean	Sets a parameter property for a stored procedure
StartNextSet( )	Callable	Boolean	Starts a new set of rows within a stored procedure

AcDBStatement

AcDBStatement provides an Actuate Basic interface to a SQL statement. AcDBStatement does not inherit from other classes. AcDBStatement methods are listed in Table 4-44.

Table 4-44 AcDBStatement methods

Method	Classification	Type	Description
AllocateCursor( )	Callable	AcDBCursor	Creates a cursor to read the rows the statement returns
BindParameter( )	Callable	Boolean	Binds a statement parameter to a variable
DefineProcedureInput Parameter( )	Callable	Boolean	Defines an input parameter used by a stored procedure
DefineProcedureOutput Parameter( )	Callable	Boolean	Defines an input and output parameter or an output-only parameter for a stored procedure
DefineProcedureReturn Parameter( )	Callable	Boolean	Specifies the data type of a return value from a stored procedure
Delete( )	Callable	N/A	The destructor method
Execute( )	Callable	Boolean	Executes the SQL statement
GetOutputCount( )	Callable	Integer	Returns the number of columns in the rows that the SQL statement returns
GetOutputParameter( )	Callable	Variant	Returns an output parameter of a stored procedure by name or position
GetParameterCount( )	Callable	Integer	Returns the number of parameters in the SQL statement
GetProcedureStatus( )	Callable	Integer	Returns the return value from a stored procedure
GetStatementText( )	Callable	String	Returns the text of the SQL statement previously passed to Prepare( )
OpenCursor( )	Callable	AcDBCursor	Creates and opens a cursor to use for reading the rows that the statement returns
Prepare( )	Callable	Boolean	Prepares a SQL statement

Collection classes and methods

Use collection classes and methods to work with arrays.

AcCollection

AcCollection is the base class for collection classes. This class provides methods common to all collections. AcCollection does not inherit from other classes. AcCollection methods are listed in Table 4-45.

Table 4-45 AcCollection methods

Method	Classification	Type	Description
Compare( )	Callable	Variant	Compares two objects in a collection
Contains( )	Callable	Boolean	Determines whether an object exists in the collection
Copy( )	Callable	N/A	Copies the contents of another collection into the current collection
FindByValue( )	Callable	AnyClass	Finds an object that has the same value as the current object
GetCount( )	Callable	Integer	Returns the number of objects in the collection
IsEmpty( )	Callable	Boolean	Determines whether the collection is empty
NewIterator( )	Overridable	AcIterator	Creates an iterator for the collection
Remove( )	Callable	N/A	Removes a specified item from the collection
RemoveAll( )	Callable	N/A	Removes all contents from the collection

AcBTree

AcBTree is a subclass of AcCollection. Use AcBTree to create a balanced-tree list of objects sorted by one of the object's properties. Table 4-46 lists AcBTree methods.

Table 4-46 AcBTree methods

Method	Classification	Type	Description
Abandon( )	Overridable	N/A	Removes an object that the balanced tree no longer needs and recovers memory

Table 4-46 AcBTree methods

Method	Classification	Type	Description
CompareKey( )	Overridable	Integer	Compares the values of two keys
CreateNode( )	Overridable	N/A	Adds a new node
Find( )	Callable	AnyClass	Finds the object with the given key
FindOrCreate( )	Callable	AnyClass	Locates an object that has a specified key or creates the object if an object with the specified key does not exist in the collection
GetKey( )	Overridable	Variant	Returns the key for an object
Insert( )	Callable	AnyClass	Adds an object to the collection
New( )	Callable	N/A	Constructor method for this class

AcOrderedCollection

AcOrderedCollection is a subclass of AcCollection. AcOrderedCollection creates a collection in which you control the order of the objects. Use AcOrderedCollection methods to add and remove objects from the front or back of a collection, to insert objects within a collection, and to copy the contents of one collection into another collection. Table 4-47 lists AcOrderedCollection methods.

Table 4-47 AcOrderedCollection methods

Method	Classification	Type	Description
AddToHead( )	Callable	N/A	Adds an object to the beginning of the collection
AddToTail( )	Callable	N/A	Adds an object to the end of the collection
GetAt( )	Callable	AnyClass	Returns the object at a specified location in the collection
GetHead( )	Callable	AnyClass	Returns the first object in the collection
GetIndex( )	Callable	Integer	Returns the position of an object in the collection
GetTail( )	Callable	AnyClass	Returns the last object in the collection
InsertAfter( )	Callable	N/A	Inserts an object after a specified object in the collection

(continues)

Table 4-47 AcOrderedCollection methods (continued)

Method	Classification	Type	Description
InsertAt( )	Callable	N/A	Inserts a new object at a specific location, moving the object currently at that location and all objects above it one higher
InsertBefore( )	Callable	N/A	Inserts an object before a specified object in the collection
RemoveHead( )	Callable	AnyClass	Removes the first object in the collection
RemoveTail( )	Callable	AnyClass	Removes the last object in the collection
SetAt( )	Callable	N/A	Replaces an object at a specified position with the specified object

AcList

AcList is a subclass of AcOrderedCollection. AcList is an abstract class that defines the list interface. There are no public methods defined specifically for this class.

AcSingleList

AcSingleList is a subclass of AcList. AcSingleList processes a singly-linked list. There are no public methods defined specifically for this class.

AcObjectArray

AcObjectArray is a subclass of AcOrderedCollection. AcObjectArray creates a resizable array of objects. AcObjectArray methods are listed in Table 4-48.

Table 4-48 AcObjectArray methods

Method	Classification	Type	Description
RemoveAt( )	Callable	AnyClass	Removes the object at a specific location in the array.
RemoveEmptyEntries( )	Callable	N/A	Removes slots that contain Nothing. Resets the count by the number of slots removed.
ResizeBy( )	Callable	N/A	Resizes the array by a specific number of slots.
ResizeTo( )	Callable	N/A	Resets the size of the array to a specific number of slots.

Table 4-48 AcObjectArray methods

Method	Classification	Type	Description
SetGrowthIncrement( )	Callable	N/A	Sets the number of slots to add each time the array expands.

AcStaticIndex

A subclass of AcOrderedCollection, AcStaticIndex implements a multi-layer tree to provide fast indexing into a large collection of data. A static index pre-allocates space rather than building the index dynamically. AcStaticIndex methods are listed in Table 4-49.

Table 4-49 AcStaticIndex methods

Method	Classification	Type	Description
AddLevel( )	Callable	N/A	Adds a level if necessary when building a static index of a particular size
New( )	Callable	N/A	Creates a new static index

AcIterator

AcIterator is the base class for all iterators. This class provides the methods needed to iterate through a list. AcIterator does not inherit from other classes. AcIterator methods are listed in Table 4-50.

Table 4-50 AcIterator methods

Method	Classification	Type	Description
Copy( )	Callable	AcIterator	Copies this iterator. The copy has the same state as this iterator.
GetItem( )	Callable	AnyClass	Returns the current item in the list.
GetNext( )	Callable	AnyClass	Returns the next item in the list.
GetPosition( )	Callable	Integer	Returns the current position of the iterator.
HasMore( )	Callable	Boolean	Determines whether there are more items in the list.
IsDone( )	Callable	Boolean	Determines whether there are no more items in the list.
MoveNext( )	Callable	N/A	Moves the iterator to the next position in the list.

(continues)

Table 4-50 AcIterator methods (continued)

Method	Classification	Type	Description
Restart( )	Callable	N/A	Positions the iterator using an index.
SkipForwardTo( )	Callable	N/A	Moves the iterator to a specific location ahead of the current location in the list. Searches only from the current position to the end of the list.
SkipTo( )	Callable	N/A	Positions the iterator to a new location in the list. If the location is before the current position, the iterator rewinds.
SkipToItem( )	Callable	Boolean	Skips to a specific object. Searches the entire index for the object. If the method does not find the object, returns False and does not change the position of the iterator.

Data stream classes and methods

Use data stream classes and methods to work with data rows, data sources, and filters.

AcDataAdapter

AcDataAdapter is a subclass of AcComponent. AcDataAdapter is an abstract base class that defines the logic of data-related classes, such as data sources and data filters, that can combine to form a data stream. The data stream manages data collection and processing tasks. The parts of a data stream are called data adapters. AcDataAdapter methods are listed in Table 4-51.

Table 4-51 AcDataAdapter methods

Method	Classification	Type	Description
AddRow( )	Callable	N/A	Adds a row to the data adapter
AddSortKey( )	Callable	N/A	Adds a dynamic sort key column
CanSeek( )	Overridable	Boolean	Returns True if the data adapter supports random access
CanSortDynamically( )	Overridable	Boolean	Determines whether the data adapter supports dynamic ordering
CloseConnection( )	Overridable	N/A	Closes the connection

Table 4-51 AcDataAdapter methods

Method	Classification	Type	Description
Fetch( )	Overridable	AcDataRow	Reads the row at the position identified by GetPosition( )
Finish( )	Overridable	N/A	Closes a data adapter
FlushBuffer( )	Callable	N/A	Flushes all buffered rows
FlushBufferTo( )	Overridable	N/A	Flushes all buffered rows up to the row specified
GetConnection( )	Callable	Ac Connection	Returns the connection associated with the data adapter
GetPosition( )	Callable	Integer	Returns the position of the next row that will be fetched
IsStarted( )	Callable	Boolean	Returns True if the adapter is open
NewConnection( )	Overridable	Ac Connection	Instantiates the connection specified in the data adapter's Connection slot
NewDataRow( )	Overridable	AcDataRow	Instantiates a data row based on the DataRow property
OpenConnection( )	Overridable	Boolean	Opens a connection
Rewind( )	Callable	N/A	Moves the fetch position to the beginning of the input set
SeekBy( )	Callable	N/A	Moves the fetch position by a given amount relative to the current position
SeekTo( )	Overridable	N/A	Moves the fetch position to a given location
SeekToEnd( )	Callable	N/A	Moves the fetch position to one past the end of the input set
SetConnection( )	Callable	N/A	Provides a connection to use if the data adapter does not have its own connection
Start( )	Overridable	Boolean	Opens the data adapter

AcDataFilter

AcDataFilter is a subclass of AcDataAdapter. AcDataFilter is the base class for the two general types of data filter classes, AcMultipleInputFilter and AcSingleInputFilter. AcDataFilter defines the logic for processing data rows retrieved from another data adapter. There are no public methods defined specifically for this class.

AcMultipleInputFilter

A subclass of AcDataFilter, AcMultipleInputFilter is a base class for data filters. AcMultipleInputFilter accepts input from multiple data adapters, processes the data, and passes the data to the next data adapter or to the report. AcMultipleInputFilter methods are listed in Table 4-52.

Table 4-52 AcMultipleInputFilter methods

Method	Classification	Type	Description
GetInputCount( )	Callable	Integer	Counts the number of data adapters that provide input
NewInputAdapter( )	Overridable	AcData Adapter	Instantiates the input adapters specified in the Input slot

AcSingleInputFilter

A subclass of AcDataFilter, AcSingleInputFilter is a data filter that accepts one data adapter as its input and filters each data row. AcSingleInputFilter methods are listed in Table 4-53.

Table 4-53 AcSingleInputFilter methods

Method	Classification	Type	Description
GetInput( )	Callable	AcData Adapter	Returns the input adapter for this data filter
NewInputAdapter( )	Overridable	AcData Adapter	Instantiates the input adapter
SetInput( )	Callable	N/A	Specifies the input adapter for this data filter

AcDataRowBuffer

A subclass of AcSingleInputFilter, AcDataRowBuffer is a data filter that converts a sequential data stream into one which supports random access by buffering data rows. AcDataRowBuffer methods are listed in Table 4-54.

Table 4-54 AcDataRowBuffer methods

Method	Classification	Type	Description
AddRowToBuffer( )	Callable	N/A	Programmatically adds a row to the data row buffer. Typically called during a fetch, this method can be called by the report to save rows for later reuse.

Table 4-54 AcDataRowBuffer methods

Method	Classification	Type	Description
GetBufferCount( )	Callable	Integer	Gets the number of rows currently in the buffer.
GetBufferStart( )	Callable	Integer	Gets the position of the first row in the buffer, relative to the beginning of the input set. The first row is 1.

AcDataRowSorter

A subclass of AcDataRowBuffer, AcDataRowSorter is a data filter that reads and stores data rows. AcDataRowSorter provides a framework for subclasses to implement a sort algorithm. AcDataRowSorter methods are listed in Table 4-55.

Table 4-55 AcDataRowSorter methods

Method	Classification	Type	Description
Compare( )	Overridable	Integer	A pure virtual method that must be overridden to implement the comparison logic
CompareKeys( )	Callable	Integer	Compares two strings or numbers

AcDataSource

AcDataSource, a subclass of AcDataAdapter, is the base class for data adapters that read data from an input source. AcDataSource defines the logic for retrieving data from an external source and creating a data row for each input record. The AcDataSource method is described in Table 4-56.

Table 4-56 AcDataSource methods

Method	Classification	Type	Description
HasFetchedLast( )	Callable	Boolean	Determines whether the data source has fetched the last row

AcDatabaseSource

A subclass of AcDataSource, AcDatabaseSource is an abstract base class that provides the standard logic for retrieving rows from a relational database cursor. AcDatabaseSource methods are listed in Table 4-57.

Table 4-57 AcDatabaseSource methods

Method	Classification	Type	Description
BindDataRow( )	Overridable	N/A	Binds the data row to the cursor
BindStaticParameters( )	Overridable	N/A	Binds parameters to a statement
GetCursor( )	Callable	AcDBCursor	Gets the database cursor object associated with this data source
GetDBConnection( )	Callable	AcDB Connection	Gets the database connection against which to run this data source
GetPrepared Statement( )	Callable	AcDB Statement	Gets the statement on which to execute the cursor
OpenCursor( )	Callable	N/A	Opens a cursor on a statement
SetStatementProperty( )	Callable	N/A	Assigns a value to the specified property

AcExternalDataSource

AcExternalDataSource is a subclass of AcDatabaseSource. AcExternalDataSource is an abstract base class for generic data source objects that use a command to retrieve a single result set through a connection. The AcExternalDataSource method is described in Table 4-58.

Table 4-58 AcExternalDataSource methods

Method	Classification	Type	Description
ObtainCommand( )	Overridable	String	Obtains the command that retrieves the result set from the database

AcOdaSource

AcOdaSource is a subclass of AcExternalDataSource. AcOdaSource creates an object for an open data access (ODA) data source. AcOdaSource methods are listed in Table 4-59.

Table 4-59 AcOdaSource methods

Method	Classification	Type	Description
ClearSortKeys( )	Callable	N/A	Removes all previously assigned dynamic sort keys.
Commit( )	Callable	N/A	Commits all outstanding transactions on the specified ODA connection.

Table 4-59 AcOdaSource methods

Method	Classification	Type	Description
GetOutputParameter( )	Callable	Variant	Retrieves the output value of a specified output parameter as the parameter's default Actuate data type.
GetOutputParameterAs Type( )	Callable	Variant	Retrieves the output value of a specified output parameter and converts that value to the specified Actuate data type.
GetOutputParameters( )	Overridable	N/A	Calls GetOutputParameter( ) to retrieve the output value of each defined output parameter.
Rollback( )	Callable	N/A	Applies only if the ODA driver supports this feature. Rolls back all outstanding transactions on the specified ODA connection.
SetInputParameter( )	Callable	N/A	Assigns an input value to a specified input parameter.
SetInputParameters( )	Overridable	N/A	Calls SetInputParameter( ) to assign input values to each input parameter.
SetRuntimeProperties( )	Overridable	N/A	Assigns a value to each public and private run-time property.
SetStatement Attributes( )	Overridable	N/A	Sets attributes on the prepared statement before executing the statement or allocating a cursor.
StartNextSet( )	Callable	Boolean	Starts the next result set on the allocated cursor if the result set is not referenced by name.

AcQuerySource

A subclass of AcDatabaseSource, AcQuerySource is an abstract base class that provides the core logic for a query data source you build using Query Editor or Textual Query Editor. AcQuerySource methods are listed in Table 4-60.

AcSqlQuerySource

A subclass of AcQuerySource, AcSqlQuerySource creates a data source for a SELECT statement provided by the report, using the parts of the statement in the variables provided. AcSqlQuerySource assembles the variables to form the statement. There are no public methods defined specifically for this class.

Table 4-60 AcQuerySource methods

Method	Classification	Type	Description
GetStatementText( )	Callable	String	Returns the text of the SELECT statement for the query source.
ObtainSelect Statement( )	Overridable	String	Returns the SELECT statement of the query source.
SetupAdHoc Parameters( )	Overridable	N/A	The framework overrides this method to call AcSqlQuerySource ::SetAdHocParameter( ) or AcTextQuerySource::SetAdHoc Condition( ) repeatedly until all ad hoc parameters are processed.

AcTextQuerySource

A subclass of AcQuerySource, AcTextQuerySource is the class for writing textual SQL SELECT statements using the Textual Query Editor. There are no public methods defined specifically for this class.

AcStoredProcedureSource

A subclass of AcDatabaseSource, AcStoredProcedureSource is the base class for creating stored procedure data sources. The AcStoredProcedureSource method is described in Table 4-61.

Table 4-61 AcStoredProcedureSource methods

Method	Classification	Type	Description
GetOutputParameters( )	Overridable	N/A	Gets the output parameters for the stored procedure

AcDataRow

AcDataRow is a subclass of AcComponent. AcDataRow defines the characteristics of a data row. A data row is a record structure that contains data from a single record in a format that the report accepts. AcDataRow methods are listed in Table 4-62.

Table 4-62 AcDataRow methods

Method	Classification	Type	Description
GetValue( )	Callable	Variant	Gets the value of the specified column or variable

Table 4-62 AcDataRow methods (continued)

Method	Classification	Type	Description
OnRead( )	Overridable	N/A	Called after the associated data adapter has written its output to the data row
SetValue( )	Overridable	Boolean	Sets the value of the specified column or variable

Excel classes and methods

Use Excel classes and methods to generate Excel files.

AcExcelObject

Classes derived from AcExcelObject create and manage the Excel workbooks, worksheets, ranges, rows, columns, and cells you use in an Actuate report. There are no public methods defined specifically for this class. AcExcelObject does not inherit from other classes.

AcExcelApp

A subclass of AcExcelObject, AcExcelApp is the root class that contains all instances of classes you use to generate and work with Excel files. AcExcelApp methods are listed in Table 4-63.

Table 4-63 AcExcelApp methods

Method	Classification	Type	Description
AddWorkbook( )	Callable	AcExcel Workbook	Adds a new Excel file
DeleteWorkbook( )	Callable	Integer	Deletes a workbook
FindWorkbook( )	Callable	AcExcel Workbook	Finds a workbook
New( )	Callable	N/A	Creates an Excel application instance
SetFontScalingFactor( )	Callable	Integer	Specifies scaling factor to apply to a font

AcExcelRange

A subclass of AcExcelObject, AcExcelRange is the base class for the AcExcelCell, AcExcelColumn, and AcExcelRow classes. AcExcelRange methods are listed in Table 4-64.

Table 4-64 AcExcelRange methods

Method	Classification	Type	Description
AddImage( )	Callable	N/A	Adds an image to an Excel file
DrawLine( )	Callable	N/A	Sets properties of a line in the range
GetBackgroundColor( )	Callable	AcColor	Returns the background color
GetBorder( )	Callable	AcExcel Border	Returns the border
GetFont( )	Callable	AcFont	Returns the font
GetHorizontal Alignment( )	Callable	AcExcel Horizontal Alignment	Returns the horizontal alignment
GetIndent( )	Callable	Integer	Returns the number of indent characters
GetMergeCells( )	Callable	Boolean	Returns the setting of the merge cells option
GetNumberFormat( )	Callable	String	Returns the string used for formatting the numeric data
GetValue( )	Callable	Variant	Returns the contents of the range
GetValueAsDate( )	Callable	Date	Converts the contents of the range into date format
GetVerticalAlignment( )	Callable	AcExcel Vertical Alignment	Returns the vertical alignment
GetWrapText( )	Callable	Boolean	Returns the setting of the wrap text option
SetBackgroundColor( )	Callable	N/A	Sets the background color
SetBorder( )	Callable	N/A	Sets the border for one or more sides of the range
SetBorderAround( )	Callable	N/A	Sets the border around the entire range
SetFont( )	Callable	N/A	Sets the font
SetHorizontal Alignment( )	Callable	N/A	Sets the horizontal alignment
SetIndent( )	Callable	N/A	Sets the number of characters for the indent

Table 4-64 AcExcelRange methods

Method	Classification	Type	Description
SetMergeCells( )	Callable	N/A	Turns the merge cells option on and off
SetNumberFormat( )	Callable	N/A	Sets the format used for displaying numeric data
SetValue( )	Callable	Variant	Sets the contents for the range
SetVerticalAlignment( )	Callable	N/A	Sets the vertical alignment for the range
SetWrapText( )	Callable	N/A	Turns the wrap text option on and off

AcExcelCell

A subclass of AcExcelRange, AcExcelCell represents a cell in a worksheet. There are no public methods defined specifically for this class.

AcExcelColumn

A subclass of AcExcelRange, AcExcelColumn represents a column in a worksheet. AcExcelColumn methods are listed in Table 4-65.

Table 4-65 AcExcelColumn methods

Method	Classification	Type	Description
Autofit( )	Callable	Integer	Calculates the column width expressed as an integer
GetColumnWidth( )	Callable	Double	Returns the column width in number of characters that can be displayed in a column
SetAutofitFont( )	Callable	N/A	Sets the font to use to calculate column width
SetAutofitString( )	Callable	N/A	Sets the string to use to calculate column width
SetColumnWidth( )	Callable	N/A	Sets the column width

AcExcelRow

A subclass of AcExcelRange, AcExcelRow represents a row in a workbook. AcExcelRow methods are listed in Table 4-66.

Table 4-66 AcExcelRow methods

Method	Classification	Type	Description
GetRowHeight( )	Callable	Double	Returns the row height, in points
SetRowHeight( )	Callable	N/A	Sets the row height

AcExcelWorkbook

A subclass of AcExcelObject, AcExcelWorkbook adds, removes, and locates worksheets in a workbook. You also use AcExcelWorkbook to save and get the name of the workbook. AcExcelWorkbook methods are listed in Table 4-67.

Table 4-67 AcExcelWorkbook methods

Method	Classification	Type	Description
AddWorksheet( )	Callable	AcExcelWork Sheet	Adds a worksheet to the workbook
DeleteWorksheet( )	Callable	Integer	Deletes a worksheet from the workbook
FindWorksheet( )	Callable	AcExcelWork Sheet	Finds a worksheet in the workbook
GetFullName( )	Callable	String	Returns the name of the workbook
Save( )	Callable	N/A	Saves the workbook
SaveAs( )	Callable	N/A	Saves the workbook with the specified file name

AcExcelWorksheet

A subclass of AcExcelObject, AcExcelWorksheet provides information about a specific worksheet in a workbook. AcExcelWorksheet methods are listed in Table 4-68.

Table 4-68 AcExcelWorksheet methods

Method	Classification	Type	Description
Autofit( )	Callable	Integer	Adjusts the column width to fit the contents
GetCell( )	Callable	AcExcelCell	Returns the handle to the cell to access
GetColumn( )	Callable	AcExcel Column	Returns a handle to the column to access
GetDisplayGridlines( )	Callable	Boolean	Returns the gridline settings
GetName( )	Callable	String	Returns the name of the worksheet

Table 4-68 AcExcelWorksheet methods

Method	Classification	Type	Description
GetRange( )	Callable	AcExcel Range	Returns the handle to the cells to access
GetRow( )	Callable	AcExcelRow	Returns the handle to the row to access
SetDisplayGridlines( )	Callable	N/A	Turns the gridlines on or off
SetName( )	Callable	Integer	Sets the name of the worksheet

Visitor class and methods

Use the AcVisitor class and methods to customize processing on a report component.

AcVisitor

AcVisitor creates a utility to visit a report component and perform an action on the component. AcVisitor does not inherit from other classes. AcVisitor methods are listed in Table 4-69.

Table 4-69 AcVisitor methods

Method	Classification	Type	Description
VisitBaseFrame( )	Overridable	N/A	Visits a base frame component
VisitBasePage( )	Overridable	N/A	Visits a base page component
VisitChart( )	Overridable	N/A	Visits a chart component
VisitComponent( )	Overridable	N/A	Visits components of a report
VisitConditional Section( )	Overridable	N/A	Visits a conditional section component
VisitContents( )	Overridable	N/A	Visits the contents of a report's data hierarchy components
VisitControl( )	Overridable	N/A	Visits a control component
VisitCurrencyControl( )	Overridable	N/A	Visits a currency control component
VisitDataControl( )	Overridable	N/A	Visits a data control component
VisitDataFrame( )	Overridable	N/A	Visits a data frame component
VisitDataSection( )	Overridable	N/A	Visits a data section component
VisitDateTimeControl( )	Overridable	N/A	Visits a date and time control component

(continues)

Table 4-69 AcVisitor methods (continued)

Method	Classification	Type	Description
VisitDoubleControl( )	Overridable	N/A	Visits a double control component
VisitFlow( )	Overridable	N/A	Visits a flow component
VisitFrame( )	Overridable	N/A	Visits a frame component
VisitGroupSection( )	Overridable	N/A	Visits a group section component
VisitImageControl( )	Overridable	N/A	Visits an image control component
VisitIntegerControl( )	Overridable	N/A	Visits an integer control component
VisitLabelControl( )	Overridable	N/A	Visits a label control component
VisitLeftRightPage List( )	Overridable	N/A	Visits a left-right page list component
VisitLinearFlow( )	Overridable	N/A	Visits a linear flow component
VisitLineControl( )	Overridable	N/A	Visits a line control component
VisitPage( )	Overridable	N/A	Visits a page component
VisitPages( )	Overridable	N/A	Visits the contents of the report's page hierarchy components
VisitPageList( )	Overridable	N/A	Visits a page list component
VisitPageNumber Control( )	Overridable	N/A	Visits a page number control component
VisitParallelSection( )	Overridable	N/A	Visits a parallel section component
VisitRectangleControl( )	Overridable	N/A	Visits a rectangle component
VisitReport( )	Overridable	N/A	Visits an AcReport component
VisitReport Component( )	Overridable	N/A	Visits an AcReportComponent component
VisitReportSection( )	Overridable	N/A	Visits a report section component
VisitSection( )	Overridable	N/A	Visits a section component
VisitSequentialSection( )	Overridable	N/A	Visits a sequential section component
VisitSimplePageList( )	Overridable	N/A	Visits a simple page list component
VisitSubpage( )	Overridable	N/A	Visits a subpage component
VisitTextControl( )	Overridable	N/A	Visits a text control component
VisitTextualControl( )	Overridable	N/A	Visits a textual control component
VisitTitleBodyPage List( )	Overridable	N/A	Visits a title and body page list
VisitVisual Component( )	Overridable	N/A	Visits a visual component

Chapter

5

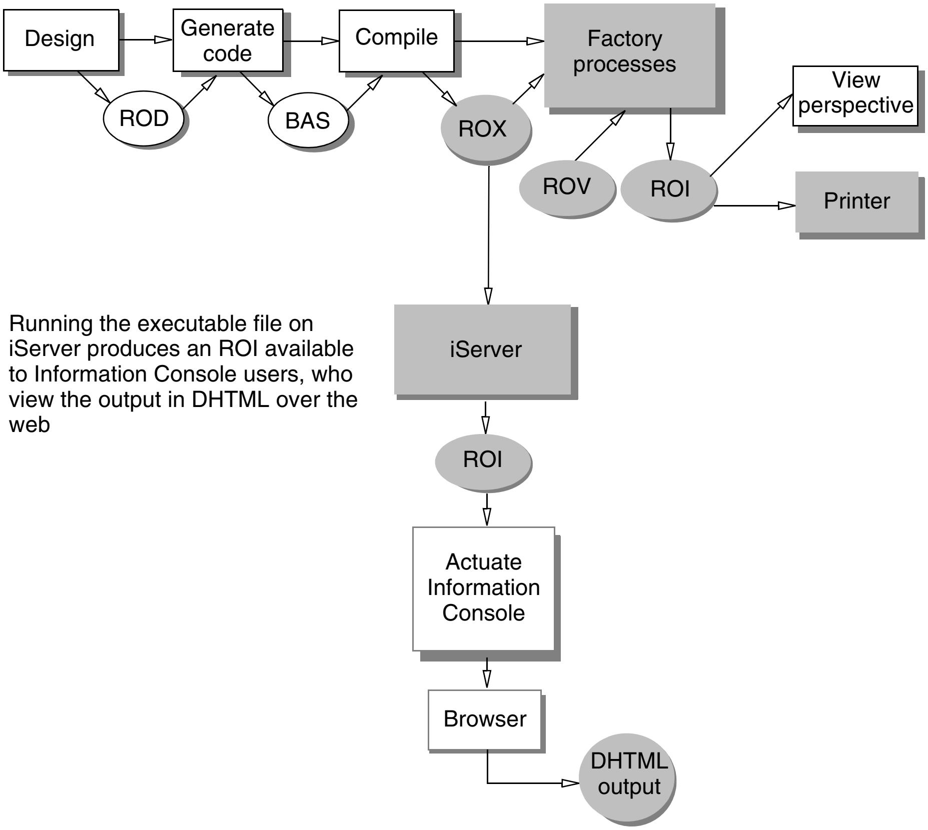
Understanding report generation

This chapter contains the following topics:

- Understanding the report generation process
- Creating content
- Understanding page creation

Understanding the report generation process

The Factory manages the processes that run an executable file and display or print a file. Factory processes result in a report object instance (.roi) file, which consists of persistent objects. The report developer can print the output or view the file locally using e.Report Designer Professional's View perspective. If you publish the executable file to iServer, the report user views the output in DHTML using a web browser and a tool such as Actuate Information Console. Figure 5-1 shows the Factory operations that occur during the build process and the file types that result from those processes.



Running the executable file on iServer produces an ROI available to Information Console users, who view the output in DHTML over the web

Figure 5-1 Factory processes result in on-screen display or printed output

The process that results in a report instance begins when a data stream delivers a data row to a section. The section creates its contents, which can be another section or a frame and controls. The section then passes the frame to the page list, which instantiates pages and flows as necessary. Figure 5-2 gives an overview of this process.

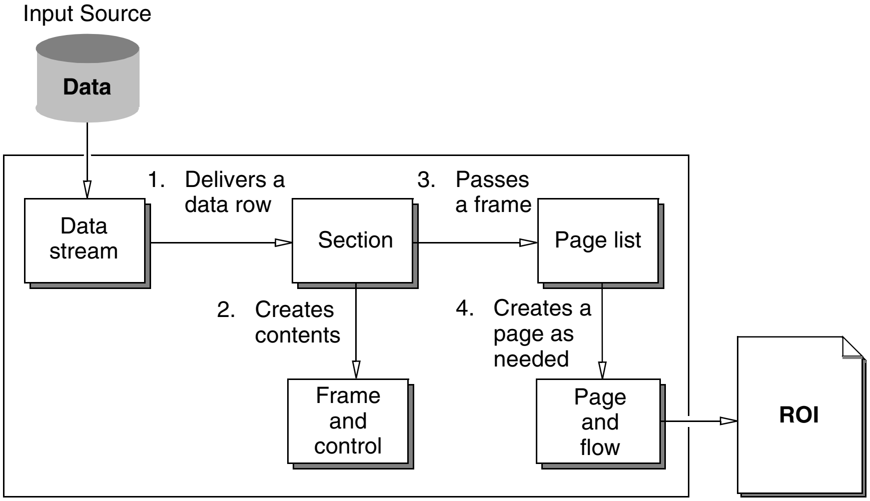


Figure 5-2 Overview of the process of creating an ROI

The following sections describe how the Factory generates a report and the class protocols that determine how objects in a report fit together.

Generating a report

When you build and run a report, Actuate software executes an internal method called Factory( ), which is specific to your file. Factory( ) performs the following tasks:

- Creates a report object instance (.roi) file. The Factory uses either a default name with the same root name as the report object design (.rod) file or the name the user specifies when Requester prompts for the output file name.
- Creates a component relationship map. To show how components interact, the component relationship map stores component reference information derived from the report design's structure.
- Instantiates the report as a subclass of AcReport.
- Calls the file's Build( ) method to start the report-generation process.
- Closes the ROI.

Adding startup and cleanup code

You can add code that runs before the Factory initiates the report-generation task and after it generates the report. You do so by overriding the report's Start( ) and Finish( ) methods. Table 5-1 describes the use of the Start( ) and Finish( ) methods.

Table 5-1 Using the Start( ) and Finish( ) methods

Method	Called ...	Example of use
Start( )	After the Factory instantiates the report, before report generation begins	■ Initialize a global variable. ■ Verify or adjust parameter values a user entered. ■ Open a log file to track the objects or the number of pages the report creates. If you override Start( ), you must call Super::Start( ) first.
Finish( )	After the report generates, before the ROI closes	■ Send a completion notice to the user. ■ Write statistics to a log file. If you override Finish( ), you must call Super::Finish( ) after your code.

Starting the build process

One of the Factory's first tasks is to instantiate the top-level report object. The report object, a subclass of AcReport, is the object that contains all other components of a report. AcReport establishes the report's content and page structures.

The content structure consists of objects that contain data. The page structure consists of objects that determine how to display report content. Figure 5-3 shows examples of the two structures.

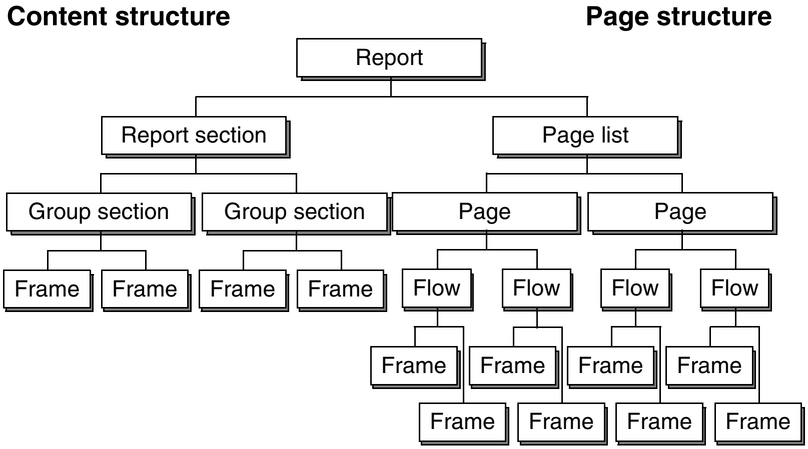


Figure 5-3 Overview of content structure and page structure

After instantiating the report class, the Factory calls the report's Build( ) method, which performs the following tasks:

- Calls NewPageList( ) to instantiate the page list the report design specifies.
- Calls NewContent( ) to instantiate the top-level component the report design specifies. In a typical design, the top-level component is a report.
- Calls the top-level content's Build( ) method to build report content.

Creating content

All report components that contain content, such as a section, a frame, or a control, follow a protocol that determines how to build the content. The protocol makes it possible to connect report components into a variety of configurations. For example, the top-level Content component can be a report section, a sequential section, or a frame. You can nest a report within a report, a section within a report, a section within a section, a frame within a frame, and so on. The configuration determines the order in which to create the components.

Understanding how the core protocol creates content

AcReportComponent is the abstract base class that defines the protocol for creating report components and putting them together to form a report. Table 5-2 describes the methods of AcReportComponent that form the core content-creation protocol for all persistent content objects.

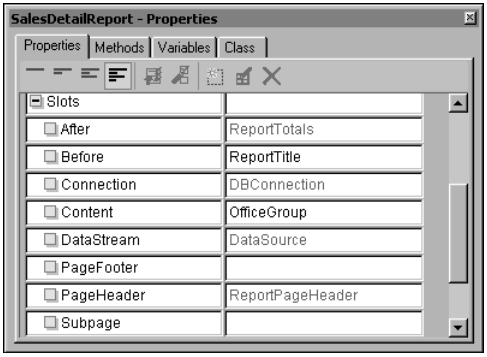
Table 5-2 Methods that form the core content-creation protocol

Method	Description
New( )	Initializes the object.
Start( )	Prepares the object for build operations. For example, a report section's Start( ) method instantiates the connection and the data stream. The frame's Start( ) method instantiates the controls the frame contains.
Build( ) or BuildFromRow( )	Builds the object's contents. For example, a report section's Build( ) method reads data rows from the data stream, instantiates the contents, and passes the data rows to them. The frame's BuildFromRow( ) method passes the data rows to the controls it contains.
Finish( )	Prepares the object to write to the report object instance (.roi) file. For example, the report section's Finish( ) method closes the data stream and connection.

Understanding how a component reference creates content

A component reference defines the relationship among components in the structure of the report design. When a component uses or contains another component, the first component refers to the other component. For example, the report object creates and refers to the report section contained immediately within it. The references that develop between a container and its contents determine the hierarchy of objects in the report structure. The hierarchy determines which processes occur and the order in which those processes occur. The components in a report follow a predefined set of component reference rules that you cannot change.

The Slots property group of a component shows the references for the component. For example, Figure 5-4 shows the Slots property group for a report section.



⊟ Slots	
After	ReportTotals
Before	ReportTitle
Connection	DBConnection
Content	OfficeGroup
DataStream	DataSource
PageFooter	
PageHeader	ReportPageHeader
Subpage	

Figure 5-4 Slots property group

Every report section has all of these references available, even if the slots contain no components.

It is possible for a container to have more than one component reference of the same type. For example, a parallel or sequential section can have multiple Content slots, and a frame can contain multiple components, all bearing the same type of component reference. When there are multiple component references, you can control the order in which the component references occur by using the up and down arrows in the Report Structure toolbar. If the components you want to move are references to library components, you must change the order in the library. Figure 5-5 shows how the component reference relationships determine how the Factory builds report contents.

Each component reference that a component supports has a corresponding method. This method has a New prefix followed by the name of the reference. For example, AcReportSection's connection component reference has a corresponding NewConnection( ) method and its data stream component reference has a corresponding NewDataStream( ) method.

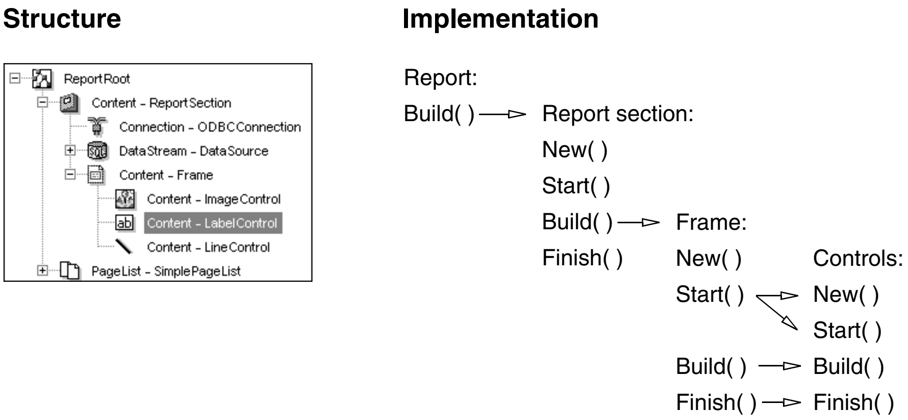


Figure 5-5 Overview of how report structure drives report generation

A container object uses the New<component_reference> methods to instantiate its content. To conditionally instantiate a component, you can override the New<component_reference> method. For example, if a report uses a different connection depending on the data stream it uses, you can override NewConnection( ) to write the conditional logic. To determine where to place code to add custom processing, you must understand the key methods involved in content creation and how each component implements the core content-creation protocol. The following sections provide this information.

Understanding how a report section creates content

The report section retrieves data rows from the data stream and passes the rows to its contents. The report section, a subclass of AcReportSection, is typically the top-level content in a report design. Table 5-3 describes how the report section implements the core content-creation protocol.

Table 5-3 Core methods that a report section uses to create content

Method	Task
Start( )	Instantiates the connection. Instantiates the data stream. Passes the connection to the data stream. Sets the sort key.
Build( )	Opens the data stream. Creates the Before and After frames. Reads a row from the data stream.

(continues)

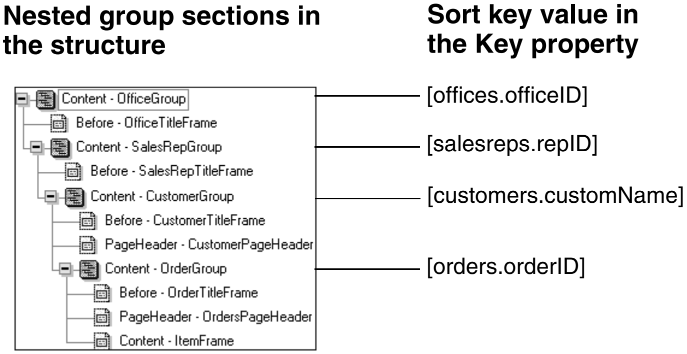
Table 5-3 Core methods that a report section uses to create content (continued)

Method	Task
Build( ) *(continued)*	Processes the row: ■ If content already exists, the report verifies that the content needs the row. The report passes the row to the content if the content needs it. ■ If content does not exist or if an existing component rejects a row, the report instantiates new content and passes the row to the new content. Reads and processes rows until it retrieves all data rows.
Finish( )	Closes the data stream and connection.

Understanding how a group section creates content

A group section organizes data by grouping rows on a key field, such as grouping customers by sales representative. A group section is typically the content of a report section. A group section contains other group sections or frames.

Each group section uses a unique sort key to define a group of rows. Set the group section's sort key, typically the name of a table column, using the Key property. Figure 5-6 shows typical sort keys for a report's group sections.



Nested group sections in the structure
Sort key value in the Key property
Content - OfficeGroup
Before - OfficeTitleFrame
Content - SalesRepGroup
Before - SalesRepTitleFrame
Content - CustomerGroup
Before - CustomerTitleFrame
PageHeader - CustomerPageHeader
Content - OrderGroup
Before - OrderTitleFrame
PageHeader - OrdersPageHeader
Content - ItemFrame
[offices.officeID]
[salesreps.repID]
[customers.customName]
[orders.orderID]

Figure 5-6 Group sections and sort keys

The group section tests the key of each data row it receives. Rows using the same key value belong in the same group section. A change in a key's value between rows indicates the end of one group section and the start of another. Table 5-4 describes how the group section implements the core content-creation protocol.

Understanding how a frame creates content

A frame typically contains controls that display data from a data row. A frame gets data rows from its container object, typically a section, and passes those rows to the controls. Table 5-5 describes how a frame implements the core content-creation protocol.

Table 5-4 Core methods that a group section uses to create content

Method	Task
Start( )	Initializes the group section.
BuildFromRow( )	Accepts a data row from its container object. Processes the row: ■ If the row is the first one, the group section determines and stores the value of the sort key. The group section also creates the Before and After frames, passes the row to its content, and returns ContinueBuilding. This return value indicates to the container object that the group section needs the next data row. ■ If the row is not the first one, the section verifies that the row's key is the same as the stored key value. If the keys match, the group section passes the row to its content. If the keys do not match, BuildFromRow( ) returns Rejected Row, indicating the end of a group section. Repeats these steps until it retrieves all data rows, returning ContinueBuilding.
Finish( )	Finishes the group section.

Table 5-5 Core methods that a frame uses to create content

Method	Task
Start( )	Instantiates and starts the frame's contents in the order in which they appear in the structure. The frame's contents can be other frames or controls.
BuildFromRow( )	Passes data rows to the frame's contents.
Finish( )	Calls each control's Finish( ) method.

Understanding how a control creates content

A control typically displays a value from the data row it receives from its container, a frame. A control does not contain other components. Table 5-6 describes how a control implements the core content-creation protocol.

Table 5-6 Core methods that a control uses to create content

Method	Task
Start( )	Typically, a control needs only the default logic.
BuildFromRow( )	Sets the control's value using data from a data row: ■ If a control, such as a line or label control, does not need the data row, BuildFromRow( ) returns FinishedBuilding. ■ If a control needs only one row, BuildFromRow( ) sets the value of the control and returns FinishedBuilding. ■ If a control is an aggregate control, it uses all data rows. BuildFromRow( ) returns ContinueBuilding.
Finish( )	For an aggregate control, performs final calculations. For other controls, Finish( ) does nothing.

Understanding page creation

The content-creation process drives the page-creation process. The two processes occur concurrently. As each frame completes, the section that contains the frame passes the frame to the page list. As each page begins or ends, the page list notifies the section and the section generates a page header or footer.

Figure 5-7 shows how the content-creation and page-creation processes work together.

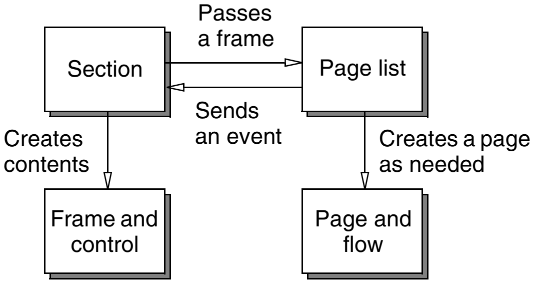


Figure 5-7 Interaction of the content-creation and page-creation processes

Figure 5-8 shows the order in which headers and footers go into a report.

Determining the page on which a frame appears

A section passes a frame to the page list by calling the page list's AddFrame( ) method. AddFrame( ) determines the page on which to place the frame by checking the following conditions:

- If the frame's PageBreakBefore property is True, AddFrame( ) finishes the current page, if one exists, then starts a new page.
- If the frame's PageBreakBefore property is False, AddFrame( ) adds the frame to the page if it fits. If it does not fit, the page list starts a new page.
- If the frame does not fit on a new page, AddFrame( ) clips the frame to the available space.
- After placing the frame on a page, AddFrame( ) checks the value of the frame's PageBreakAfter property. If PageBreakAfter is True, AddFrame( ) finishes the current page. AddFrame( ) starts a new page when it gets the next frame.

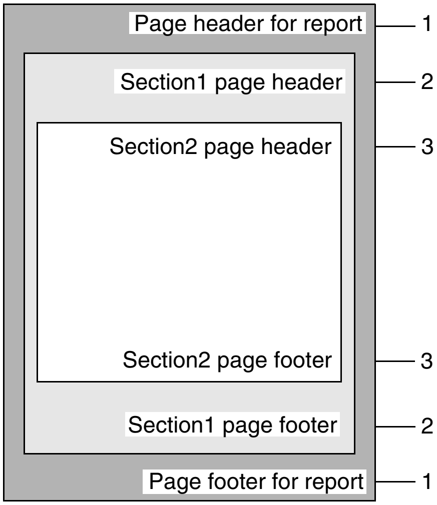


Figure 5-8 The order in which headers and footers are added to a page

About page list styles

There are three page list styles:

- AcSimplePageList, in which each page uses the same design
- AcLeftRightPageList, in which the right and left pages mirror each other
- AcTitleBodyPageList, in which the title page uses a different design from other pages

A section can work on any style because a page list follows a standard protocol. The protocol, defined in AcPageList, consists of the AddFrame( ) method. Before instantiating a new page, the page list checks whether a section or frame sets a page style in its NewPage( ) method. If a frame or section sets a page style, the page list uses that style. Otherwise, the page list uses its default page style.

About page list events

A page list sends a notice to a section when any of the following events occurs:

- StartFlow
- StartPage
- FinishFlow
- FinishPage

These events determine whether the framework places a header at the start of a new flow or page, or a footer at the end of a flow or page.

Part Two

Actuate Foundation Class Reference

Chapter 6

AFC data types

This chapter covers the topic "About the AFC data types."

About the AFC data types

Actuate products use two categories of data types, those provided by Actuate Basic, and those that are defined specifically for use with Actuate Foundation Classes (AFC). This chapter discusses the AFC data types. For information about Actuate Basic data types, see *Programming with Actuate Basic.* AFC data types include aliased types, structures, and enums.

About AFC aliased types

AFC defines aliases for some Actuate Basic data types. These aliases are recognized and handled specially by e.Report Designer Professional. For example, AcColor is an alias of Actuate Basic's Integer data type, but e.Report Designer Professional presents an AcColor property as a drop-down list of colors with a custom color picker. Sections that follow describe the purpose and behavior of each AFC aliased type.

About AFC structures

A structure is a data type that contains multiple named values, called members. The members of a structure can be Visual Basic data types, such as Integer or Boolean, or AFC data types. For example, AcDrawingLineStyle is a structure that defines the format for a line. Its members, Color, Pen, and Width, define the line color, the pattern of the line, and the line width, respectively. Structures can be nested. In other words, structure members can be structures. e.Report Designer Professional displays a structure property as an expandable group of values. Sections that follow list the members for each AFC structure.

About AFC enums

An enum is a data type whose value is one of a set of named values. e.Report Designer Professional displays an enum property as drop-down list of values. For example, the value of a TrafficLightColor enum might be Red, Yellow, or Green. Sections that follow list the values for each enum defined in AFC.

AFC data types

The following sections describe all the AFC data types.

AcAutoSplit

AcAutoSplit is an enum that specifies how a component, such as a cross tab or frame, splits into multiple flows. AcAutoSplit values are listed in Table 6-1.

Table 6-1 AcAutoSplit values

Constant	Description
DefaultSplitting	If the component is a frame that contains dynamically sized content, such as a crosstab or dynamic text control, or it is a dynamically sized control, it may be split to maximize use of space within a flow.
DoNotSplit	The component must not be split.
SplitIfPossible	The component splits to maximize use of space within a flow.
SplitIfNecessary	The component splits only if it cannot fit as or within the first non-decoration frame in a flow.

AcBrowserClipping

AcBrowserClipping is an enum that specifies how to clip text in a browser scripting control when it is viewed in a web browser. AcBrowserClipping values are listed in Table 6-2.

Table 6-2 AcBrowserClipping values

Constant	Description
AutoScrollbar	Display scrollbars when necessary to support viewing text that does not fit in the control.
ClipToControlSize	Clip text to fit within the control.
NoClipping	Allow text to overflow the bounds of the control.
Scrollbar	Always display scrollbars.

AcChartAxisLabelPlacement

AcChartAxisLabelPlacement is an enum that specifies the placement of the labels on a chart axis. AcChartAxisLabelPlacement values are listed in Table 6-3.

Table 6-3 AcChartAxisLabelPlacement values

Constant	Description
ChartAxisLabel PlacementNone	Does not display the axis labels.
ChartAxisLabel PlacementNextToAxis	Places the axis label next to the axis.
ChartAxisLabel PlacementLeftOr Bottom	If the axis is vertical, places the axis labels at the left of the chart. If the axis is horizontal, places the axis labels at the bottom of the chart.
ChartAxisLabel PlacementRightOrTop	If the axis is vertical, places the axis labels at the right of the chart. If the axis is horizontal, places the axis labels at the top of the chart.

AcChartAxisLetter

AcChartAxisLetter is an enum that specifies the type of a chart axis. AcChartAxisLetter values are listed in Table 6-4.

Table 6-4 AcChartAxisLetter values

Constant	Description
ChartAxisLetterX	The axis is an *x*-axis.
ChartAxisLetterY	The axis is a *y*-axis.
ChartAxisLetterZ	The axis is a *z*-axis.

AcChartAxisPlacement

AcChartAxisPlacement is an enum that specifies the placement of a chart axis. AcChartAxisPlacement values are listed in Table 6-5.

Table 6-5 AcChartAxisPlacement values

Constant	Description
ChartAxisPlacement Auto	Places the chart axis automatically. If the opposite axis includes zero, the axis crosses it at zero. If all the values on the opposite axis are positive, the axis crosses it at the lowest value. If all the values on the opposite axis are negative, the axis crosses it at the highest value.

Table 6-5 AcChartAxisPlacement values

Constant	Description
ChartAxisPlacement Custom	The chart axis crosses the opposite axis at a specified value.
ChartAxisPlacement LeftOrBottom	If the axis is vertical, places it at the left of the chart. If the axis is horizontal, places it at the bottom of the chart.
ChartAxisPlacement RightOrTop	If the axis is vertical, places it at the right of the chart. If the axis is horizontal, places it at the top of the chart.

AcChartBarShape

AcChartBarShape is an enum that specifies the cross-section of a three-dimensional bar in a chart. AcChartBarShape values are listed in Table 6-6.

Table 6-6 AcChartBarShape values

Constant	Description
ChartBarShapeElliptical	Cylinder
ChartBarShapeFlat	Flat two-dimensional rectangle
ChartBarShapeHexagonal	Hexagon
ChartBarShapeOctagonal	Octagon
ChartBarShapeRectangular	Rectangle
ChartBarShapeTriangular	Triangle

AcChartComparisonOperator

AcChartComparisonOperator is an enum that specifies the Boolean operator to use to compare two values in a chart. Valid values are listed in Table 6-7.

Table 6-7 AcChartComparisonOperator values

Constant	Description
ChartComparisonOperatorEQ	=
ChartComparisonOperatorGE	>=
ChartComparisonOperatorGT	>

(continues)

Table 6-7 AcChartComparisonOperator values (continued)

Constant	Description
ChartComparisonOperatorLE	<=
ChartComparisonOperatorLT	<
ChartComparisonOperatorNone	No comparison is required.

AcChartDefaultMarkerSettings

AcChartDefaultMarkerSettings is a structure that defines the default shape and color for a chart marker. AcChartDefaultMarkerSettings members are listed in Table 6-8.

Table 6-8 AcChartDefaultMarkerSettings members

Member name	Type	Description
Filled	Boolean	Indicates whether the default marker uses a fill color
Shape	AcChart MarkerShape	Indicates the default shape of the marker

AcChartLayerType

AcChartLayerType is an enum that specifies the type of a layer in a chart. AcChartLayerType values are listed in Table 6-9.

Table 6-9 AcChartLayerType values

Constant	Description
ChartLayerTypeBase	The layer is the base layer of a chart.
ChartLayerTypeOverlay	The layer is the overlay layer of a chart.
ChartLayerTypeStudy	The layer is one of the study layers of a chart.

AcChartLegendPlacement

AcChartLegendPlacement is an enum that specifies the placement of the legend of a chart. AcChartLegendPlacement values are listed in Table 6-10.

Table 6-10 AcChartLegendPlacement values

Constant	Description
ChartLegendPlacementBottom	Displays the legend at the bottom of the chart
ChartLegendPlacementBottom Left	Displays the legend at the bottom left of the chart
ChartLegendPlacementBottom Right	Displays the legend at the bottom right of the chart
ChartLegendPlacementLeft	Displays the legend at the left of the chart
ChartLegendPlacementNone	Does not display the legend
ChartLegendPlacementRight	Displays the legend at the right of the chart
ChartLegendPlacementTop	Displays the legend at the top of the chart
ChartLegendPlacementTopLeft	Displays the legend at the top left of the chart
ChartLegendPlacementTop Right	Displays the legend at the top right of the chart

AcChartMarkerShape

AcChartMarkerShape is an enum that specifies the shape of a point marker in a chart. AcChartMarkerShape values are listed in Table 6-11.

Table 6-11 AcChartMarkerShape values

Constant	Description
ChartMarkerShapeCircle	Circle
ChartMarkerShapeClose	Stock chart Close symbol: a small horizontal dash offset to the right of the point
ChartMarkerShapeCross	Diagonal cross
ChartMarkerShapeDiamond	Diamond
ChartMarkerShapeHigh	Stock chart High symbol: a wide horizontal dash
ChartMarkerShapeLow	Stock chart Low symbol: a wide horizontal dash
ChartMarkerShapeNone	No marker

(continues)

Table 6-11 AcChartMarkerShape values (continued)

Constant	Description
ChartMarkerShapeOpen	Stock chart Open symbol: a small horizontal dash offset to the left of the point
ChartMarkerShapePlus	Plus sign
ChartMarkerShapeSquare	Square
ChartMarkerShapeStar	Star
ChartMarkerShapeTriangle Down	Triangle pointing down
ChartMarkerShapeTriangleUp	Triangle pointing up

AcChartMissingPoints

AcChartMissingPoints is an enum that specifies how to handle missing points in a chart. AcChartMissingPoints values are listed in Table 6-12.

Table 6-12 AcChartMissingPoints values

Constant	Description
ChartMissingPointsDoNotPlot	Does not plot missing points. If the chart is a line chart, the lines break each side of missing points.
ChartMissingPointsInterpolate	Plots missing points as points whose values are linear interpolations of the points either side of them. If the chart is a line chart, does not display markers at the missing points, but keeps the lines unbroken.
ChartMissingPointsPlotAsZero	Plots missing points as points with zero values.

AcChartPieExplode

AcChartPieExplode is an enum that specifies how to explode sectors in a pie chart. AcChartPieExplode values are listed in Table 6-13.

Table 6-13 AcChartPieExplode values

Constant	Description
ChartPieExplodeAllSlices	Explodes all sectors

Table 6-13 AcChartPieExplode values

Constant	Description
ChartPieExplodeNone	Does not explode sectors
ChartPieExplodeSpecificSlices	Explodes only those sectors that are explicitly flagged to be exploded

AcChartPointHighlight

AcChartPointHighlight is an enum that specifies how to highlight a point in a chart. AcChartPointHighlight values are listed in Table 6-14.

Table 6-14 AcChartPointHighlight values

Constant	Description
ChartPointHighlightExplode	The point is an exploded sector in a pie chart.
ChartPointHighlightNone	Does not highlight the point.

AcChartPointLabelPlacement

AcChartPointLabelPlacement is an enum that specifies where to place a point label in a chart. AcChartPointLabelPlacement values are listed in Table 6-15.

Table 6-15 AcChartPointLabelPlacement values

Constant	Description
ChartPointLabelPlacement Above	Displays the label above the point. This placement is supported for line, scatter, and stock charts.
ChartPointLabelPlacementAuto	Places the label automatically to give a reasonable appearance. This placement is supported for all chart types.
ChartPointLabelPlacemen Below	Displays the label below the point. This placement is supported for line, scatter, and stock charts.
ChartPointLabelPlacement Center	For line, scatter, and stock charts, displays the label centered on the point. For area, bar, pie, and step charts, displays the label

(continues)

Table 6-15 AcChartPointLabelPlacement values (continued)

Constant	Description
ChartPointLabelPlacement Center *(continued)*	centered in the area, bar, pie sector, or step representing the point. This placement is supported for all chart types.
ChartPointLabelPlacement InsideBase	Displays the label inside the base of the bar representing the point (next to the x-axis). This placement is supported for bar charts.
ChartPointLabelPlacement InsideEnd	Displays the label inside the outer end of the bar or pie sector representing the point. This placement is supported for bar and pie charts.
ChartPointLabelPlacementLeft	Displays the label to the left of the point. This placement is supported for line, scatter, and stock charts.
ChartPointLabelPlacementNone	Does not display a label. This placement is supported for all chart types.
ChartPointLabelPlacement OutsideEnd	Displays the label outside the outer end of the bar or pie sector representing the point. This placement is supported for bar and pie charts.
ChartPointLabelPlacementRight	Displays the label to the right of the point. This placement is supported for line, scatter, and stock charts.

AcChartPointLabelSource

AcChartPointLabelSource is an enum that specifies how to calculate the value of a point label in a chart. AcChartPointLabelSource values are listed in Table 6-16.

Table 6-16 AcChartPointLabelSource values

Constant	Description
ChartPointLabelSourceCategory	The label value is the category label value for the point's category. This setting is not supported for bubble and scatter charts.

Table 6-16 AcChartPointLabelSource values (continued)

Constant	Description
ChartPointLabelSourceCategoryAndPercentage	In a pie chart, the label value is the category label value for the sector's category, followed by the sector's value as a percentage of the whole pie. In other chart types, the label value is the category label value for the point's category, followed by the point's value as a percentage of the point's category. This setting is supported only for pie and stacked charts.
ChartPointLabelSourceCustom	The label value is a custom point label value stored in the data point object.
ChartPointLabelSourcePercentage	In a pie chart, the label value is the sector's value as a percentage of the whole pie. In other types of chart, the label value is the point's y value as a percentage of the point's category. This setting is supported only for pie and stacked charts.
ChartPointLabelSourceSeries	The label value is the series label value for the point's series. This setting is not supported for pie charts.
ChartPointLabelSourceSeriesAndPercentage	The label value is the series label value for the point's series, followed by the point's y value as a percentage of the point's category. This setting is supported only for stacked charts.
ChartPointLabelSourceXValue	The label value is the point's x value. This setting is supported only for bubble and scatter charts.
ChartPointLabelSourceYValue	In a pie chart, the label value is the sector's value. In other types of chart, the label value is the point's y value.
ChartPointLabelSourceYValueAndPercentage	In a pie chart, the label value is the sector's value, followed by the sector's value as a percentage of the whole pie.

(continues)

Table 6-16 AcChartPointLabelSource values (continued)

Constant	Description
ChartPointLabelSourceYValue AndPercentage *(continued)*	In other types of chart, the label value is the point's y value, followed by the point's y value as a percentage of the point's category. This setting is supported only for pie and stacked charts.
ChartPointLabelSourceZValue	The label value is the point's z value. This setting is supported only for bubble charts.

AcChartSeriesPlacement

AcChartSeriesPlacement is an enum that specifies how to place the points in multiple series within a chart, relative to each other. AcChartSeriesPlacement values are listed in Table 6-17.

Table 6-17 AcChartSeriesPlacement values

Constant	Description
ChartSeriesPlacementAs Percentages	For each category, stacks the values for each series on top of each other such that the total height for all the series is always 100%. The *y*-axis displays percentages, not absolute values. This placement is supported for area, bar, line, and step charts.
ChartSeriesPlacement SideBySide	Places the values for each series side by side. All chart types support this placement.
ChartSeriesPlacementStacked	For each category, stacks the values for each series on top of each other. This placement is supported for area, bar, line, and step charts.
ChartSeriesPlacementOnZAxis	Places the values for each series front-to-back on the *z*-axis of a three-dimensional chart. This placement is supported for area, bar, and line charts.

AcChartStatus

AcChartStatus is an enum that specifies the stage a chart has reached in its life cycle. AcChartStatus values are listed in Table 6-18.

Table 6-18 AcChartStatus values

Constant	Description
ChartStatusBuilding	The chart is gathering and processing data and formatting itself.
ChartStatusFinished	
ChartStatusFinishedBuilding	The chart is complete and cannot be changed further.
ChartStatusUninitialized	The chart is still being initialized. This status lasts until all the layers within the chart have been created and initialized.

AcChartTickCalculation

AcChartTickCalculation is an enum that specifies how to calculate the spacing of the tick marks and labels on a chart axis. AcChartTickCalculation values are listed in Table 6-19.

Table 6-19 AcChartTickCalculation values

Constant	Description
ChartTickCalculationAuto	Calculates the spacing automatically, based on the data in the chart and the axis settings
ChartTickCalculationExact Interval	Specifies the spacing explicitly in the axis
ChartTickCalculationMinimum Interval	Calculates the spacing automatically in the same way as for the ChartTickCalculationAuto setting, except that the interval between ticks must be at least a specified value

AcChartTickPlacement

AcChartTickPlacement is an enum that specifies the placement of the tick marks on a chart axis. AcChartTickPlacement values are listed in Table 6-20.

Table 6-20 AcChartTickPlacement values

Constant	Description
ChartTickPlacementAcross	Ticks cross through the axis
ChartTickPlacementInside	Displays ticks inside the axis
ChartTickPlacementNone	Does not display ticks
ChartTickPlacementOutside	Displays ticks outside the axis

AcChartType

AcChartType is an enum that specifies the presentation of data on a chart layer. AcChartType values are listed in Table 6-21.

Table 6-21 AcChartType values

Constant	Description
ChartTypeArea	Presents data as filled areas. The *x*-axis shows categories, the *y*-axis shows values.
ChartTypeBar	Presents data as bars. The *x*-axis shows categories, the *y*-axis shows values.
ChartTypeBubble	Presents data as individual points, drawn as circles of varying sizes. Both the *x*-axis and the *y*-axis show values. The circles' sizes are controlled by the points' z values.
ChartTypeLine	Presents data as lines. The *x*-axis shows categories, the *y*-axis shows values.
ChartTypeNone	Does not specify a presentation. This setting causes the chart to throw a runtime error.
ChartTypePie	Presents data as a pie. Each sector in the pie represents a category. The layer has no axes.
ChartTypeScatter	Presents data as individual points. Both the *x*-axis and the *y*-axis show values.
ChartTypeStep	Presents data as filled steps. The *x*-axis shows categories, the *y*-axis shows values.
ChartTypeStock	Presents data as a stock chart. The *x*-axis shows a time series, the *y*-axis shows values.

AcColor

AcColor is an Integer that contains a color expressed as a standard Windows RGB value. AFC supports the standard Windows 16 million colors. The AFC framework defines the constants for common Windows colors. AcColor constants are listed in Table 6-22.

Table 6-22 AcColor constants

Constant	RGB Value
Black	RGB(0, 0, 0)
Blue	RGB(0, 0, 255)
BlueGray	RGB(89, 128, 179)
BrickRed	RGB(234, 70, 0)
Brown	RGB(204, 102, 26)
Coral	RGB(255, 115, 51)
Cream	RGB(255, 255, 166)
Crimson	RGB(198, 26, 26)
Cyan	RGB(0, 255, 255)
DarkGray	RGB(64, 64, 64)
DarkKhaki	RGB(189, 183, 107)
DarkStraw	RGB(204, 168, 0)
DeepPink	RGB(255, 26, 128)
Forest	RGB(0, 127, 0)
Gold	RGB(252, 217, 13)
GrassGreen	RGB(51, 191, 51)
Gray	RGB(128, 128, 128)
Green	RGB(0, 255, 0)
GreenYellow	RGB(128, 230, 26)
Khaki	RGB(240, 230, 140)
Lavender	RGB(230, 217, 255)
LightBlue	RGB(179, 204, 254)
LightBlueGray	RGB(179, 191, 217)
LightBrown	RGB(242, 166, 115)
LightCyan	RGB(166, 255, 255)

(continues)

Table 6-22 AcColor constants (continued)

Constant	RGB Value
LightGray	RGB(192, 192, 192)
LightGreen	RGB(166, 255, 153)
LightMagenta	RGB(242, 128, 242)
LightVioletRed	RGB(230, 153, 179)
LightYellowGreen	RGB(217, 255, 128)
Magenta	RGB(255, 0, 255)
Maroon	RGB(127, 0, 0)
MintGreen	RGB(0, 230, 102)
Navy	RGB(0, 0, 127)
Olive	RGB(127, 127, 0)
Orange	RGB(255, 166, 0)
PaleBlue	RGB(217, 230, 255)
PaleBlueGray	RGB(217, 230, 242)
PaleCyan	RGB(217, 255, 255)
PaleGray	RGB(230, 230, 230)
PaleGreen	RGB(217, 255, 191)
PaleMagenta	RGB(255, 217, 255)
PalePink	RGB(255, 217, 217)
PaleStraw	RGB(255, 242, 166)
PaleYellowGreen	RGB(230, 255, 166)
Pink	RGB(255, 179, 179)
Purple	RGB(127, 0, 127)
Red	RGB(255, 0, 0)
SeaGreen	RGB(0, 191, 191)
SkyBlue	RGB(102, 166, 255)
SmokeGray	RGB(242, 242, 242)
Straw	RGB(255, 230, 115)
Taupe	RGB(204, 179, 140)
Teal	RGB(0, 127, 127)
Transparent	N/A
Turquoise	RGB(89, 242, 217)

Table 6-22 AcColor constants (continued)

Constant	RGB Value
Violet	RGB(140, 51, 217)
VioletRed	RGB(204, 51, 102)
White	RGB(255, 255, 255)
Yellow	RGB(255, 255, 0)
YellowGreen	RGB(191, 230, 26)

AcControlValueType

AcControlValueType is an enum that determines whether a control processes a single data row or multiple data rows. If you override methods within a control to perform custom aggregation, set this property to SummaryControl to ensure that the control processes all the data rows. If you set this property to PerRowControl, and the control has value expression properties that contain aggregate functions, those aggregate functions will not be evaluated correctly. AcControlValueType values are listed in Table 6-23.

Table 6-23 AcControlValueType values

Constant	Description
AutoValueControl	If any value expression property of the control contains an aggregate function, the control will process multiple data rows. If no value expression property of the control contains an aggregate function, the control will process only one data row.
PerRowControl	The control will process only one data row.
SummaryControl	The control will process multiple data rows.

AcCrosstabBorderStyle

AcCrosstabBorderStyle is a structure that describes the border of a cross tab. AcCrosstabBorderStyle members are listed in Table 6-24.

Table 6-24 AcCrosstabBorderStyle members

Member name	Type	Description
Color	AcColor	The color of the border
Thickness	AcTwips	The thickness of the border

AcCrosstabTotalColumnPlacement

AcCrosstabTotalColumnPlacement is an enum that specifies how a summary column appears, relative to its subgroups. AcCrosstabTotalColumnPlacement values are listed in Table 6-25.

Table 6-25 AcCrosstabTotalColumnPlacement values

Constant	Description
NoTotalColumn	Does not display the summary column in the cross tab
TotalColumnLeft	Displays the summary column to the left of its subgroups in the cross tab
TotalColumnRight	Displays the summary column to the right of its subgroups in the cross tab

AcCrosstabTotalRowPlacement

AcCrosstabTotalRowPlacement is an enum that specifies how a summary row appears, relative to its subgroups AcCrosstabTotalRowPlacement values are listed in Table 6-26.

Table 6-26 AcCrosstabTotalRowPlacement values

Constant	Description
NoTotalRow	Does not display the summary row
TotalRowAbove	Displays the summary row above its subgroups
TotalRowBelow	Displays the summary row below its subgroups

AcCrosstabValueLayout

AcCrosstabValueLayout is an enum. In cross-tab cells that contain more than one value, AcCrosstabValueLayout determines whether the values appear side by side or one above the other. AcCrosstabValueLayout values are listed in Table 6-27.

Table 6-27 AcCrosstabValueLayout values

Constant	Description
ValuesHorizontal	In a cross-tab cell that contains more than one value, displays values side by side

Table 6-27 AcCrosstabValueLayout values

Constant	Description
ValuesVertical	In a cross-tab cell that contains more than one value, displays values in a vertical stack

AcDataGroupingMode

AcDataGroupingMode is an enum that specifies how to group data from multiple data rows in a chart or cross tab. Valid values are listed in Table 6-28.

Table 6-28 AcDataGroupingMode values

Constant	Description
DataGroupingMode Interval	Groups data into a series of ranges of equal sizes based on a key value in each data row. Values for all data rows whose key values fall into a single range are aggregated. For example, the sum of daily stock trade volumes grouped by calendar month.
DataGroupingMode None	Does not group data. No aggregation is performed.
DataGroupingMode Ranges	Groups data into a series of explicitly specified ranges that might be of different sizes.
DataGroupingMode UniqueKey	Groups data based on a key value in each data row. Values for all data rows that have the same key value are aggregated. For example, a count of customers grouped by credit rank.

AcDataGroupingUnit

AcDataGroupingUnit is an enum that specifies a range unit to use to group data from multiple data rows in a chart or cross tab. AcDataGroupingUnit values are listed in Table 6-29.

Table 6-29 AcDataGroupingUnit values

Constant	Description
DataGroupingUnit Day	The key values used to group data are date and time values that are truncated to days.

(continues)

Table 6-29 AcDataGroupingUnit values (continued)

Constant	Description
DataGroupingUnit Half	The key values used to group data are date and time values that are truncated to halves. For example, data rows whose key values are 2003-01-01 and 2003-06-30 are grouped together.
DataGroupingUnit Hour	The key values used to group data are date and time values that are truncated to hours.
DataGroupingUnit Integer	The key values used to group data are numbers that are truncated to integers. For example, data rows whose key values are 1.0 and 1.9 are grouped together.
DataGroupingUnit Minute	The key values used to group data are date and time values that are truncated to minutes. For example, data rows whose key values are 07:55:00 and 07:55:59 are grouped together.
DataGroupingUnit Month	The key values used to group data are date and time values that are truncated to months.
DataGroupingUnit None	Does not group data. No aggregation is performed.
DataGroupingUnit Quarter	The key values used to group data are date and time values that are truncated to quarters. For example, data rows whose key values are 2003-01-01 and 2003-03-31 are grouped together.
DataGroupingUnit Second	The key values used to group data are date and time values that are truncated to seconds.
DataGroupingUnit Week	The key values used to group data are date and time values that are truncated to weeks. By default, a week is Sunday through Saturday but this setting can be configured elsewhere.
DataGroupingUnit Year	The key values used to group data are date and time values that are truncated to years.

AcDataType

AcDataType is an enum that specifies the format of a cross-tab row, column, or cell value. AcDataType values are listed in Table 6-30.

Table 6-30 AcDataType values

Constant	Description
DataTypeAutomatic	The value is set automatically.
DataTypeText	The value is text.
DataTypeNumber	The value is a number.
DataTypeDateTime	The value is in the date and time format.

AcDay

AcDay is an enum that specifies a day of the week. AcDay values are listed in Table 6-31.

Table 6-31 AcDay values

Constant	Description
Sunday	Sunday
Monday	Monday
Tuesday	Tuesday
Wednesday	Wednesday
Thursday	Thursday
Friday	Friday
Saturday	Saturday

AcDrawingBorderStyle

AcDrawingBorderStyle is a structure that specifies the style of the border around an element of a drawing. AcDrawingBorderStyle members are listed in Table 6-32.

Table 6-32 AcDrawingBorderStyle members

Member name	Type	Description
Color	AcColor	The color of the border
Shadow	Boolean	True if the border is drawn with a shadow effect
Pen	AcDrawing LinePen	The pattern of the border
Width	AcTwips	The width of the border, in twips

See also AcColor
AcDrawingLinePen
AcTwips

AcDrawingFillPattern

AcDrawingFillPattern is an enum that specifies the pattern to use for a filled area in a drawing. AcDrawingFillPattern values for gradient patterns are listed in Table 6-33.

Table 6-33 AcDrawingFillPattern gradient values

Constant	Description
DrawingFillGradientCenter	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The color gradient is at the center of the filled area.
DrawingFillGradientCenter Diagonal	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The color gradient is at the center of the filled area on the diagonal.
DrawingFillGradientCorner BottomLeft	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The color gradient is at the lower left corner of the filled area.
DrawingFillGradientCorner BottomRight	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The color gradient is at the lower right corner of the filled area.
DrawingFillGradientCorner TopLeft	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The color gradient is at the upper left corner of the filled area.
DrawingFillGradientCorner TopRight	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The color gradient is at the upper right corner of the filled area.
DrawingFillGradientDiagonal Down	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. Color1 starts at the upper left of the filled area and transitions to Color2 at the lower right on the diagonal.
DrawingFillGradientDiagonal DownMiddle	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. Color1 starts in the middle of the filled area and transitions to Color2 on the diagonal.

Table 6-33 AcDrawingFillPattern gradient values

Constant	Description
DrawingFillGradientDiagonalUp	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. Color1 starts in the lower left of the filled area and transitions to Color2 at the upper right on the diagonal.
DrawingFillGradientDiagonalUp Middle	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. Color1 starts in the lower left of the filled area and transitions to Color2 in the upper right on the diagonal.
DrawingFillGradientHorizontal	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The colors are displayed horizontally across the filled area.
DrawingFillGradientHorizontal Middle	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The colors are displayed horizontally across the middle of the filled area.
DrawingFillGradientVertical	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The colors are displayed vertically across the filled area.
DrawingFillGradientVertical Middle	Displays a gradient between the Color1 and Color2 values specified in AcDrawingFillStyle. The colors are displayed vertically across the middle of the filled area.

AcDrawingFillPattern values for solid colors are listed in Table 6-34.

Table 6-34 AcDrawingFillPattern solid color values

Constant	Description
DrawingFillPatternNone	The area is transparent.
DrawingFillPattern05Percent	A finely shaded pattern in which the foreground color makes up 5% of the area.
DrawingFillPattern10Percent	A finely shaded pattern in which the foreground color makes up 10% of the area.
DrawingFillPattern20Percent	A finely shaded pattern in which the foreground color makes up 20% of the area.
DrawingFillPattern25Percent	A finely shaded pattern in which the foreground color makes up 25% of the area.
DrawingFillPattern30Percent	A finely shaded pattern in which the foreground color makes up 30% of the area.

(continues)

Table 6-34 AcDrawingFillPattern solid color values (continued)

Constant	Description
DrawingFillPattern40Percent	A finely shaded pattern in which the foreground color makes up 40% of the area.
DrawingFillPattern50Percent	A finely shaded pattern in which the foreground color makes up 50% of the area.
DrawingFillPattern60Percent	A finely shaded pattern in which the foreground color makes up 60% of the area.
DrawingFillPattern70Percent	A finely shaded pattern in which the foreground color makes up 70% of the area.
DrawingFillPattern75Percent	A finely shaded pattern in which the foreground color makes up 75% of the area.
DrawingFillPattern80Percent	A finely shaded pattern in which the foreground color makes up 80% of the area.
DrawingFillPattern90Percent	A finely shaded pattern in which the foreground color makes up 90% of the area.
DrawingFillPatternSolid	The area is filled with the background color.

AcDrawingFillPattern values for line patterns are listed in Table 6-35.

Table 6-35 AcDrawingFillPattern line values

Constant	Description
DrawingFillPatternDiagonalDown Dark	Heavy solid diagonal lines that run from top left to bottom right
DrawingFillPatternDiagonalDown Dash	Light dashed diagonal lines that run from top left to bottom right
DrawingFillPatternDiagonalDown Light	Light solid diagonal lines that run from top left to bottom right
DrawingFillPatternDiagonalDown Wide	Wide solid diagonal lines that run from top left to bottom right
DrawingFillPatternDiagonalUp Dark	Heavy solid diagonal lines that run from bottom left to top right
DrawingFillPatternDiagonalUp Dash	Light dashed diagonal lines that run from bottom left to top right
DrawingFillPatternDiagonalUp Light	Light solid diagonal lines that run from bottom left to top right
DrawingFillPatternDiagonalUp Wide	Wide solid diagonal lines that run from bottom left to top right

Table 6-35 AcDrawingFillPattern line values

Constant	Description
DrawingFillPatternHorizontalDark	Heavy solid horizontal lines
DrawingFillPatternHorizontalDash	Light dashed horizontal lines
DrawingFillPatternHorizontalLight	Light solid horizontal lines
DrawingFillPatternHorizontalNarrow	Narrow solid horizontal lines
DrawingFillPatternVerticalDark	Heavy solid vertical lines
DrawingFillPatternVerticalDash	Light dashed vertical lines
DrawingFillPatternVerticalLight	Light solid vertical lines
DrawingFillPatternVerticalNarrow	Narrow solid vertical lines

AcDrawingFillPattern values for decorative patterns are listed in Table 6-36.

Table 6-36 AcDrawingFillPattern decorative values

Constant	Description
DrawingFillPatternBrickHorizontal	A pattern that resembles a brick wall
DrawingFillPatternBrickDiagonalUp	A pattern that resembles a brick wall rotated 45 degrees counterclockwise
DrawingFillPatternCheckerBoardLarge	A checkerboard with large squares
DrawingFillPatternCheckerBoardSmall	A checkerboard with small squares
DrawingFillPatternConfettiLarge	A pattern that resembles a shower of large confetti
DrawingFillPatternConfettiSmall	A pattern that resembles a shower of small confetti
DrawingFillPatternDiamondDotted	A coarse diagonal grid of dotted lines
DrawingFillPatternDiamond	A coarse diagonal grid of solid lines
DrawingFillPatternDiamondSolid	A checkerboard with large squares rotated 45 degrees
DrawingFillPatternDivot	Alternating rows of < and > symbols
DrawingFillPatternGridDotted	A coarse grid of dotted lines

(continues)

Table 6-36 AcDrawingFillPattern decorative values (continued)

Constant	Description
DrawingFillPatternGridLarge	A coarse grid of solid lines
DrawingFillPatternGridSmall	A fine grid of solid lines
DrawingFillPatternPlaid	A coarse checkered pattern of shed horizontal bands and solid vertical bands
DrawingFillPatternShingle	A pattern that resembles a shingled roof
DrawingFillPatternSphere	A pattern of spheres with a three-dimensional appearance
DrawingFillPatternTrellis	A pattern that resembles a trellis
DrawingFillPatternWave	A pattern of light dashed horizontal wavy lines
DrawingFillPatternWeave	A diagonal pattern of interwoven dotted lines
DrawingFillPatternZigzag	A pattern of light solid horizontal wavy lines

AcDrawingFillStyle

AcDrawingFillStyle is a structure that specifies the style of a filled area in a drawing. AcDrawingFillStyle members are listed in Table 6-37.

Table 6-37 AcDrawingFillStyle members

Member name	Type	Description
Color1	AcColor	Ignores this color if the value of the Pattern member is DrawingFillPatternNone. Fills the area with this color if the value of the Pattern member is DrawingFillPatternSolid. This color is the background color of the pattern if the value of the Pattern member is a pattern. This color is the start color of the gradient if the value of the Pattern member is a gradient.
Color2	AcColor	Ignores this color if the value of the Pattern member is DrawingFillPatternNone or DrawingFillPatternSolid. This color is the foreground color of the pattern if the value of the Pattern member is a pattern. This color is the finish color of the gradient if the value of the Pattern member is a gradient.
Pattern	AcDrawing FillPattern	The pattern used to fill the area.

See also AcColor
AcDrawingFillPattern

AcDrawingLinePen

AcDrawingLinePen is an enum that specifies the appearance of a line in a drawing. AcDrawingLinePen values are listed in Table 6-38.

Table 6-38 AcDrawingLinePen values

Constant	Description
DrawingLinePenDash	A line in the following format: ----- ----- ----- -----
DrawingLinePenDashDot	A line in the following format: ----- - ----- - ----- -
DrawingLinePenDashDotDot	A line in the following format: ----- -- -- ----- -- --
DrawingLinePenDot	A line in the following format: -- -- -- -- -- -- -- --
DrawingLinePenNone	No line
DrawingLinePenSolid	A solid line

AcDrawingLineStyle

AcDrawingLineStyle is a structure that specifies the style of a line in a drawing. AcDrawingLineStyle members are listed in Table 6-39.

Table 6-39 AcDrawingLineStyle members

Member name	Type	Description
Color	AcColor	The color of the line
Pen	AcDrawingLinePen	The pattern of the line
Width	AcTwips	The width of the line in twips

See also AcColor
AcDrawingLinePen
AcTwips

AcDrawingTextOrientation

AcDrawingTextOrientation is an enum that specifies the orientation of some text in a drawing. AcDrawingTextOrientation values are listed in Table 6-40.

Table 6-40 AcDrawingTextOrientation values

Constant	Description
DrawingTextOrientationAuto	Determines the angle of the text automatically. For example, the axis labels on a chart rotate automatically if there is not enough space to fit them horizontally.
DrawingTextOrientationCustom	Draws the text at a specified angle.
DrawingTextOrientation Horizontal	Draws the text horizontally.
DrawingTextOrientationVertical	Draws the text vertically with the characters stacked on top of another.

AcDrawingTextStyle

AcDrawingTextStyle is a structure that specifies the style of some text in a drawing. AcDrawingTextStyle members are listed in Table 6-41.

Table 6-41 AcDrawingTextStyle members

Member name	Type	Description
Background Color	AcColor	The background color for the text.
Border	AcDrawing BorderStyle	The style of the border around the text.
CustomAngle	AcAngle	If the value of the Orientation member is DrawingTextOrientationCustom, the angle of the text in degrees counterclockwise from horizontal. Otherwise, the CustomAngle value is ignored. Text angles are rounded to integer values internally.
Font	AcFont	The font of the text.
Orientation	AcDrawingText Orientation	The orientation of the text.

See also AcDrawingBorderStyle
AcDrawingTextOrientation
AcFont

AcExcelBorder

AcExcelBorder is a structure that describes the border. AcExcelBorder members are listed in Table 6-42.

Table 6-42 AcExcelBorder members

Member name	Type	Description
Style	AcExelBorderType	The style of the border
Color	AcColor	The color of the border

AcExcelBorderType

AcExcelBorderType specifies the line style of the border. AcExcelBorderType values are listed in Table 6-43.

Table 6-43 AcExcelBorderType values

Constant	Description
ExcelBorderDashDot	A dash-dot line
ExcelBorderDashDotDot	A dash-dot-dot line
ExcelBorderDashed	A dashed line
ExcelBorderDotted	A dotted line
ExcelBorderDouble	A double line
ExcelBorderHair	A hairline
ExcelBorderMedium	A medium line
ExcelBorderMediumDashDot	A dash-dot line of medium thickness
ExcelBorderMediumDashDotDot	A dash-dot-dot line of medium thickness
ExcelBorderMediumDashed	A dashed-line of medium thickness
ExcelBorderNone	No border
ExcelBorderSlantedDashDot	A slanted dash-dot line
ExcelBorderThick	A thick line
ExcelBorderThin	A thin line

AcExcelHorizontalAlignment

AcExcelHorizontalAlignment specifies the horizontal alignment of data in cells. AcExcelHorizontalAlignment values are listed in Table 6-44.

Table 6-44 AcExcelHorizontalAlignment values

Constant	Description
ExcelHAlignCenter	Centers data in the cell
ExcelHAlignGeneral	The default alignment: ■ Aligns text at the left edge of the cell ■ Aligns numbers, dates, and times at the right edge of the cell ■ Centers logical and error values
ExcelHAlignJustify	Adjusts the spacing between words so that all lines are as wide as the cell
ExcelHAlignLeft	Aligns data at the left edge of the cell
ExcelHAlignRight	Aligns data at the right edge of the cell

AcExcelVerticalAlignment

AcExcelVerticalAlignment specifies the vertical alignment of data in cells. AcExcelVerticalAlignment values are listed in Table 6-45.

Table 6-45 AcExcelVerticalAlignment values

Constant	Description
ExcelVAlignBottom	Aligns data at the bottom of the cell
ExcelVAlignCenter	Aligns data at the center of the cell
ExcelVAlignJustify	Adjusts the spacing between lines so that the spacing is even and the lines fill the cell
ExcelVAlignTop	Aligns data at the top of the cell

AcFlowPlacement

AcFlowPlacement is an enum that specifies how a frame appears within a flow that is wider than the frame. AcFlowPlacement values are listed in Table 6-46.

Table 6-46 AcFlowPlacement values

Constant	Description
FlowAlignLeftOrTop	Aligns a frame to the left of a flow or at the top of the flow
FlowAlignCenter	Aligns a frame in the center of a flow
FlowAlignRightOrBottom	Aligns a frame to the right of a flow or at the bottom of the flow
FlowAlignCustom	Aligns frames in the flow at the position given by the frame's Position.X member

AcFont

AcFont is a structure that describes a font in a device-independent way. AcFont members are listed in Table 6-47.

Table 6-47 AcFont members

Member name	Type	Description
Bold	Boolean	If True, the text is bold
Color	AcColor	Color of the text
FaceName	String	Font name of the text
Italic	Boolean	If True, the text is italic
Script	String	Specifies a subset of a large font
Size	Integer	Size of the text in points
StrikeThrough	Boolean	If True, a line is drawn through the text
Underline	Boolean	If True, the text is underlined

AcGroupOnType

AcGroupOnType defines how to group data in a group section. AcGroupOnType values are listed in Table 6-48.

Table 6-48 AcGroupOnType values

Constant	Description
GroupOnCustom	Group based on key value set in the GetGroupKey method

(continues)

Table 6-48 AcGroupOnType values (continued)

Constant	Description
GroupOnDay	Group data by full date
GroupOnEveryValue	Group on the full key
GroupOnHour	Group data by hour
GroupOnInterval	Group on option for group section keys having data types other than Currency, Date, Double, Integer, Single, or String
GroupOnMinute	Group data by minute
GroupOnMonth	Group data by month
GroupOnPrefix	Group on the first n characters of text
GroupOnQuarter	Group data by calendar quarter
GroupOnWeek	Group data by week
GroupOnYear	Group data by year

AcHorizontalPosition

AcHorizontalPosition is an enum that specifies how to position a visual object horizontally. AcHorizontalPosition values are listed in Table 6-49.

Table 6-49 AcHorizontalPosition values

Constant	Description
HorizontalPositionDefault	If the object's left edge is at or to the right of the horizontal midpoint of the reference object, the object moves to keep the distance between its left edge and the right edge of the reference object constant. Otherwise, the object does not move or resize.
HorizontalPositionFrameCenter	The object moves to keep the distance between its horizontal midpoint and the horizontal midpoint of the frame constant.
HorizontalPositionFrameLeft	The object does not move.
HorizontalPositionFrameRight	The object moves to keep the distance between its right edge and the right edge of the frame constant.
HorizontalPositionLeft	If the object's left edge is to the left of the right edge of the reference object, the object does not move. Otherwise, the object moves to keep the distance between its left edge and the right edge of the reference object constant.

Table 6-49 AcHorizontalPosition values

Constant	Description
HorizontalPositionRight	If the object's left edge is to the left of the reference object's left edge, the object does not move. Otherwise, the object moves to keep the distance between its left edge and the right edge of the reference object constant.

AcHorizontalSize

AcHorizontalSize is an enum that specifies how to resize a visual object horizontally. AcHorizontalSize values are listed in Table 6-50.

Table 6-50 AcHorizontalSize values

Constant	Description
HorizontalSizeFixed	The object is not resized.
HorizontalSizeFrameRelative	The object's width adjusts to keep the distance between its right edge and the right edge of the frame constant.
HorizontalSizeRelative	If the object's left edge is at or to the left of the reference object's left edge and its right edge is at or to the right of the reference object's right edge, the object's width increases by the amount that the reference object's width increases. If more than one dynamic content object exists, the object's width increases in one of the following ways to give the greatest width increase: ■ The distance between the object's right edge and the right edge of the reference object remains constant. ■ The object's width increases by the amount the reference object's width increases. In this case, the object also moves left, if the object's CanMoveLeft property is set to True. The object moves left in one of the following ways, to give the smallest movement: ▪ The distance between the object's right edge and the reference object's right edge remains constant. ▪ The object moves left by the amount its width increased.

AcImageEmbedType

AcImageEmbedType defines when to include the image in the report. AcImageEmbedType values are listed in Table 6-51.

Table 6-51 AcImageEmbedType values

Constant	Description
ImageDesignTime	Include image at compile time.
ImageFactoryTime	Include image when the report builds.
ImageFactoryTimeSingle	Include only a single copy of the image when the report builds.
ImageViewTime	Include image when the report appears.
ImageViewTimeSingle	Include only a single copy of the image when the report appears.

AcLayoutOrientation

AcLayoutOrientation is an enum that defines the orientation for the report. AcLayoutOrientation values are listed in Table 6-52.

Table 6-52 AcLayoutOrientation values

Constant	Description
LeftToRight	The report has left-to-right orientation.
RightToLeft	The report has right-to-left orientation.

AcLinePen

AcLinePen is an enum that specifies the style of line to draw. Note that although these styles mimic the Windows line styles, they are not meant to duplicate the Windows line style values or have a direct numeric mapping to Windows styles. AcLinePen values are listed in Table 6-53.

Table 6-53 AcLinePen values

Constant	Description
DashLine	Draws a dashed line

Table 6-53 AcLinePen values

Constant	Description
DashDotLine	Draws a line in the following format: ----- - ----- - ----- -
DashDotDotLine	Draws a line in the following format: ----- -- -- ----- -- --
DotLine	Draws a dotted line
DoubleLine	Draws a double solid line
InsideFrameBorder	Draws a solid line inside the frame or control
NullLine	Does not draw a line
ShortDotLine	Draws a line using very small dots
SingleLine	Draws a single solid line

AcLineStyle

AcLineStyle is a structure that describes how a line is drawn. AcLineStyle members are listed in Table 6-54.

Table 6-54 AcLineStyle members

Member name	Type	Description
Color	AcColor	The color of the line
Pen	AcLinePen	The style of the line
Width	AcTwips	The width of the line in twips

AcMargins

AcMargins is a structure that describes the margins of a textual control. AcMargins members are listed in Table 6-55.

Table 6-55 AcMargins members

Member name	Type	Description
Bottom	AcTwips	The top margin
Left	AcTwips	The left margin
Right	AcTwips	The right margin
Top	AcTwips	The top margin

AcMonth

AcMonth is an enum that specifies a month of the year. AcMonth values are listed in Table 6-56.

Table 6-56 AcMonth values

Constant	Description
January	First month of the year
February	Second month of the year
March	Third month of the year
April	Fourth month of the year
May	Fifth month of the year
June	Sixth month of the year
July	Seventh month of the year
August	Eighth month of the year
September	Ninth month of the year
October	Tenth month of the year
November	Eleventh month of the year
December	Twelfth month of the year

AcPageHeaderOptions

AcPageHeaderOptions is an enum that determines whether and where to place a page header. AcPageHeaderOptions values are listed in Table 6-57.

Table 6-57 AcPageHeaderOptions values

Constant	Description
AsColumnHeader	Places the header above data columns
AsPageHeader	Places the header on every page
NoHeaderOnFirst	Places the header on every page except the first page

AcPageNumberStyle

AcPageNumberStyle is an enum that determines how to calculate and display page numbers. AcPageNumberStyle values are listed in Table 6-58.

Table 6-58 AcPageNumberStyle values

Constant	Description
ActualPageCount	The total number of pages (visible and invisible to the user) in the report.
ActualPageN	The actual page number, regardless of how many pages are visible to the user.
ActualPageNofM	The current page number relative to the total pages in the report displayed in the form: Page N of M. Includes both visible and invisible pages.
ActualPageNumber	The current page number considering all the pages (both visible and invisible to the user) in the report.
FormattedPageNumber	Page number is presented using the format string specified in the PageNumberFormat property. The value presented here does not consider page security.
VisiblePageCount	The total number of pages in the report that the user can see considering page security.
VisiblePageN	The number of the current page, based on the total number of pages visible to the user.
VisiblePageNofM	The current page number relative to the total pages in the report displayed in the form: Page N of M. Considers page security.
VisiblePageNumber	The current page number in the report that the user can see considering page security.

AcPercentage

AcPercentage is a Double data type used to hold percentage values.

Percentage values are represented as fractions internally, so that 50% is stored as 0.5. This makes calculations easier, because you can simply multiply a number by an AcPercentage value with no need to scale the result by a factor of 100.

e.Report Designer Professional's property sheet displays AcPercentage property values multiplied by 100 and with a trailing percentage sign. For example, a value of 0.75 will be displayed as 75%.

AcPoint

AcPoint is a structure that defines a position. AcPoint members are listed in Table 6-59.

Table 6-59 AcPoint members

Member name	Type	Description
X	AcTwips	The horizontal coordinate of the position
Y	AcTwips	The vertical coordinate of the position

AcRectangle

AcRectangle is a structure that describes a rectangle relative to the origin of an enclosing rectangle by giving the bounding points, the corners, of the rectangle. AcRectangle members are listed in Table 6-60.

Table 6-60 AcRectangle members

Member name	Type	Description
Bottom	AcTwips	The location of the bottom of the rectangle measured relative to the top of the enclosing rectangle
Left	AcTwips	The location of the left of the rectangle measured relative to the left of the enclosing rectangle
Right	AcTwips	The location of the right of the rectangle measured relative to the left of the enclosing rectangle
Top	AcTwips	The location of the top of the rectangle measured relative to the top of the enclosing rectangle

AcSearchType

AcSearchType is an enum that determines whether users can search for the component using values for the DataValue property. AcSearchType values are listed in Table 6-61.

Table 6-61 AcSearchType values

Constant	Description
NotSearchable	User cannot search for the component.
SearchableNo Index	User can search for the component. The client viewing software searches the entire report.
SearchableWith Index	User can search for the component using a high-performance indexed search.

AcSize

AcSize is a structure that describes the width and height of a rectangle. AcSize members are listed in Table 6-62.

Table 6-62 AcSize members

Member name	Type	Description
Height	AcTwips	The height of the rectangle
Width	AcTwips	The width of the rectangle

AcSortingOptions

AcSortingOptions is an enum that determines the sorting rules for a report section. AcSortingOptions values are listed in Table 6-63.

Table 6-63 AcSortingOptions values

Constant	Description
AutoSort	e.Report Designer Professional sorts the data rows according to the groups in the report section. If the data source uses a SQL query, sorting specified in the ORDER BY clause is applied after the automatic sorting that AutoSort applies.
CompatibleSort	This constant provides backward compatibility for reports converted from an Actuate release earlier than 3.1.
PreSorted	e.Report Designer Professional does not sort the data unless the report developer codes a sort filter. Data rows appear in the report in the same order in which they appear in the data source. Typically, this constant is useful when the SQL query has an ORDER BY clause or when the data source provides the rows in the order in which you want them to appear in the report.

AcTextClipStyle

AcTextClipStyle is an enum that specifies how to handle text that is too long for its enclosing rectangle. Leading truncation removes the first part of the string, while trailing truncation removes the end part. There is also an option to use overflow characters to show truncation. AcTextClipStyle applies only to single-line controls. AcTextClipStyle values are listed in Table 6-64.

Table 6-64 AcTextClipStyle values

Constant	Description
ClipLeading	Clips the leftmost characters of the text. Displays an ellipsis (…) before the truncated text, when the Ellipsis property of AcTextPlacement is set to True.
ClipTrailing	Clips the rightmost characters of the text. Displays an ellipsis (…) after the truncated text, when the Ellipsis property of AcTextPlacement is set to True.
ShowOverflowChar	Displays overflow characters (*) when text is too long to display.

AcTextFormat

AcTextFormat is an enum that indicates the tagging format of text. AcTextFormat values are listed in Table 6-65.

Table 6-65 AcTextFormat values

Constant	Description
TextFormatHTML	The text contains HTML tags.
TextFormatPlain	The text is not tagged.
TextFormatRTF	The text contains RTF tags.

AcTextJustify

AcTextJustify is an enum that specifies how to align text. AcTextJustify values are listed in Table 6-66.

Table 6-66 AcTextJustify values

Constant	Description
TextAlignCenter	Aligns text in the center of the control
TextAlignLeft	Aligns text at the left of the control
TextAlignRight	Aligns text at the right of the control

AcTextPlacement

AcTextPlacement is a structure that describes the placement of text in a frame or control. AcTextPlacement members are listed in Table 6-67.

Table 6-67 AcTextPlacement members

Member name	Type	Description
Clip	AcTextClipStyle	Specifies how to clip text that is too large to fit into the control. Applies only to single-line controls.
Ellipsis	Boolean	If set to True, places an ellipsis after the text within the control if the text is too long to fit. Applies only to single-line controls.
FillPattern	String	Specifies the fill pattern to use for any space after the text within the control.
Horizontal	AcTextJustify	Specifies horizontal text placement and justification.
MultiLine	Boolean	Specifies whether the control can contain more than one line of text.
Vertical	AcTextVerticalPlacement	Determines vertical text placement. Applies only to single-line controls.
WordWrap	AcWordWrapStyle	Specifies how to split text that is too long to fit onto a single line.

AcTextVerticalPlacement

AcTextVerticalPlacement is an enum that specifies how single lines of text align vertically within the enclosing rectangle. AcTextVerticalPlacement values are listed in Table 6-68.

Table 6-68 AcTextVerticalPlacement values

Constant	Description
TextAlignBottom	Aligns text at the bottom of a control
TextAlignMiddle	Aligns text in the vertical middle of a control
TextAlignTop	Aligns text at the top of a control

AcTOCNodeType

AcTOCNodeType is an enum that determines whether a component appears in a report's table of contents. AcTOCNodeType values are listed in Table 6-69.

Table 6-69 AcTOCNodeType values

Constant	Description
TOCAlwaysAdd	Always add the component to the table of contents.
TOCIfAllVisible	Add the component to the table of contents only if the user can view at least one page generated from the component based on page-level security.
TOCIfAnyVisible	Add the component to the table of contents even if the user cannot view any of the pages generated from the component based on page-level security.
TOCSkip	Never add the component to the table of contents.

AcTwips

AcTwips is an Integer data type used to hold values in the internal unit of measurement of the AFC framework, the twip. A twip is 1/20 of an integer point, or 1/1440 of an inch.

e.Report Designer Professional's property sheet displays AcTwips property values converted to the default unit of measurement, and with a suffix indicating the unit of measurement. For example, if the default unit of measurement is points, a value of 1440 twips is displayed as 72pt.

If you type an AcTwips property value with a suffix indicating a unit of measurement, e.Report Designer Professional automatically converts that value to the default unit of measurement. For example, if the default unit of measurement is points, and you type 1cm, e.Report Designer Professional converts the value to points and displays 28.35pt.

If you enter an AcTwips property value with no unit of measurement suffix, e.Report Designer Professional uses the default unit of measurement. For example, if the default unit of measurement is points, and you type 36, e.Report Designer Professional displays 36pt.

AFC defines a set of constants that you can use to convert AcTwips values to and from other units. AcTwips conversion constants are listed in Table 6-70.

Table 6-70 AcTwips conversion constants

Constant	Value	Description
OneCM	567	The number of twips in one centimeter
OneInch	1440	The number of twips in one inch
OneMM	57	The number of twips in one millimeter
OnePoint	20	The number of twips in one point

Example The following example creates a label control and sets its height to 14 points:

```
Dim l As AcLabelControl
Set l = New Persistent AcLabelControl
l.Size.Height = 14 * OnePoint
```

AcVerticalPosition

AcVerticalPosition is an enum that specifies how to position a visual object vertically. AcVerticalPosition values are listed in Table 6-71.

Table 6-71 AcVerticalPosition values

Constant	Description
VerticalPositionBottom	If the top of the object is above the top of the reference object, it does not move. Otherwise, the object moves to keep the distance between its bottom edge and the bottom of the reference object constant.
VerticalPositionDefault	If the top of the object is at or below the midpoint of the reference object, the behavior is the same as VerticalPositionBottom. Otherwise, the object does not move.
VerticalPositionFrame Bottom	The object moves to keep the distance between its bottom edge and the bottom of the frame constant.
VerticalPositionFrame Middle	The object moves to keep the distance between its middle and the middle of the frame constant.
VerticalPositionFrameTop	The object does not move.
VerticalPositionTop	If the top of the object is above the bottom of the reference object, the object does not move. Otherwise, the object repositions to keep the distance between its top and the bottom of the reference object constant.

AcVerticalSize

AcVerticalSize is an enum that specifies how to resize a visual object vertically. AcVerticalSize values are listed in Table 6-72.

Table 6-72 AcVerticalSize values

Constant	Description
VerticalSizeFixed	The object does not resize.
VerticalSizeFrameRelative	The object resizes to keep the distance between its bottom edge and the bottom of the frame constant.
VerticalSizeRelative	If the top of the object is at or above the top of the reference object and its bottom edge is at or below the bottom of the reference object, the object's height increases by the amount that the reference object's width increases. If more than one dynamic content object exists, the object increases in one of the following ways, to give the greatest height increase: ■ The distance between the object's bottom edge and the bottom of the reference object remains constant. ■ The object height increases by the same amount as the reference object's height increase. In this case, the object also moves up, if the object's CanMoveUp property is set to True. The object moves up in one of the following ways, to give the smallest movement: ▪ The distance between the object's bottom edge and the reference object's bottom edge remains constant. ▪ The object moves up by the amount its height increases. ▪ If the top of the object is below the top of the reference object or its bottom edge is above the bottom of the reference object, the object moves according to the setting of its VerticalPosition property.

AcWordWrapStyle

AcWordWrapStyle is an enum that specifies the actions for lines in a multi-line control when a line is longer than the size of the control. AcWordWrapStyle values are listed in Table 6-73.

Table 6-73 AcWordWrapStyle values

Constant	Description
TextCharacterWrap	Wraps text from one line to the next breaking the text at a character boundary
TextTruncateLines	Truncates any lines that do not fit
TextWordWrap	Wraps text from one line to the next breaking the text at a word boundary

AcXMLType

AcXMLType is an enum that specifies the type of XML to create for the component. AcXMLType values are listed in Table 6-74.

Table 6-74 AcXMLType values

Constant	Description
XMLAttribute	Converts component to an XML attribute
XMLCustom	Custom XML to be generated by AcXMLDataVisitor class functions
XMLElement	Converts component to an XML element
XMLEmptyElement	Converts component to an empty XML element
XMLIgnore	Does not convert the component into XML
XMLText	Converts component into XML text

Chapter

7

AFC classes

This chapter provides an alphabetical listing of the Actuate Foundation Classes. Each class entry includes a general description of the class and a summary of its variables, properties, and methods followed by an alphabetical listing of methods for that class.

For the most part, the class documentation does not include repeated descriptions of inherited variables, properties, and methods. For example, OnRow( ) is described only in the AcReportComponent base class. A method is described in a subclass as well as a superclass if the implementation details are significantly different or enhanced in the subclass. For example, BuildFromRow( ) is described in several class entries, including AcReportComponent, AcBaseFrame, and AcChart, because its implementation varies from class to class.

Class AcBaseFrame

An abstract base class that defines the core logic that is common to pages and frames. Figure 7-1 shows the class hierarchy for AcBaseFrame.

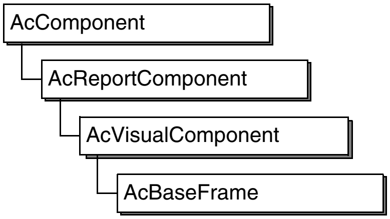


Figure 7-1 AcBaseFrame

Description AcBaseFrame is the abstract base class for pages and frames. It provides the core logic for creating and working with contents in a page or a frame. The contents of a page can include flows and controls. The contents of a frame can include other frames and controls. AcBaseFrame defines

- Methods for accessing the contents of a frame or page. The FindContentByClass( ), FindContentByClassID( ), and GetControl( ) methods locate the specified content in a frame or a page, and the GetControlValue( ) method returns the value of a data control in a frame.
- Methods for adjusting the contents of frames. The SplitContents( ) and SplitFrame( ) methods distribute the contents of frames across pages.
- Page-specific methods, such as GetPageNumber( ), that are not applicable to frames. If you call these methods in a frame, the framework displays an error message.

Class protocol

Table 7-1 describes the class protocol for AcBaseFrame.

Table 7-1 Class protocol for AcBaseFrame

Method	Task
Start( )	Instantiates and starts the contents of the frame or the page
Build( )	Builds the contents for frames that are not dependent on data
BuildFromRow( )	Populates the contained frames, charts, and controls with data
Finish( )	Finishes each of the content objects

Preparing the frame or page

The framework instantiates the contents of a frame or page using Start( ). Start( ), in turn, calls MakeContents( ) to instantiate each of the contents in the order in which they were added in the design perspective.

Start( ) is part of the framework's core protocol. Override Start( ) in the AcFrame class to perform custom processing in the frame that is unrelated to its contents. For example, you can conditionally change the background color of the frame. Always call Super::Start( ) before making your own programming changes to Start( ).

Building the frame or page

The Build( ) method of the frame's container calls the frame's Build( ) method, instead of BuildFromRow( ), when you place the frame in a slot where the frame does not receive data rows. The following list includes some of the situations in which the frame does not receive data rows:

- The frame is in a sequential or conditional section that is directly within the report component.
- The frame is placed directly on a page.
- The frame is nested within any of the frames described in the previous bullets.

The frame's Build( ) method, in turn, calls the Build( ) method for each of the controls in the frame.

You can override the Build( ) method of a frame or page to perform custom processing, such as conditionally adding or deleting frame or page contents, or setting the values or properties of the contents.

Subclassing AcBaseFrame

Because AcBaseFrame is an abstract base class, do not derive directly from it.

Variables

AcBaseFrame variables are listed in Table 7-2.

Table 7-2 AcBaseFrame variables

Variable	Type	Description
BackgroundColor	AcColor	The color with which to fill the frame before displaying the frame's contents. The default value is Transparent.
Border	AcLineStyle	The style, thickness, and color of the border. The default value is no border.

Properties

AcBaseFrame properties are listed in Table 7-3.

Table 7-3 AcBaseFrame properties

Property	Type	Description
BackgroundColor	AcColor	The color with which to fill the frame before displaying the frame's contents. The default value is Transparent.
Border	AcLineStyle	The style, thickness, and color of the border. The default value is no border.

Example The following example shows how to change the background color of a flow on the first page to teal. The flows on the other pages use the color set at design time. The example overrides Build( ) in PageStyle to change the color of the flow. The call to GetPageIndex( ) identifies the first page.

```
Sub Build( )
' Find the flow and change its background color to teal
' only on the first page.
   Super::Build( )
   Dim flow As AcFlow
   Set flow = FindContentByClass( "Flow" )
   If GetPageIndex( ) = 1 Then
      flow.BackgroundColor = teal
   End If
End Sub
```

Methods for Class AcBaseFrame

Methods defined in Class AcBaseFrame

AddToAdjustSizeList, BindToFlow, FindContentByClassID, GetControl, GetControlValue, GetPageNumber, GetSearchValue, IsDataFrame, IsFooter, IsHeader, MakeContents, RebindToFlow, SearchAttributeName

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcBaseFrame::AddToAdjustSizeList method

Adds a component to its container's list of components to resize and causes a call to the component's AdjustSize( ) method. To make a component resizable, you must add a call to the AddToAdjustSizeList( ) method of the component's Start( ) method.

Syntax Sub AddToAdjustSizeList(component As AcVisualComponent)

Parameter **component**
The component to add to the list.

AcBaseFrame::AdjustSize method

Adjusts the size of the frame. Using this method, you can also adjust the sizes and positions of the frame's contents in response to the frame's size adjustment.

Syntax Sub AdjustSize( )

Example The following code calls AdjustSizeList( ) to get the frames to resize. After completing the size adjustments, this code sets AdjustSizeList( ) to Nothing.

```
Sub AdjustSize( )
   If Not AdjustSizeList Is Nothing Then
      Dim i As Integer
      Dim content As AcVisualComponent
```

```
         ' Adjust the sizes of components that requested size
         ' adjustment.
         For i = 1 To AdjustSizeList.Count
         set content = AdjustSizeList.Contents(i)
            content.AdjustSize( )
         Next i
         ' Adjust the geometry of the frame's contents. Get the
         ' amount of the frame's size change as a result.
         Dim deltaFrameWidth  As AcTwips
         Dim deltaFrameHeight As AcTwips
         AdjustContentGeometry( deltaFrameWidth, deltaFrameHeight )

         ' Resize the frame.
         AdjustSizeBy( deltaFrameWidth, deltaFrameHeight )

         ' Adjust the contents of all contents in response to the
         ' frame’s adjustment.
         For i = 1 To FrameContents.Count
            set content = FrameContents.Contents(i)
            content.AdjustContents( )
         Next i

         ' Recover memory when size adjustments are complete.
         Set AdjustSizeList = Nothing
   End If
End Sub
```

AcBaseFrame::BindToFlow method

Called when the framework places the frame into a flow on the page. You can override this method to perform actions based on where the frame is in a flow. For example, the flow can keep track of an alternating color for its frames and the frame can ask the flow for the correct color.

If you override this method, you must call Super::BindToFlow( ).

Syntax Sub BindToFlow(flow As AcFlow)

Parameter **flow**
The flow that contains the component.

Example The following example shows how to override BindToFlow( ) to assign a different color to every alternate frame in a flow. The code example assumes the following variables are declared for the frame: AlternateColor, AlternateLines, MostRecentContainer, MostRecentFlow, and RowNumber.

```
Sub BindToFlow( flow As AcFlow )
   Super::BindToFlow( flow )
   ' If the current flow is different from the
   ' previous flow, then reset the color.
   If Not MostRecentFlow Is flow Then
      RowNumber = 0
      Set MostRecentFlow = flow
   End If

   ' If the container is different, this frame is for
   ' a different group than the previous one; therefore,
   ' reset the color.
   If Not MostRecentContainer Is Container Then
      RowNumber = 0
      Set MostRecentContainer = Container
   End If
   ' Choose one of the two colors. AlternateLines and
   ' AlternateColor are user-defined properties that can be set
   ' in the Component Editor
   RowNumber = RowNumber + 1
   If RowNumber > AlternateLines * 2 Then
      RowNumber = 1
   End If
   If RowNumber > AlternateLines Then
      BackgroundColor = AlternateColor
   End If
End Sub
```

AcBaseFrame::FindContentByClassID method

You can uniquely identify a component within a frame or page using the component's class ID. Accessing the component using its class ID is faster than accessing the component by class name. Use the Actuate Basic function GetClassID to identify the class ID for the component. For more information about GetClassID, see *Programming with Actuate Basic.*

Syntax Function FindContentByClassID(classID As Integer) As AcVisualComponent

Parameter **classID**
The integer class ID of the component to find.

Returns A reference to the component if found.
Nothing if the component was not found.

See also AcReportComponent::FindContentByClass method
AcBaseFrame::GetControl method

AcBaseFrame::GetControl method

Use GetControl( ) to obtain a reference to a control in a frame. You specify the control by using either the last part of the control's name, such as PriceControl, or its fully qualified name, such as OrdersReport::ItemFrame::PriceControl.

GetControlValue( ) finds the control, then calls the control's GetValue( ) method to obtain the value.

Syntax Function GetControl(controlName As String) As AcControl

Parameter **controlName**
The name of the control.

Returns A reference to the control if the control exists.
Nothing if the control does not exist.

See also AcReportComponent::FindContentByClass method
AcBaseFrame::FindContentByClassID method

AcBaseFrame::GetControlValue method

Returns the value of a specified data control within the frame. You specify the control by using either the last part of the control's name, such as PriceControl, or its fully qualified name, such as OrdersReport::ItemFrame::PriceControl. GetControlValue( ) finds the control, then calls the control's GetValue( ) method to obtain the value.

If you call this method from BuildFromRow( ), you must consider the order in which controls are built. Generally, the controls of a frame are built in the same order that they appear in Report Structure. If you call GetControlValue( ) to get the value of a control that is not yet created, GetControlValue( ) returns Null.

Syntax Function GetControlValue(controlName As String) As Variant

Parameter **controlName**
The name of the control for which you want the value.

Returns The value of the control if the control exists.
Null if the control does not exist.

See also AcControl::GetControlValue method

AcBaseFrame::GetPageNumber method

Returns the page number. The page number is a formatted string that represents the page number as it appears in the generated report. The page number can be the same as the page index or it can be different from the page index.

Call GetPageNumber( ) from a page only, not from a frame.

Syntax Function GetPageNumber( ) As String

Returns The page number of a page.

AcBaseFrame::GetSearchValue method

Differentiates between subclasses of a parent class when a user is searching for values, activating a hyperlink, or generating reportlet content from a report.

Syntax Function GetSearchValue( ) As String

AcBaseFrame::IsDataFrame method

Indicates whether the frame is a data frame. A data frame contains data components, such as a text control, integer control, or chart.

Syntax Function IsDataFrame( ) As Boolean

Returns True if the frame contains data components.
False if the frame contains only labels, images, or other non-data components.

AcBaseFrame::IsFooter method

Indicates whether the frame is a PageFooter component.

Syntax Function IsFooter( ) As Boolean

Returns True if the frame is a footer.
False if the frame is not a footer.

AcBaseFrame::IsHeader method

Indicates whether the frame is a PageHeader component.

Syntax Function IsHeader( ) As Boolean

Returns True if the frame is a header.
False if the frame is not a header.

AcBaseFrame::MakeContents method

Creates the frame contents dynamically when specific conditions are present.

Syntax Sub MakeContents( )

AcBaseFrame::RebindToFlow method

The framework calls this method for controls that appear within a subpage when the BalanceFlows( ) property of the subpage is set to True. RebindToFlow( ) informs the control that the flow that contains the control changed as a result of the rebalancing. If you override this method, you must call the superclass implementation.

Syntax Sub RebindToFlow(flow As AcFlow)

Parameter **flow**
The flow that contains the component.

AcBaseFrame::SearchAttributeName method

Search value for reportlets. Returns the name of the attribute that a reportlet uses to find a frame. By default, the attribute is SearchValue. If you override this method, you must also override GetSearchValue( ) to return the appropriate value.

Syntax Function SearchAttributeName( ) As String

Returns The name of the attribute that a reportlet uses to find a frame.

Class AcBasePage

An abstract base class that defines the logic for instantiating the contents of pages. Figure 7-2 shows the class hierarchy of AcBasePage.

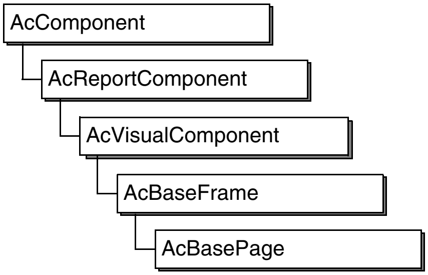


Figure 7-2 AcBasePage

Description AcBasePage is the abstract base class for the two types of page components in a report design, AcPage and AcSubPage. AcPage describes the physical attributes of a page, such as size and page numbering. AcSubpage supports placing a subpage within a page. A subpage exists in a flow and adds a set of flows within a page. For example, you can use a subpage to combine a one-column flow with a two-column flow on a single page.

Subclassing AcBasePage

Because AcBasePage is an abstract base class, do not derive directly from it.

See also Class AcPage

Properties

AcBasePage properties are listed in Table 7-4.

Table 7-4 AcBasePage properties

Property	Type	Description
BalanceFlows	Boolean function	Specifies whether to redistribute the contents of the page to make all flows on the page the same height. The default value is False.
CanIncrease Width	Boolean function	Specifies whether the page width can increase. The default value is False.

Methods for Class AcBasePage

Methods defined in Class AcBasePage

BalanceFlows, GetFirstDataFrame, GetLastDataFrame

Methods inherited from ClassAcBaseFrame

AddToAdjustSizeList, BindToFlow, FindContentByClass, FindContentByClassID, GetControl, GetControlValue, GetPageNumber, GetSearchValue, IsDataFrame, IsFooter, IsHeader, MakeContents, RebindToFlow, SearchAttributeName

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcBasePage::BalanceFlows method

Implements the BalanceFlows property. The BalanceFlows property specifies whether the contents of the page should be redistributed to make all flows on the page the same height. The default value is False.

Syntax Function BalanceFlows( ) As Boolean

Returns True if the contents of a page should be redistributed.
False if the contents of a page should not be redistributed.

AcBasePage::GetFirstDataFrame method

Retrieves the first data frame on a page. A data frame contains data rows.

Syntax Function GetFirstDataFrame( ) As AcFrame

Returns The first frame that contains data rows.

AcBasePage::GetLastDataFrame method

Retrieves the last data frame on a page. A data frame contains data rows.

Syntax Function GetLastDataFrame( ) As AcFrame

Returns The last frame that contains data rows.

Class AcBrowserScriptingControl

Supports the insertion of custom browser code in a report design. Figure 7-3 shows the class hierarchy of AcBrowserScriptingControl.

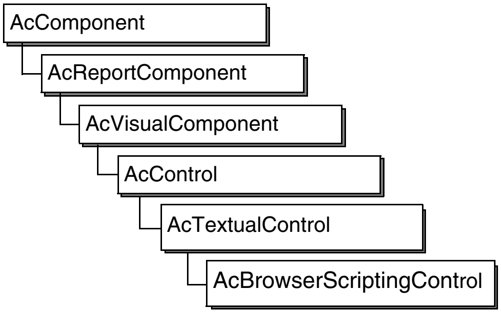


Figure 7-3 AcBrowserScriptingControl

Description Use AcBrowserScriptingControl to insert custom browser code into a report design. For example, you can create a drop-down list and make it available to report users who view the report on the web. Custom browser code can be any code interpreted by a web browser, including:

- JavaScript
- Java applets
- VBScript

Characters in the browser scripting control that have special meaning for the web browser are not converted by the DHTML converter. Instead, the DHTML converter creates a block of HTML code called the context block. The web browser then interprets the code in the BrowserCode property when the report user views the report in DHTML format.

Properties

AcBrowserScriptingControl properties are listed in Table 7-5.

Table 7-5 AcBrowserScriptingControl properties

Property	Type	Description
AlternateText	String	Specifies the string to show when viewing or printing the report in any environment except a web browser. The default value is “”.

Table 7-5 AcBrowserScriptingControl properties

Property	Type	Description
BrowserClipping	AcBrowserClipping	Specifies how to clip text in the browser scripting control when it is viewed in a web browser. The default value is NoClipping.
BrowserCode	String	The custom browser code. The default value is “”.
DebugOption	Boolean	Selects whether the alternate text or browser code is displayed in a web browser. True displays the alternate text, False displays the custom browser code. The default value is False.
Selectable	Boolean	Indicates whether a user can select the control. The default value is True.

Methods for Class AcBrowserScriptingControl

Methods defined in class AcBrowserScriptingControl

BrowserCode, GetText, OnViewCode

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcBrowserScriptingControl::BrowserCode method

Retrieves the value of the BrowserCode property. Override BrowserCode( ) to generate DHTML code dynamically.

Syntax Function BrowserCode( ) As String

AcBrowserScriptingControl::GetText method

Retrieves the value of the AlternateText property. Override GetText( ) to generate the string for PDF output dynamically.

Syntax Function GetText( ) As String

Class AcBTree

A collection class that describes objects organized in a balanced tree. Figure 7-4 shows the class hierarchy of AcBTree. A balanced tree is a sorted list of objects. An attribute of the object contains the object's key. Each key is unique.

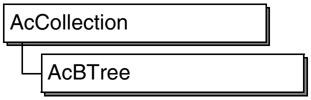


Figure 7-4 AcBTree

Description Use the AcBTree collection class to create a list of objects sorted by one of the object's attributes. To create an object in a BTree, use CreateNode( ) or Insert( ). To locate an object in a BTree, call Find( ) or FindOrCreate( ) to compare a key specified as an argument to the keys of the objects in the balanced tree. Objects can be located with speed comparable to a simple binary search. To keep the storage needed small, the key is contained in the object only.

Letting a reference to a BTree go out of scope does not recover memory. For this reason, to remove a BTree and recover memory, you must call Abandon( ).

To remove all contents from the collection, call RemoveAll( ).

Subclassing AcBTree

To create a balanced tree collection, perform the following steps:

- Subclass AcBTree.
- Specify the NodeSize as an integer equal to the maximum number of objects in the balanced tree.
- Override GetKey( ) to specify how to determine an object's key.

See also Class AcCollection
Class AcList
Class AcObjectArray
Class AcSingleList

Variable

Table 7-6 describes the AcBTree variable.

Table 7-6 AcBTree variable

Variable	Type	Description
NodeSize	Integer	The maximum size of the collection

Methods for Class AcBTree

Methods defined in Class AcBTree

Abandon, CompareKey, CreateNode, Find, FindOrCreate, GetKey, Insert, New

Methods inherited from Class AcCollection

Compare, Contains, Copy, FindByValue, GetCount, IsEmpty, NewIterator, Remove, RemoveAll

AcBTree::Abandon method

Removes an object that the balanced tree no longer needs and recovers memory.

Syntax Sub Abandon( )

AcBTree::CompareKey method

Compares the values of two keys.

Syntax Function CompareKey(key1 As Variant, obj2 As AnyClass) As Integer

Parameters **key1**
The first key in the comparison.

obj2
The object whose key is being compared to key1.

Returns -1 if key1 is less than the value of the key for obj2.
-1 if key1 is Null and the key to obj2 is not Null.
1 if key1 is greater than the key for obj2.
1 if the key for obj2 is Null and key1 is not Null.
0 if the two keys are equal or if both keys are Null.

AcBTree::CreateNode method

Adds an object to the collection. The logic to determine the object's key in the GetKey( ) override must be able to handle the new object's data type. Also, you must ensure that the new object's key does not already exist in the collection. If an object with the same key already exists in the collection, CreateNode( ) returns an Actuate Basic error. If you do not know whether the new object's key is unique, call FindOrCreate( ) to add the object.

Syntax Function CreateNode (key As Variant) As AnyClass

Parameter **obj**
The object to create in the collection.

See also AcBTree::FindOrCreate method

AcBTree::GetKey method
AcBTree::Insert method

AcBTree::Find method

Finds an object in the collection that has the specified key. Before you call Find to locate objects, you must override GetKey( ) to tell the framework how to determine the object's key.

Syntax Function Find(key As Variant) As AnyClass

Parameter **key**
The key of the object to be located.

Returns An object.
Nothing if the object does not exist.

See also AcBTree::FindOrCreate method
AcBTree::GetKey method

AcBTree::FindOrCreate method

Finds an object in the collection with the specified key or creates an object if there is no match to the specified key. If no object with the specified key exists, FindOrCreate( ) adds a new object to the collection with the specified key. Use FindOrCreate( ) when you need to add an object but you do not know if the object already exists in the collection. By using FindOrCreate( ), you eliminate the need to first locate the object using Find( ) and then add the object using CreateNode( ). Before you call FindOrCreate( ), you must override GetKey( ) to describe how to determine an object's key.

Syntax Function FindOrCreate(key As Variant) As AnyClass

Parameter **key**
The key of the object to be located.

Returns An object that has the specified key.
Nothing if no object with the specified key exists.

See also AcBTree::GetKey method

AcBTree::GetKey method

Determines the key for an object in a collection. Override this method to define the logic that determines the object's key.

Syntax Function GetKey(obj As AnyClass) As Variant

Parameter **obj**
The object in the collection.

Returns The key stored as a Variant data type.

Example In the following example, the objects in the collection are text controls in a report. The object's key is the text control's DataValue property. The following code shows how to override the GetKey( ) method to determine the key:

```
Function GetKey( obj As AnyClass ) As Variant
   Dim textControl As AcTextControl
   Set textControl = obj
   GetKey = textControl.DataValue
End Function
```

AcBTree::Insert method

Adds an object to the collection. The key determination logic in the GetKey( ) method override must be able to handle the new object's data type. Also, you must ensure that the new object's key does not already exist in the collection. If an object with the same key already exists in the collection, Insert( ) returns an Actuate Basic error. If you do not know whether the new object's key is unique, call FindOrCreate( ) to add the object.

Syntax Sub Insert(obj As AnyClass)

Parameter **obj**
The object to add to the collection.

Example The following code inserts an object into a balanced tree:

```
Sub Insert( obj As AnyClass )
   Dim newNode As AcBTreeNode

' Call a private method, CreateRoot( ), to create the root
' if no root exists.
   If Root Is Nothing Then
      CreateRoot( )
   End If

' Insert the object and increment the object count.
   Set newNode = Root.Insert( GetKey( obj ), obj )
   Count = Count + 1
   If Not newNode Is Nothing Then
      SplitRoot( newNode )
   End If

' Cache the last object seen.
   Set LastObjectSeen = obj
End Sub
```

See also AcBTree::FindOrCreate method
AcBTree::GetKey method

AcBTree::New method

The constructor method for the AcBTree class.

Syntaxes

Sub New( )

Sub New(size As Integer)

Parameter

size
The maximum number of items in a node in the BTree. If an addition to the node increases the size beyond this value, the node splits.

Class AcChart

Displays a chart. Figure 7-5 shows the class hierarchy of AcChart.

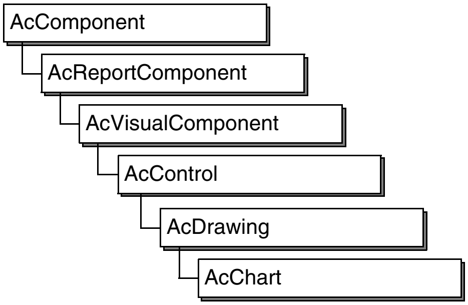


Figure 7-5 AcChart

Description

AcChart takes data from one or more data rows and organizes it into one or more series of data points displayed as a chart. AcChart builds a data structure of objects that represent the various elements of a chart, such as axes, categories, and points. This data structure is built from the following classes:

- AcChartLayer
- AcChartAxis
- AcChartCategory
- AcChartSeries
- AcChartPoint
- AcChartPointStyle
- AcChartSeriesStyle

You can define most charts using the Chart Builder and Advanced Chart Options dialogs. If you have more advanced requirements, you can usually get the results you need by overriding one or two methods in a chart. You also can create a chart dynamically without Chart Builder, using only the standard Actuate Foundation Classes. Using the AFC gives you total control over the content and appearance of the resulting chart.

Chart life cycle

When a report runs, a chart is created in the following series of steps:

- Within the chart's Start( ) method:
  - The high-level structure of the chart is initialized. For example, this is where overlay and study layers are enabled.

  - The chart's CustomizeChart( ) method is called. You can override this method to alter the high-level structure of the chart. For example, you could override CustomizeChart( ) to disable the overlay layer.
  - The chart's layers are created and initialized.
  - The chart's CustomizeLayers( ) method is called. You can override this method to modify the appearance of individual layers within the chart. For example, you could override CustomizeLayers( ) to change the type of chart displayed from a bar chart to a pie chart.
  - The chart's status is set to ChartStatusBuilding.
- Within the chart's BuildFromRow( ) method:
  - The chart's OnRow( ) method is called. You can override this method to use data from the data rows to modify chart settings. For example, you could override OnRow( ) to set the chart's title using data from a data row.
  - The chart's layers accumulate data from data rows until no more rows are available.
  - The chart's layers create categories, series, and points from the accumulated data.
  - The chart's CustomizeCategoriesAndSeries( ) method is called. You can override this method to adjust the data displayed in the chart. For example, you could override CustomizeCategoriesAndSeries( ) to add a category that shows the average of all the other categories.
  - For all types of chart layers other than pie chart layers, series styles are created for each series. For pie chart layers, series styles are created for each category.
  - The chart's CustomizeSeriesStyles( ) method is called. You can override this method to modify the appearance of individual series or pie slices in the chart. For example, you could override CustomizeSeriesStyles( ) to change the colors of the series.
  - The minimum and maximum data values in each chart layer and the chart as a whole are calculated.
  - The chart's axes are created and initialized.
  - The chart's CustomizeAxes( ) method is called. You can override this method to change the appearance of individual axes in the chart. For example, you could override CustomizeAxes( ) to add minor grid lines to one of the chart's axes.
  - The chart's ComputeScales( ) method is called to compute the axis scales.
  - The chart's AdjustChart( ) method is called. You can override this method to make final adjustments to the chart once all its automatic layout has been created. For example, you could override AdjustChart( ) to scale the

labels on one of the chart's axes depending on the automatically computed label values, then modify the axis's title to match.

- The chart's status is set to ChartStatusFinishedBuilding.

- Within the chart's Finish( ) method, the chart's status is set to ChartStatusFinished.

When a report is being viewed within the chart's ApplyVisitor( ) method, the chart's Localize( ) method is called. You can override this method to localize the chart at view time, using the user's viewing locale.

Example The following example overrides a frame's Finish( ) method to create and populate a simple chart with a bar chart base layer and a bar chart study layer using the standard AcChart class:

```
Sub Finish( )
   Dim chart As AcChart
   ' Create a new chart object.
   Set chart = New Persistent AcChart
   chart.Size.Width = 8 * OneInch
   chart.Size.Height = 6 * OneInch
   ' Add the chart to this frame.
   AddContent( chart )
   ' Initialize the chart.
   chart.StartEmpty( )
   ' Add a title.
   chart.SetTitleText( "2003 Sales & Profits" )

   ' Create the chart layers.
   chart.EnableStudyLayers( 1 )
   chart.MakeLayers( )
   Dim baseLayer As AcChartLayer
   Set baseLayer = chart.GetBaseLayer( )
   Dim studyLayer As AcChartLayer
   Set studyLayer = chart.GetStudyLayer( 1 )
   ' Make the chart a bar chart with a bar study.
   baseLayer.SetChartType( ChartTypeBar,
+     ChartSeriesPlacementSideBySide )
   studyLayer.SetChartType( ChartTypeBar,
+     ChartSeriesPlacementSideBySide )
   ' Initialize the chart layers.
   chart.StartLayers( )
   ' The chart is now building itself.
   chart.SetStatus( ChartStatusBuilding )

   ' Add 4 categories.
   baseLayer.AddCategory( #2003-01-01# )
   baseLayer.AddCategory( #2003-04-01# )
   baseLayer.AddCategory( #2003-07-01# )
```

```
    baseLayer.AddCategory( #2003-10-01# )
    ' Format the category labels as quarters.
    baseLayer.SetCategoryLabelFormat( "Short Quarter" )
    ' Add 2 base series with points.
    Dim series As AcChartSeries
    Set series = baseLayer.AddSeries( "Domestic" )
    series.AddPoint( 100 )
    series.AddPoint( 150 )
    series.AddPoint( 200 )
    series.AddEmptyPoint( )
    Set series = baseLayer.AddSeries( "Export" )
    series.AddEmptyPoint( )
    series.AddPoint( 75 )
    series.AddPoint( 150 )
    series.AddEmptyPoint( )
    ' Add 2 overlay series with points.
    Set series = studyLayer.AddSeries( "Domestic" )
    series.AddPoint( 10 )
    series.AddPoint( 15 )
    series.AddPoint( 40 )
    series.AddEmptyPoint( )
    Set series = studyLayer.AddSeries( "Export" )
    series.AddEmptyPoint( )
    series.AddPoint(  5 )
    series.AddPoint( 20 )
    series.AddEmptyPoint( )
    ' Compute the minimum and maximum data values.
    chart.ComputeMinMaxDataValues( )

    ' Add the axes.
    chart.MakeAxes( )
    ' Give the base y-axis a title.
    Dim yAxis As AcChartAxis
    Set yAxis = baseLayer.GetYAxis( )
    yAxis.SetTitleText( "Sales (US$M)" )
    ' Give the study y-axis a title.
    Set yAxis = studyLayer.GetYAxis( )
    yAxis.SetTitleText( "Profit (US$M)" )
    ' Compute the axis scales.
    chart.ComputeScales( )
    ' Mark the chart complete.
    chart.SetStatus( ChartStatusFinishedBuilding )
    Super::Finish( )
End Sub
```

See also Class AcDrawing
Class AcChartAxis
Class AcChartCategory

Class AcChartGridLine
Class AcChartLayer
Class AcChartPoint
Class AcChartPointStyle
Class AcChartSeries
Class AcChartSeriesStyle
Class AcChartTrendline

Property

Table 7-7 describes the AcChart property.

Table 7-7 AcChart property

Property	Type	Description
Definition	AcPointer	An internal representation of a chart definition. Use the Chart Builder and Advanced Chart Options dialogs to view and change this definition.

Methods for Class AcChart

Methods defined in Class AcChart

AdjustChart, BaseAndOverlayScalesAreMatched, BuildFromRow, BuildSampleCategoryScaleData, BuildSampleValueScaleData, ComputeMinMaxDataValues, ComputeScales, CustomizeAxes, CustomizeCategoriesAndSeries, CustomizeChart, CustomizeLayers, CustomizeSeriesStyles, DescribeLayout, DisableHyperchart, DisableOverlayLayer, DisableStudyLayers, DrawOnChart, EnableHyperchart, EnableOverlayLayer, EnableStudyLayers, FlipAxes, GetBaseLayer, GetBorderStyle, GetChartDrawingPlane, GetFillStyle, GetHyperchartLink, GetLayer, GetLegendBackgroundColor, GetLegendBorderStyle, GetLegendFont, GetLegendPlacement, GetNumberOfLayers, GetNumberOfStudyLayers, GetOverlayLayer, GetStudyLayer, GetTitleStyle, GetTitleText, HasOverlayLayer, IsHyperchart, IsThreeD, Localize, MakeAxes, MakeLayers, SetBackgroundColor, SetBorderStyle, SetFillStyle, SetFlipAxes, SetLegendBackgroundColor, SetLegendBorderStyle, SetLegendFont, SetLegendPlacement, SetMatchBaseAndOverlayScales, SetStatus, SetThreeD, SetTitleStyle, SetTitleText, StartEmpty, StartLayers

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcChart::AdjustChart method

Override AdjustChart( ) to make final adjustments to a chart after all its automatic layout has been created. You can make most adjustments to a chart while it is being constructed by overriding methods such as CustomizeLayers( ). There are a few adjustments that you can only make when the chart finishes computing its automatic layout. Override the AdjustChart( ) method to make those adjustments.

Syntax Sub AdjustChart(baseLayer As AcChartLayer, overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer)

Parameters **baseLayer**
A reference to the chart's base layer object.

overlayLayer
A reference to the chart's overlay layer object. Nothing if the chart has no overlay layer.

studyLayers( )
An array of references to the chart's study layer objects. To determine how many study layers there are in a chart, call the chart's GetNumberOfStudyLayers( ) method.

Example The following example adjusts the upper limit of a study layer's *y*-axis so that it is at least 100. This adjustment can only be made in AdjustChart( ) because it relies on the automatically computed upper limit.

```
Sub AdjustChart( baseLayer As AcChartLayer,
+  overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = studyLayers(1).GetYAxis( )
   If (yAxis.GetMaximumValue( ) < 100) Then
      yAxis.SetMaximumValue( 100 )
      ' Recompute the ticks and labels.
      yAxis.ComputeScale( )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChart::CustomizeCategoriesAndSeries method
AcChart::CustomizeChart method
AcChart::CustomizeLayers method
AcChart::CustomizeSeriesStyles method
AcChart::GetNumberOfStudyLayers method
AcChart::Localize method

AcChart::BaseAndOverlayScalesAreMatched method

Determines whether the base and overlay *y*-axis scales of a chart have been forced to be identical.

Syntax Function BaseAndOverlayScalesAreMatched( ) As Boolean

Returns True if the base and overlay *y*-axis scales of the chart are forced to be identical.
False if the base and overlay *y*-axis scales are not forced to be identical.

See also AcChart::SetMatchBaseAndOverlayScales method

AcChart::BuildFromRow method

Override the BuildFromRow( ) method to manipulate the raw data to be displayed in a chart. The AFC framework calls the BuildFromRow( ) method automatically once for each data row in the chart's parent data section, and once with dataRow set to Nothing. Within this method, a chart accumulates the data to display.

To manipulate the raw data to be displayed in a chart, override this method. Within BuildFromRow( ) you can

- Skip data rows by not calling Super::BuildFromRow( ) and returning ContinueBuilding.
- Add calculated data to a chart by creating your own data rows and calling Super::BuildFromRow( ) repeatedly.
- Finish building a chart at any time by calling Super::BuildFromRow(Nothing) and returning the result.

When you override this method, you must

- Always handle the case where dataRow is Nothing.
- Always call Super::BuildFromRow(Nothing) to finish building the chart.
- Always return FinishedBuilding when the chart is complete.

Syntax Function BuildFromRow(dataRow As AcDataRow) As AcBuildStatus

Parameter **dataRow**
A reference to a data row whose values are to be displayed in the chart.

When the AFC framework calls BuildFromRow( ) with dataRow set to Nothing, the chart finishes building itself.

Returns The build status of the chart.

Examples A chart's default behavior is to process multiple data rows. If a chart is placed in a Content frame, the result is a single chart that displays all the data rows for the Content frame's parent section.

The following example overrides BuildFromRow( ) to make a chart from a single row. If the chart is in a Content frame, a separate chart appears for each data row.

```
Function BuildFromRow( dataRow As AcDataRow ) As AcBuildStatus
   ' Process the first data row.
   BuildFromRow = Super::BuildFromRow( dataRow )
   If Not dataRow Is Nothing Then
      ' Force the chart to finish building itself.
      BuildFromRow = Super::BuildFromRow( Nothing )
   End If
End Function
```

In the following example, BuildFromRow( ) filters out data rows for New York:

```
Function BuildFromRow( dataRow As AcDataRow ) As AcBuildStatus
   If Not dataRow Is Nothing Then
      If (GetValue( dataRow, "customers_state" ) = "NY") Then
         ' Do not process the row.
         BuildFromRow = ContinueBuilding
         Exit Function
      End If
   End If
   ' Process the row as usual.
   BuildFromRow = Super::BuildFromRow( dataRow )
End Function
```

See also AcReportComponent::BuildFromRow method

AcChart::BuildSampleCategoryScaleData method

Call the BuildSampleCategoryScaleData( ) method to generate sample data for a chart whose *x*-axis is based on categories. If you are developing a custom chart class with overridden methods, you can make a simple test harness that does not require a data source. You can use BuildSampleCategoryScaleData( ) to populate a custom chart with realistic sample data.

Call this method from AcChart::BuildFromRow( ). You cannot call BuildSampleCategoryScaleData( ) on a scatter chart.

If you call BuildSampleCategoryScaleData( ) on a pie chart, numberOfBaseSeries must be set to 1. Both numberOfOverlaySeries and numberOfStudySeries must be set to 0.

Syntax Sub BuildSampleCategoryScaleData(numberOfCategories As Integer, numberOfBaseSeries As Integer, numberOfOverlaySeries As Integer, numberOfStudySeries As Integer, baseMinimumValue As Double, baseMaximumValue As Double, overlayMinimumValue As Double, overlayMaximumValue As Double, studyMinimumValue As Double, studyMaximumValue As Double)

Parameters **numberOfCategories**
The number of sample categories to be displayed on the *x*-axis.

numberOfBaseSeries
The number of sample series to be displayed in the chart's base layer.

numberOfOverlaySeries
The number of sample series to be displayed in the chart's overlay layer.

numberOfStudySeries
The number of sample series to be displayed in the chart's first study layer.

baseMinimumValue
The minimum sample value to be displayed in the chart's base layer.

baseMaximumValue
The maximum sample value to be displayed in the chart's base layer.

overlayMinimumValue
The minimum sample value to be displayed in the chart's overlay layer.

overlayMaximumValue
The maximum sample value to be displayed in the chart's overlay layer.

studyMinimumValue
The minimum sample value to be displayed in the chart's first study layer.

studyMaximumValue
The maximum sample value to be displayed in the chart's first study layer.

Example If the report containing a chart has no data source, the chart's BuildFromRow( ) method is called once with its parameter set to Nothing. The following code overrides a chart's BuildFromRow( ) method to create sample category scale data:

```
Function BuildFromRow( dataRow As AcDataRow ) As AcBuildStatus
   Assert( dataRow Is Nothing )
   ' Build sample data with 12 categories and 2 series, sample
   ' values in the range -35 through 35, no overlay or study.
   BuildSampleCategoryScaleData( 12, 2, 0, 0, -35.0, 35.0, 0, 0,
      0, 0 )
   BuildFromRow = Super::BuildFromRow( dataRow )
End Function
```

See also AcChart::BuildFromRow method
AcChart::BuildSampleValueScaleData method

AcChart::BuildSampleValueScaleData method

Call the BuildSampleValueScaleData( ) method to create sample data for a scatter chart. If you are developing a custom chart class with overridden methods, you can make a simple test harness that does not require a data source. You can use the BuildSampleValueScaleData( ) method to populate a custom chart with realistic sample data. Call this method from AcChart::BuildFromRow( ).

You can call this method only on a scatter chart.

Syntax Sub BuildSampleValueScaleData(numberOfPoints As Integer,
   numberOfBaseSeries As Integer, numberOfOverlaySeries As Integer,
   numberOfStudySeries As Integer, minimumXValue As Double,
   maximumXValue As Double, baseMinimumYValue As Double,
   baseMaximumYValue As Double, overlayMinimumYValue As Double,
   overlayMaximumYValue As Double, studyMinimumYValue As Double,
   studyMaximumYValue As Double)

Parameters **numberOfPoints**
The number of sample points to be displayed in each series.

numberOfBaseSeries
The number of sample series to be displayed in the chart's base layer.

numberOfOverlaySeries
The number of sample series to be displayed in the chart's overlay layer.

numberOfStudySeries
The number of sample series to be displayed in the chart's first study layer.

minimumXValue
The minimum sample x value to be displayed in the chart.

maximumXValue
The maximum sample x value to be displayed in the chart.

baseMinimumYValue
The minimum sample y value to be displayed in the chart's base layer.

baseMaximumYValue
The maximum sample y value to be displayed in the chart's base layer.

overlayMinimumYValue
The minimum sample y value to be displayed in the chart's overlay layer.

overlayMaximumYValue
The maximum sample y value to be displayed in the chart's overlay layer.

studyMinimumYValue
The minimum sample y value to be displayed in the chart's first study layer.

studyMaximumYValue
The maximum sample y value to be displayed in the chart's first study layer.

Example If the report containing a chart has no data source, the chart's BuildFromRow( ) method is still called once with its dataRow parameter set to Nothing. The following code overrides a chart's BuildFromRow( ) method to create sample category scale data:

```
Function BuildFromRow( dataRow As AcDataRow ) As AcBuildStatus
   Assert( dataRow Is Nothing )
   ' Build sample data with 4 points per series, 3 series,
   ' sample x values in the range -35 through 35, sample y
   ' values in the range -5 through 5, no overlay or study.
   BuildSampleValueScaleData( 4, 3, 0, 0, -35.0, 35.0, -5.0,
      5.0, 0, 0, 0, 0 )
   BuildFromRow = Super::BuildFromRow( dataRow )
End Function
```

See also AcChart::BuildFromRow method
AcChart::BuildSampleCategoryScaleData method

AcChart::ComputeMinMaxDataValues method

Call ComputeMinMaxDataValues( ) to compute the minimum and maximum data values for each layer of a chart and the chart as a whole from the individual data points. If you create a chart dynamically and do not use the standard chart building mechanism, you must call ComputeMinMaxDataValues( ) to compute the minimum and maximum data values in the chart. You must make this call after you create all the categories, series and points in the chart and before you call the chart's MakeAxes( ) method.

Syntax Sub ComputeMinMaxDataValues( )

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

AcChart::ComputeScales method

Call ComputeScales( ) to compute the scales for all the axes of a chart. If you create a chart dynamically and do not use the standard chart building mechanism, you must call ComputeScales( ) to compute the scales for all the axes of the chart. You must make this call after you call the chart's MakeAxes( ) method and before you call the chart's Finish( ) method.

Syntax Sub ComputeScales( )

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

AcChart::CustomizeAxes method

Override the CustomizeAxes( ) method to modify the appearance of a chart's axes. Within this method, you can obtain references to all the axes in a chart and call methods on those axes.

Syntax Sub CustomizeAxes(baseLayer As AcChartLayer, overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer)

Parameters **baseLayer**
A reference to the chart's base layer object.

overlayLayer
A reference to the chart's overlay layer object.

Nothing if the chart has no overlay layer.

studyLayers( )
An array of references to the chart's study layer objects. To find out how many study layers there are in a chart, call the chart's GetNumberOfStudyLayers( ) method.

Example The default scaling for a chart's *y*-axis automatically leaves a small margin between the greatest value and the top of the axis. The following example overrides a chart's CustomizeAxes( ) method to increase the automatic margin to 20 percent of the axis's height:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers( ) As
   AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   yAxis.SetOuterMarginRatio( 0.2 )
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeCategoriesAndSeries method
AcChart::CustomizeChart method
AcChart::CustomizeLayers method
AcChart::CustomizeSeriesStyles method
AcChart::GetNumberOfStudyLayers method
AcChart::Localize method
AcChartLayer::GetXAxis method
AcChartLayer::GetYAxis method
Class AcChartAxis

AcChart::CustomizeCategoriesAndSeries method

Override the CustomizeCategoriesAndSeries( ) method to add, remove, or modify categories, series, and points within a chart.

Syntax Sub CustomizeCategoriesAndSeries(baseLayer As AcChartLayer, overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer)

Parameters **baseLayer**
A reference to the chart's base layer object.

overlayLayer
A reference to the chart's overlay layer object. Nothing if the chart has no overlay layer.

studyLayers( )
An array of references to the chart's study layer objects. To find out how many study layers there are in a chart, call the chart's GetNumberOfStudyLayers( ) method.

Example The following example overrides a chart's CustomizeCategoriesAndSeries( ) method to insert a new category. The new category appears as the first category on the *x*-axis. The points for each series in the new category are populated with the mean value of the other points in the same series.

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' Insert a new category.
   Dim newCategory As AcChartCategory
   Set newCategory = baseLayer.InsertCategory( 1, "Mean" )
   ' Loop through all the series.
   Dim numberOfSeries As Integer
   numberOfSeries = baseLayer.GetNumberOfSeries( )
   Dim seriesIndex As Integer
   For seriesIndex = 1 To numberOfSeries
      Dim series As AcChartSeries
      Set series = baseLayer.GetSeries( seriesIndex )
      ' Get the mean value of the points in the series.
      Dim numberOfPoints As Integer
      numberOfPoints = series.GetNumberOfPoints( )
      Dim point As AcChartPoint
      Dim pointIndex As Integer
      Dim total As Double
      total = 0
      Dim count As Integer
      count = 0
      ' Ignore the first point in each series, because
      ' that point belongs to the new category.
      For pointIndex = 2 To numberOfPoints
         Set point = series.GetPoint( pointIndex )
         ' Ignore missing values.
         If Not point.IsMissing( ) Then
            total = total + point.GetYValue( )
            count = count + 1
         End If
      Next pointIndex

      ' Put the mean value into the point for the new category.
      Set point = series.GetPoint( 1 )
      point.SetYValue( total / count )
   Next seriesIndex
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::CustomizeChart method
AcChart::CustomizeLayers method
AcChart::CustomizeSeriesStyles method
AcChart::DisableStudyLayers method
AcChart::Localize method
AcChartLayer::AddCategory method
AcChartLayer::AddSeries method
AcChartLayer::GetCategory method
AcChartLayer::GetNumberOfCategories method
AcChartLayer::GetNumberOfSeries method
AcChartLayer::GetSeries method
AcChartLayer::InsertCategory method

AcChartLayer::InsertSeries method
AcChartLayer::RemoveCategory method
AcChartLayer::RemoveSeries method
AcChartSeries::AddEmptyPoint method
AcChartSeries::AddPoint method
AcChartSeries::GetNumberOfPoints method
AcChartSeries::GetPoint method
AcChartSeries::InsertEmptyPoint method
AcChartSeries::InsertPoint method
AcChartSeries::RemovePoint method
Class AcChartLayer
Class AcChartPoint
Class AcChartSeries

AcChart::CustomizeChart method

Override the CustomizeChart( ) method to modify the basic structure and appearance of a chart.

Syntax Sub CustomizeChart( )

Example The following example overrides a chart's CustomizeChart( ) method to disable the overlay layer, depending on the value of a Boolean parameter:

```
Sub CustomizeChart( )
   If Not parmShowOverlay Then
      DisableOverlayLayer( )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::CustomizeCategoriesAndSeries method
AcChart::CustomizeLayers method
AcChart::CustomizeSeriesStyles method
AcChart::Localize method

AcChart::CustomizeLayers method

Override the CustomizeLayers( ) method to modify the appearance of the individual layers of a chart.

Syntax Sub CustomizeLayers(baseLayer As AcChartLayer, overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer)

Parameters **baseLayer**
A reference to the chart's base layer object.

overlayLayer
A reference to the chart's overlay layer object. Nothing if the chart has no overlay layer.

studyLayers( )
An array of references to the chart's study layer objects. To find out how many study layers there are in a chart, call the chart's GetNumberOfStudyLayers( ) method.

Example The following example overrides a chart's CustomizeLayers( ) method to show values side-by-side or as stacked percentages, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers( ) As
   AcChartLayer )
   If parmShowAsPercentages Then
      baseLayer.SetSeriesPlacement(
      ChartSeriesPlacementAsPercentages )
   Else
      baseLayer.SetSeriesPlacement(
      ChartSeriesPlacementSideBySide )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::CustomizeCategoriesAndSeries method
AcChart::CustomizeChart method
AcChart::CustomizeSeriesStyles method
AcChart::GetNumberOfStudyLayers method
AcChart::Localize method
Class AcChartLayer

AcChart::CustomizeSeriesStyles method

Override the CustomizeSeriesStyles( ) method to modify the appearance of individual series or pie slices in a chart.

Syntax Sub CustomizeSeriesStyles(baseLayer As AcChartLayer, overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer)

Parameters **baseLayer**
A reference to the chart's base layer object.

overlayLayer
A reference to the chart's overlay layer object. Nothing if the chart has no overlay layer.

studyLayers( )
An array of references to the chart's study layer objects. To determine the number of study layers in a chart, call the chart's GetNumberOfStudyLayers( ) method.

Example The following example overrides a chart's CustomizeSeriesStyles( ) method to give the first series a green background:

```
Sub CustomizeSeriesStyles( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim seriesStyle As AcChartSeriesStyle
   Set seriesStyle = baseLayer.GetSeriesStyle( 1 )
   seriesStyle.SetBackgroundColor( Green )
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::CustomizeCategoriesAndSeries method
AcChart::CustomizeChart method
AcChart::CustomizeLayers method
AcChart::GetNumberOfStudyLayers method
AcChart::Localize method
AcChartLayer::GetSeriesStyle method
Class AcChartSeriesStyle

AcChart::DescribeLayout method

Call this method to compute the layout of a chart without rendering it. You can get information about the chart's layout by calling DescribeLayout( ) then calling the following methods:

- AcChartLayer::GetPieCenter method
- AcChartLayer::GetPieRadius method
- AcChartLayer::GetPlotAreaPosition method
- AcChartLayer::GetPlotAreaSize method

You can call DescribeLayout( ) only from the DrawOnChart( ) method.

Syntax Sub DescribeLayout( )

Example For an example of how to use this method, see the example for the AcChart::DrawOnChart( ) method.

See also AcChart::DrawOnChart method
AcChartLayer::GetPieCenter method
AcChartLayer::GetPieRadius method
AcChartLayer::GetPlotAreaPosition method
AcChartLayer::GetPlotAreaSize method

AcChart::DisableHyperchart method

If you define hyperchart links in a chart, you can disable the links in code by calling the DisableHyperchart( ) method from the chart's CustomizeChart( ) method. You can call DisableHyperchart( ) only from CustomizeChart( ).

Syntax Sub DisableHyperchart( )

Example The following example overrides a chart's CustomizeChart( ) method to disable hyperchart links, depending on the value of a Boolean parameter:

```
Sub CustomizeChart( )
   If Not parmAllowHyperlinks Then
      DisableHyperchart( )
   End If
End Sub
```

See also AcChart::CustomizeChart method
AcChart::EnableHyperchart method

AcChart::DisableOverlayLayer method

If you define an overlay layer in a chart, you can disable the overlay layer programmatically by calling the DisableOverlayLayer( ) method from the chart's CustomizeChart( ) method. You can call DisableOverlayLayer( ) only from the CustomizeChart( ) method.

Syntax Sub DisableOverlayLayer( )

Example The following example overrides a chart's CustomizeChart( ) method to disable the overlay layer, depending on the value of a Boolean parameter:

```
Sub CustomizeChart( )
   If Not parmShowOverlay Then
      DisableOverlayLayer( )
   End If
End Sub
```

See also AcChart::CustomizeChart method
AcChart::DisableStudyLayers method
AcChart::EnableOverlayLayer method

AcChart::DisableStudyLayers method

If you define a study layer in a chart, you can disable the study layer programmatically by calling the DisableStudyLayers( ) method from the chart's CustomizeChart( ) method. You can call DisableStudyLayers( ) only from the CustomizeChart( ) method.

Syntax Sub DisableStudyLayers( )

Example The following example overrides a chart's CustomizeChart( ) method to disable the study layer, depending on the value of a Boolean parameter:

```
Sub CustomizeChart( )
   If Not parmShowStudy Then
      DisableStudyLayers( )
   End If
End Sub
```

See also AcChart::CustomizeChart method
AcChart::DisableOverlayLayer method
AcChart::EnableStudyLayers method

AcChart::DrawOnChart method

Override this method to add drawing elements such as lines, rectangles, and text to a chart.

Syntax Sub DrawOnChart(baseLayer As AcChartLayer,
   overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer)

Example In the following example, the values in a line chart are known to have a margin of error of ±10%. The chart's DrawOnChart( ) method has been overridden to draw points as bars that show the range of possible values.

```
Sub DrawOnChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
  ' Get the position and size of the chart's plot area
  DescribeLayout( )
  Dim plotAreaPosition As AcPoint
  plotAreaPosition = baseLayer.GetPlotAreaPosition( )
  Dim plotAreaSize As AcSize
  plotAreaSize = baseLayer.GetPlotAreaSize( )

  ' Convert values to points
  Dim x As Double
  x = plotAreaPosition.X / OnePoint
  Dim y As Double
  y = plotAreaPosition.Y / OnePoint
  Dim w As Double
  w = PlotAreaSize.Width / OnePoint
  Dim h As Double
  h = PlotAreaSize.Height / OnePoint

  Dim series As AcChartSeries
  Set series = baseLayer.GetSeries( 1 )
  ' Get the visual attributes of the bars for the series
  Dim seriesStyle As AcChartSeriesStyle
  Set seriesStyle = baseLayer.GetSeriesStyle( 1 )
  Dim fillStyle As AcDrawingFillStyle
  fillStyle = seriesStyle.GetFillStyle( )
  Dim seriesColor As AcColor
  seriesColor = fillStyle.Color1
  Dim numberOfPoints As Integer
  numberOfPoints = series.GetNumberOfPoints( )
  Dim categoryWidth As Double
  categoryWidth = (w / numberOfPoints)
  Dim yAxis As AcChartAxis
  Set yAxis = baseLayer.GetYAxis( )
  Dim minY As Double
  minY = yAxis.GetMinimumValue( )
  Dim maxY As Double
  maxY = yAxis.GetMaximumValue( )
  Dim yRange As Double
  yRange = maxY - minY

  Dim errorMargin As Double
  errorMargin = 0.1 ' +/- 10% error

  ' Create SVG to draw the error bars
  Dim svg As String
  svg = "<svg version='1.1'"
+ ' Standard SVG 1.1 namespaces
+ & " xmlns='http://www.w3.org/2000/svg'"
+ & " xmlns:xlink='http://www.w3.org/1999/xlink'"
+ ' Do not collapse whitespace in text
+ & " xml:space='preserve'"
+ ' Scale the SVG to use points as the default units
+ & " viewBox='0 0 " & SVGDbl( w ) & " " & SVGDbl( h ) & "'>"

  ' Draw the error bars for each point
  Dim pointIndex As Integer
  For pointIndex = 1 To numberOfPoints
    Dim point As AcChartPoint
    Set point = series.GetPoint( pointIndex )
    If Not point.IsMissing( ) Then
      Dim pointYValue As Double
      pointYValue = point.GetYValue( )
      Dim pointX As Double
      pointX = (pointIndex - 0.5) * categoryWidth
      Dim pointY As Double
      pointY = (1 - ((pointYValue - minY) / yRange)) * h
      Dim errorBarHeight As Double
      errorBarHeight
+     = (pointYValue / yRange) * (errorMargin * 2) * h
      svg = svg
+     & "<rect"
+     & SVGAttr( "x", pointX - 8 )
+     & SVGAttr( "y", pointY - (errorBarHeight / 2) )
+     & SVGAttr( "width", 16 )
+     & SVGAttr( "height", errorBarHeight )
+     & SVGColorAttr( "fill", seriesColor )
+     & " stroke='black' stroke-width='0.667'/>"
    End If
    ' Hide the original bar
    Dim pointStyle As AcChartPointStyle
    Set pointStyle = point.AddCustomStyle( )
    fillStyle.Color1 = Transparent
    pointStyle.SetFillStyle( fillStyle )
    borderStyle.Pen = DrawingLinePenNone
    pointStyle.SetBorderStyle( borderStyle )
  Next pointIndex

  svg = svg & "</svg>"

  Dim svgPlane As AcDrawingSVGPlane
  Set svgPlane = AddDrawingPlane( DrawingPlaneTypeSVG )
  ' Position and size the SVG drawing plane
  ' to match the chart's plot area
  svgPlane.SetPosition( plotAreaPosition.X,
+   plotAreaPosition.Y )
  svgPlane.SetSize( PlotAreaSize.Width, PlotAreaSize.Height )
  svgPlane.SetSVG( svg )
End Sub
```

See also AcChart::DescribeLayout method
Class AcDrawing

AcChart::EnableHyperchart method

Call EnableHyperchart( ) to enable hyperchart links in a chart. Hyperchart links are enabled automatically if any of the category, series, or point link expressions are set in Advanced Chart Options. If you override GetHyperchartLink( ) to compute hyperchart links instead of using one of the hyperchart link expressions, hyperchart links are not enabled automatically. In this case, you must call EnableHyperchart( ) from the chart's CustomizeChart( ) method.

You can call EnableHyperchart( ) only from the CustomizeChart( ) method.

Syntax Sub EnableHyperchart( )

Example The following example overrides a chart's CustomizeChart( ) method to enable hyperchart links:

```
Sub CustomizeChart( )
   EnableHyperchart( )
End Sub
```

See also AcChart::CustomizeChart method
AcChart::DisableHyperchart method
AcChart::GetHyperchartLink method

AcChart::EnableOverlayLayer method

Call the EnableOverlayLayer( ) method to enable the overlay layer of a chart.

You can call this method only from:

- A chart's CustomizeChart( ) method
- Code that is creating a chart dynamically, before you call the chart's MakeLayers( ) method

If you call EnableOverlayLayer( ), you must also populate the overlay layer programmatically. It is often easier to define an overlay layer using Chart Builder. To populate the overlay layer programmatically, leave Chart Builder's Overlay Data page blank and create series and points in the chart's CustomizeCategoriesAndSeries( ) method.

Syntax Sub EnableOverlayLayer( )

Examples The following example overrides a chart's CustomizeChart( ) method to enable the chart's overlay layer:

```
Sub CustomizeChart( )
   EnableOverlayLayer( )
End Sub
```

For another example of how to use this method, see the dynamic chart example for the AcChart class.

See also AcChart::CustomizeCategoriesAndSeries method
AcChart::CustomizeChart method
AcChart::DisableOverlayLayer method
AcChart::EnableStudyLayers method
Class AcChart
Class AcChartLayer

AcChart::EnableStudyLayers method

Call EnableStudyLayers( ) to add one or more study layers to a chart programmatically. This method is the only way you can add multiple study layers to a chart.

If you call EnableStudyLayers( ), you must also populate the study layer programmatically. It is often easier to define a study layer using Chart Builder. To populate the study layer leave Chart Builder's Study Data page blank and create series and points in the chart's CustomizeCategoriesAndSeries( ) method.

You can call EnableStudyLayers only from:

- A chart's CustomizeChart( ) method
- Code that is creating a chart dynamically, before you call the chart's MakeLayers( ) method

Syntax Sub EnableStudyLayers(numberOfStudyLayers As Integer)

Examples The following example overrides a chart's CustomizeChart( ) method to enable two study layers:

```
Sub CustomizeChart( )
   EnableStudyLayers( 2 )
End Sub
```

For another example of how to use this method, see the dynamic chart example for the AcChart class.

See also AcChart::CustomizeCategoriesAndSeries method
AcChart::CustomizeChart method
AcChart::DisableStudyLayers method
AcChart::EnableOverlayLayer method
Class AcChart
Class AcChartLayer

AcChart::FlipAxes method

Determines whether a chart's *x*-axis and *y*-axis are reversed. If the axes are reversed, the *x*-axis displays vertically and the *y*-axis displays horizontally.

Syntax Function FlipAxes( ) As Boolean

Returns True if the chart's *x*-axis and *y*-axis are reversed.
False if the chart's *x*-axes and *y*-axes are not reversed.

See also AcChart::SetFlipAxes method

AcChart::GetBaseLayer method

Returns a reference to the base layer of a chart.

Syntax Function GetBaseLayer( ) As AcChartLayer

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

Returns A reference to the base layer of the chart.

See also AcChart::GetLayer method
AcChart::GetOverlayLayer method
AcChart::GetStudyLayer method

Class AcChart
Class AcChartLayer

AcChart::GetBorderStyle method

Returns the style of the border around a chart. To change the border around a chart, call this method to get the default settings.

Syntax Function GetBorderStyle( ) As AcDrawingBorderStyle

Returns The style of the border around the chart.

Example The following example overrides a chart's CustomizeChart( ) method to create a border around the chart, depending on the value of a Boolean parameter. GetBorderStyle( ) gets the default settings so that only the border style's Pen member needs to be changed.

```
Sub CustomizeChart( )
   If parmAddABorder Then
      Dim borderStyle As AcDrawingBorderStyle
      borderStyle = GetBorderStyle( )
      borderStyle.Pen = DrawingLinePenSolid
      SetBorderStyle( borderStyle )
   End If
End Sub
```

See also AcChart::SetBorderStyle method
AcChartLayer::GetPlotAreaBorderStyle method
AcDrawingBorderStyle

AcChart::GetChartDrawingPlane method

Returns a reference to the chart drawing plane of a chart.

Syntax Function GetChartDrawingPlane( ) As AcDrawingChartPlane

Returns A reference to the chart drawing plane of the chart.

Example For an example of how to use this method, see the example for the AcDrawingChartPlane class.

See also Class AcDrawingChartPlane

AcChart::GetFillStyle method

Returns the background fill style for a chart. To change the background of a chart, call GetFillStyle( ) to get the default settings.

Syntax Function GetFillStyle( ) As AcDrawingFillStyle

Returns The background fill style for the chart.

Example The following example overrides a chart's CustomizeChart( ) method to create a patterned background, depending on the value of a Boolean parameter. GetFillStyle( ) is used to get the default settings so that only the fill style's Pattern member needs to be changed.

```
Sub CustomizeChart( )
   If parmAddBackgroundPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = GetFillStyle( )
      fillStyle.Pattern = DrawingFillPattern05Percent
      SetFillStyle( fillStyle )
   End If
End Sub
```

See also AcChart::SetFillStyle method
AcChartLayer::GetPlotAreaFillStyle method
AcDrawingFillStyle

AcChart::GetHyperchartLink method

Override the GetHyperchart( ) method to provide the hyperlink URL for a given layer, category, and series within a chart. In most cases, you can use the hyperlink expressions in Advanced Chart Options to define hyperlinks. If you need more advanced control over hyperlinking, you can override a chart's GetHyperchartLink( ) method to compute the hyperlink URL for a given layer, category, and series.

Hyperchart links are enabled automatically if any of the category, series, or point link expressions are set in Advanced Chart Options. If you override a chart's GetHyperchartLink( ) method to compute hyperchart links instead of using one of the hyperchart link expressions, hyperchart links are not enabled automatically. In this case, you must call EnableHyperchart( ) from the chart's CustomizeChart( ) method.

Syntax Function GetHyperchartLink(layerNumber As Integer, categoryNumber As Integer, seriesNumber As Integer) As String

Parameters **layerNumber**
The selected layer, specified as an index into the chart's list of layers. The first layer is number 1.

If layerNumber is less than 1, no layer was selected.

categoryNumber
The selected category, specified as an index into the selected layer's list of categories. The first category is number 1.

If the selected layer is a scatter chart, the layer has no categories. In this case, categoryNumber is the index of the selected point within the selected series. The first point in a series is number 1.

If categoryNumber is less than 1, no category or point was selected.

seriesNumber
The selected series, specified as an index into the selected layer's list of series. The first series is number 1.

If seriesNumber is less than 1, no series was selected.

Returns The hyperlink URL corresponding to the specified layer, category and series. An empty string if there is no valid hyperlink.

Example In the following example, a chart has one category for each sales region. The example overrides the GetHyperchartLink( ) method of the chart to link to different sales team web sites, depending on the target category.

```
Function GetHyperchartLink( layerNumber As Integer,
+  categoryNumber As Integer, seriesNumber As Integer ) As String
   If (layerNumber < 1) Or (categoryNumber < 1) Or (seriesNumber
      < 1) Then
      Exit Function
   End If
   Dim layer As AcChartLayer
   Set layer = GetLayer( layerNumber )
   Dim category As AcChartCategory
   Set category = layer.GetCategory( categoryNumber )
   Dim region As String
   region = category.GetKeyValue( )
   Select Case region
   Case "North"
      GetHyperchartLink = "http://www.detroit.icharts.com/sales"
   Case "East"
      GetHyperchartLink = "http://www.boston.icharts.com/
         salesgroup"
   Case "South"
      GetHyperchartLink = "http://www.houston.icharts.com/
         salesteam"
   Case "West"
      GetHyperchartLink = "http://www.portland.icharts.com/
         salesportal"
   Case Else
      ' Unknown region - no link.
      GetHyperchartLink = ""
   End Select
End Function
```

See also AcChart::CustomizeChart method
AcChart::EnableHyperchart method
AcChart::GetLayer method
AcChartLayer::GetCategory method
AcChartLayer::GetSeries method

Class AcChartCategory
Class AcChartLayer
Class AcChartSeries

AcChart::GetLayer method

Returns a reference to the specified layer of a chart. Where possible, use one of the following more specific methods to get a chart layer:

- GetBaseLayer( )
- GetOverlayLayer( )
- GetStudyLayer( )

Use GetLayer( ) in the following situations:

- To access all the layers in a chart within a loop
- To override a method that has a layer index parameter, such as GetHyperchartLink( )

To determine the number of layers in a chart, call the chart's GetNumberOfLayers( ) method.

Syntax Function GetLayer(index As Integer) As AcChartLayer

Parameter **index**
An index into the chart's list of layers. The first layer is index 1.

Returns A reference to the specified layer of the chart.

Example In the following example, all a chart's layers are bar chart layers. The example overrides the chart's CustomizeCategoriesAndSeries( ) method to adjust the gaps between categories in all its layers so that the total width of the bars in each category is the same in each layer.

```
Sub CustomizeCategoriesAndSeries(baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim numberOfLayers As Integer
   numberOfLayers = GetNumberOfLayers( )
   Dim layerIndex As Integer

   For layerIndex = 1 To numberOfLayers
      Dim layer As AcChartLayer
      Set layer = GetLayer( layerIndex )
      ' Adjust the gap between categories so that
      ' all layers' bars take up the same space.
      Dim gapRatio As Integer
      gapRatio = layer.GetNumberOfSeries( )
      ' The maximum permitted gap ratio is 5.
      If (gapRatio > 5) Then
         gapRatio = 5
      End If
      layer.SetCategoryGapRatio( gapRatio )
   Next layerIndex
End Sub
```

See also AcChart::GetBaseLayer method
AcChart::GetNumberOfLayers method
AcChart::GetOverlayLayer method
AcChart::GetStudyLayer method
Class AcChartLayer

AcChart::GetLegendBackgroundColor method

Returns the background color of a chart's legend.

Syntax Function GetLegendBackgroundColor( ) As AcColor

Returns The background color of the chart's legend.

See also AcChart::SetLegendBackgroundColor method

AcChart::GetLegendBorderStyle method

Returns the style of the border around a chart's legend. To change the border around a chart's legend, call GetLegendBorderStyle( ) to get the default settings.

Syntax Function GetLegendBorderStyle( ) As AcDrawingBorderStyle

Returns The style of the border around the chart.

Example The following example overrides a chart's CustomizeChart( ) method to change the color of the border around the chart's legend to a value specified by a parameter. GetLegendBorderStyle( ) is used to get the default settings so that only the border style's Color member needs to be changed.

```
Sub CustomizeChart( )
   Dim legendBorderStyle As AcDrawingBorderStyle
   legendBorderStyle = GetLegendBorderStyle( )
   legendBorderStyle.Color = parmLegendBorderColor
   SetLegendBorderStyle( legendBorderStyle )
End Sub
```

See also AcChart::SetLegendBorderStyle method
AcDrawingBorderStyle

AcChart::GetLegendFont method

Returns the font used for a chart's legend. The GetLegendFont( ) method returns the font used for a chart's legend.

Syntax Function GetLegendFont( ) As AcFont

Returns The font used for the chart's legend.

Example The following example overrides a chart's CustomizeChart( ) method to make the chart's legend bold, depending on the value of a Boolean parameter. GetLegendFont( ) is used to get the default font so that only the Bold member needs to be changed.

```
Sub CustomizeChart( )
   If parmBoldLegend Then
      Dim legendFont As AcFont
      legendFont = GetLegendFont( )
      legendFont.Bold = True
      SetLegendFont( legendFont )
   End If
End Sub
```

See also AcChart::SetLegendFont method
AcFont

AcChart::GetLegendPlacement method

Returns the placement of a chart's legend relative to the chart.

Syntax Function GetLegendPlacement( ) As AcChartLegendPlacement

Returns The placement of the legend in the chart.

See also AcChartLegendPlacement
AcChart::SetLegendPlacement method

AcChart::GetNumberOfLayers method

Returns the number of layers in a chart. Use this method when you want to access all the layers of a chart within a loop, using the GetLayer( ) method.

Syntax Function GetNumberOfLayers( ) As Integer

Returns The number of layers in the chart.

Example For an example of how to use this method, see the example for the GetLayer( ) method.

See also AcChart::GetLayer method
AcChart::GetNumberOfStudyLayers method
AcChart::HasOverlayLayer method

AcChart::GetNumberOfStudyLayers method

Returns the number of study layers in a chart. Use this method:

- To test whether a chart has any study layers
- To access all the study layers of a chart within a loop, using the GetStudyLayer( ) method

Syntax Function GetNumberOfStudyLayers( ) As Integer

Returns The number of study layers in the chart.
Zero if the chart has no study layers.

See also AcChart::GetNumberOfLayers method
AcChart::GetStudyLayer method
AcChart::HasOverlayLayer method

AcChart::GetOverlayLayer method

Returns a reference to the overlay layer of a chart.

Syntax Function GetOverlayLayer( ) As AcChartLayer

Returns A reference to the overlay layer of the chart.
Nothing if the chart has no overlay layer.

See also AcChart::GetBaseLayer method
AcChart::GetLayer method
AcChart::GetStudyLayer method
AcChart::HasOverlayLayer method
Class AcChartLayer

AcChart::GetStudyLayer method

Returns a reference to the specified study layer of a chart. To determine the number of study layers in a chart, call the chart's GetNumberOfStudyLayers( ) method.

Syntax Function GetStudyLayer(index As Integer) As AcChartLayer

Parameter **index**
An index into the chart's list of study layers. The first study layer is index 1.

Returns A reference to the specified study layer of the chart.

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

See also AcChart::GetBaseLayer method
AcChart::GetLayer method
AcChart::GetNumberOfStudyLayers method

AcChart::GetOverlayLayer method
Class AcChart
Class AcChartLayer

AcChart::GetTitleStyle method

Returns the style of a chart's title. To change the style of a chart's title, call GetTitleStyle( ) to get the default settings.

Syntax Function GetTitleStyle( ) As AcDrawingTextStyle

Returns The chart's title style.

Example The following example overrides a chart's CustomizeChart( ) method to make the chart's title italic, depending on the value of a Boolean parameter. GetTitleStyle( ) gets the default settings so that only the title style's Font member needs to be changed.

```
Sub CustomizeChart( )
   If parmItalicTitle Then
      Dim titleStyle As AcDrawingTextStyle
      titleStyle = GetTitleStyle( )
      titleStyle.Font.Italic = True
      SetTitleStyle( titleStyle )
   End If
End Sub
```

See also AcChart::SetTitleStyle method
AcDrawingTextStyle

AcChart::GetTitleText method

Returns the text of a chart's title.

Syntax Function GetTitleText( ) As String

Returns The chart's title text.

See also AcChart::GetTitleText method

AcChart::HasOverlayLayer method

Determines whether a chart has an overlay layer.

Syntax Function GetTitleText( ) As String

Returns True if the chart has an overlay layer.
False if the chart does not have an overlay layer.

See also AcChart::DisableOverlayLayer method
AcChart::EnableOverlayLayer method

AcChart::GetOverlayLayer method

AcChart::IsHyperchart method

Determines whether a chart has hyperchart links.

Syntax Function IsHyperchart( ) As Boolean

Returns True if the chart has hyperchart links.
False if the chart does not have hyperchart links.

See also AcChart::DisableHyperchart method
AcChart::EnableHyperchart method

AcChart::IsThreeD method

Determines whether a chart has a three-dimensional appearance.

Syntax Function IsThreeD( ) As Boolean

Returns True if the chart has a three-dimensional appearance.
False if the chart has a two-dimensional appearance.

See also AcChart::SetThreeD method

AcChart::Localize method

Override the Localize( ) method to localize a chart at view time. You cannot create new persistent objects in this method. For example, you cannot add a new series to a chart layer from this method.

Changes you make in this method apply only at the instant a chart is being rendered to a viewable image and are not persistent.

Syntax Sub Localize(baseLayer As AcChartLayer, overlayLayer As AcChartLayer,
    studyLayers( ) As AcChartLayer)

Parameters **baseLayer**
A reference to the chart's base layer object.

overlayLayer
A reference to the chart's overlay layer object. Nothing if the chart has no overlay layer.

studyLayers( )
An array of references to the chart's study layer objects. To find out how many study layers there are in a chart, call the chart's GetNumberOfStudyLayers( ) method.

Example The following example overrides a chart's Localize( ) method to translate labels on the chart's *x*-axis into French at view time if the viewing locale is fr_FR:

```
Sub Localize( baseLayer As AcChartLayer, overlayLayer As
   AcChartLayer, studyLayers( ) As AcChartLayer )
   If (GetLocaleName( ) = "fr_FR") Then
      Dim xAxis As AcChartAxis
      Set xAxis = baseLayer.GetXAxis( )
      Dim numberOfLabels As Integer
      numberOfLabels = xAxis.GetNumberOfLabels

      Dim labelIndex As Integer
      For labelIndex = 1 To numberOfLabels
         Select Case xAxis.GetLabelValue( labelIndex )
         Case "North"
            xAxis.SetLabelValue( labelIndex, "Nord" )
         Case "South"
            xAxis.SetLabelValue( labelIndex, "Sud" )
         Case "East"
            xAxis.SetLabelValue( labelIndex, "Est" )
         Case "West"
            xAxis.SetLabelValue( labelIndex, "Ouest" )
         End Select
      Next labelIndex
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::CustomizeCategoriesAndSeries method
AcChart::CustomizeChart method
AcChart::CustomizeLayers method
AcChart::CustomizeSeriesStyles method
AcChart::GetNumberOfStudyLayers method
Class AcChartLayer

AcChart::MakeAxes method

Call the MakeAxes( ) method to create all the axes in a chart. If you create a chart dynamically and do not use the standard chart building mechanism, you must call the MakeAxes( ) method to create all the axes in the chart before calling the chart's ComputeScales( ) method.

Syntax Sub MakeAxes( )

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

See also Class AcChart

AcChart::MakeLayers method

Call the MakeLayers( ) method to create all the layers in a chart. If you create a chart dynamically and do not use the standard chart building mechanism, you must call MakeLayers( ) to create all the layers.

Syntax Sub MakeLayers( )

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

See also Class AcChart

AcChart::SetBackgroundColor method

Call the SetBackgroundColor( ) method to set the background color of a chart. This method sets a chart's fill style to a single solid color and sets a chart's fill style members as follows:

- The Color1 member is set to the specified background color.
- The Color2 member is not affected.
- The Pattern member is set to DrawingFillPatternSolid.

The recommended method from which to call SetBackgroundColor( ) is a chart's CustomizeChart( ) method.

Syntax Sub SetBackgroundColor(backgroundColor As AcColor)

Parameter **backgroundColor**
The background color for the chart.

Example The following example overrides a chart's CustomizeChart( ) method to set the background color to the value of a parameter:

```
Sub CustomizeChart( )
   SetBackgroundColor( parmChartBackgroundColor )
End Sub
```

See also AcChart::CustomizeChart method
AcChart::SetFillStyle method
AcChartLayer::SetPlotAreaBackgroundColor method
AcDrawingFillStyle

AcChart::SetBorderStyle method

Call the SetBorderStyle( ) method to set the style of the border around a chart. The recommended method from which to call SetBorderStyle( ) is a chart's CustomizeChart( ) method.

Syntax Sub SetBorderStyle(borderStyle As AcDrawingBorderStyle)

Parameter **borderStyle**
The border style for the chart.

Example The following example overrides a chart's CustomizeChart( ) method to create a border around the chart, depending on the value of a Boolean parameter. GetBorderStyle( ) gets the default settings so that only the border style's Pen member needs to be changed.

```
Sub CustomizeChart( )
   If parmAddABorder Then
      Dim borderStyle As AcDrawingBorderStyle
      borderStyle = GetBorderStyle( )
      borderStyle.Pen = DrawingLinePenSolid
      SetBorderStyle( borderStyle )
   End If
End Sub
```

See also AcChart::CustomizeChart method
AcChart::GetBorderStyle method
AcChartLayer::SetPlotAreaBorderStyle method
AcDrawingBorderStyle

AcChart::SetFillStyle method

Call SetFillStyle( ) to set the background fill style for a chart. The recommended method from which to call SetFillStyle( ) is a chart's CustomizeChart( ) method.

Syntax Sub SetFillStyle(fillStyle As AcDrawingFillStyle)

Parameter **fillStyle**
The background fill style for the chart.

Example The following example overrides a chart's CustomizeChart( ) method to create a patterned background. GetFillStyle( ) gets the default settings so that only the fill style's Pattern member needs to be changed.

```
Sub CustomizeChart( )
   If parmAddBackgroundPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = GetFillStyle( )
      fillStyle.Pattern = DrawingFillPattern05Percent
      SetFillStyle( fillStyle )
   End If
End Sub
```

See also AcChart::CustomizeChart method
AcChart::GetFillStyle method
AcChart::SetBackgroundColor method
AcChartLayer::SetPlotAreaFillStyle method
AcDrawingFillStyle

AcChart::SetFlipAxes method

Call the SetFlipAxes( ) method to specify whether a chart's *x*-axis and *y*-axis are reversed. The recommended method from which to call SetFlipAxes( ) is a chart's CustomizeChart( ) method.

You can use reversed axes only on a chart whose base layer is a bar chart. You cannot use reversed axes on a chart that has an overlay layer or a study layer.

Syntax Sub SetFlipAxes(flipAxes As Boolean)

Parameter **flipAxes**
True reverses the chart's *x*-axis and *y*-axis.

False does not reverse the chart's *x*-axis and *y*-axis.

Example The following example overrides a chart's CustomizeChart( ) method to reverse the chart's axes, depending on the value of a Boolean parameter:

```
Sub CustomizeChart( )
   SetFlipAxes( parmFlipAxes )
End Sub
```

See also AcChart::CustomizeChart method
AcChart::FlipAxes method

AcChart::SetLegendBackgroundColor method

Call the SetLegendBackgroundColor( ) method to set the background color of a chart's legend. The recommended method from which to call SetLegendBackgroundColor( ) is a chart's CustomizeChart( ) method.

Syntax Sub SetLegendBackgroundColor(legendBackgroundColor As AcColor)

Parameter **legendBackgroundColor**
The background color for the chart's legend.

Example The following example overrides a chart's CustomizeChart( ) method to set the background color of the chart's legend to the value of a parameter:

```
Sub CustomizeChart( )
   SetLegendBackgroundColor( parmLegendBackgroundColor )
End Sub
```

See also AcChart::CustomizeChart method
AcChart::GetLegendBackgroundColor method

AcChart::SetLegendBorderStyle method

Call the SetLegendBorderStyle( ) method to set the style of the border around a chart's legend. The recommended method from which to call SetLegendBorderStyle( ) is a chart's CustomizeChart( ) method.

Syntax Sub SetLegendBorderStyle(legendBorderStyle As AcDrawingBorderStyle)

Parameter **legendBorderStyle**
The border style for the chart's legend.

Example The following example overrides a chart's CustomizeChart( ) method to change the color of the border around the chart's legend to a value specified by a parameter. GetLegendBorderStyle( ) gets the default settings so that only the border style's Color member needs to be changed.

```
Sub CustomizeChart( )
   Dim legendBorderStyle As AcDrawingBorderStyle
   legendBorderStyle = GetLegendBorderStyle( )
   legendBorderStyle.Color = parmLegendBorderColor
   SetLegendBorderStyle( legendBorderStyle )
End Sub
```

See also AcChart::CustomizeChart method
AcChart::GetLegendBorderStyle method
AcDrawingBorderStyle

AcChart::SetLegendFont method

Call the SetLegendFont( ) method to set the font for a chart's legend. The recommended method from which to call SetLegendFont( ) is a chart's CustomizeChart( ) method. To change the legend font at view time, call this method from a chart's Localize( ) method.

Syntax Sub SetLegendFont(legendFont As AcFont)

Parameter **legendFont**
The font to be used for the chart's legend.

Example The following example overrides a chart's CustomizeChart( ) method to make the chart's legend bold, based on the value of a Boolean parameter. GetLegendFont( ) gets the default font so that only the Bold member needs to be changed.

```
Sub CustomizeChart( )
   If parmBoldLegend Then
      Dim legendFont As AcFont
      legendFont = GetLegendFont( )
      legendFont.Bold = True
      SetLegendFont( legendFont )
   End If
End Sub
```

See also AcChart::CustomizeChart method
AcChart::GetLegendFont method
AcChart::Localize method
AcFont

AcChart::SetLegendPlacement method

Call the SetLegendPlacement( ) method to set the placement of a chart's legend. The recommended method from which to call SetLegendPlacement( ) is a chart's CustomizeChart( ) method.

Syntax Sub SetLegendPlacement(legendPlacement As AcChartLegendPlacement)

Parameter **legendPlacement**
The placement of the chart's legend.

Example The following example overrides a chart's CustomizeChart( ) method to hide the chart's legend, depending on the value of a Boolean parameter:

```
Sub CustomizeChart( )
   If parmHideLegend Then
      SetLegendPlacement( ChartLegendPlacementNone )
   End If
End Sub
```

See also AcChart::CustomizeChart method
AcChart::GetLegendPlacement method
AcChartLegendPlacement

AcChart::SetMatchBaseAndOverlayScales method

Call the SetMatchBaseAndOverlayScales( ) method to specify whether the base and overlay *y*-axis scales of a chart are forced to be identical. If you call this method with matchBaseAndOverlayScales set to True, the *y*-axis settings of the overlay layer will be copied from the base layer's *y*-axis.

You can call this method only from a chart's CustomizeChart( ) method.

Syntax Sub SetMatchBaseAndOverlayScales(matchBaseAndOverlayScales
   As Boolean)

Parameter **matchBaseAndOverlayScales**
True forces the chart's base and overlay *y*-axis scales to be identical. False allows the chart's base and overlay *y*-axis scales to be independent of each other.

Example The following example overrides a chart's CustomizeChart( ) method to force the chart's base and overlay *y*-axis scales to be identical:

```
Sub CustomizeChart( )
   SetMatchBaseAndOverlayScales( True )
End Sub
```

See also AcChart::BaseAndOverlayScalesAreMatched method
AcChart::CustomizeChart method

AcChart::SetStatus method

Call the SetStatus( ) method to set the status of a chart you are creating dynamically. The chart status is part of a safety mechanism that checks whether methods are being called in the correct sequence to produce a valid chart. If you create a chart dynamically and do not use the standard chart building mechanism, you must call the SetStatus( ) method to set the status as you create and populate the chart.

Call SetStatus( ) with status set to ChartStatusBuilding after you call the chart's StartLayers( ) method and before you call the chart's Finish( ) method.

Syntax Sub SetStatus(status As AcChartStatus)

Parameter **status**
The new status for the chart.

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

See also AcChartStatus
Class AcChart

AcChart::SetThreeD method

Call the SetThreeD( ) method to specify whether a chart displays with a three-dimensional appearance. The recommended method from which to call SetThreeD( ) is a chart's CustomizeChart( ) method.

The following chart types do not support three-dimensional appearance:

- Scatter
- Step
- Stock

You cannot use three-dimensional appearance on a chart that has an overlay layer or a study layer.

If you attempt to set three-dimensional appearance on a chart that has a layer that does not support three-dimensional appearance, SetThreeD( ) throws a run-time error. For example, selecting a three-dimensional appearance on a chart with a step chart layer throws a run-time error.

Syntax Sub SetThreeD(threeD As Boolean)

Parameter **threeD**
True displays the chart with a three-dimensional appearance. False displays the chart with a two-dimensional appearance.

Example The following example overrides a chart's CustomizeChart( ) method to select between two- and three-dimensional appearances, depending on the value of a Boolean parameter:

```
Sub CustomizeChart( )
   SetThreeD( parmThreeD )
End Sub
```

See also AcChart::CustomizeChart method
AcChart::IsThreeD method

AcChart::SetTitleStyle method

Call the SetTitleStyle( ) method to set the style of a chart's title. The recommended method from which to call SetTitleStyle( ) is a chart's CustomizeChart( ) method.

To change the title style at view time, call SetTitleStyle( ) from a chart's Localize( ) method.

Syntax Sub SetTitleStyle(titleStyle As AcDrawingTextStyle)

Parameter **titleStyle**
The style for the chart's title.

Example The following example overrides a chart's CustomizeChart( ) method to make the chart's title italic, depending on the value of a Boolean parameter. GetTitleStyle( ) retrieves the default settings so that only the title style's Font member needs to be changed.

```
Sub CustomizeChart( )
   If parmItalicTitle Then
      Dim titleStyle As AcDrawingTextStyle
      titleStyle = GetTitleStyle( )
      titleStyle.Font.Italic = True
      SetTitleStyle( titleStyle )
   End If
End Sub
```

See also AcChart::CustomizeChart method
AcChart::GetTitleStyle method
AcChart::Localize method
AcDrawingTextStyle

AcChart::SetTitleText method

Call the SetTitleText( ) method to set the text of a chart's title. The recommended methods from which to call SetTitleText( ) are:

- A chart's CustomizeChart( ) method
- A chart's OnRow( ) method

To change the title text at view time, call SetTitleText( ) from a chart's Localize( ) method.

Syntax Sub SetTitleText(titleText As String)

Parameter **titleText**
The text of the chart's title. Set this parameter to "" if you do not want a title.

Examples The following example overrides a chart's OnRow( ) method to set the chart's title using a value from the data row:

```
Sub OnRow( row As AcDataRow )
   ' Only look at the first row.
   If (GetRowCount( ) = 1) Then
      SetTitleText( "Credit Rank " & GetValue( row,
         "customers_creditrank" ) )
   End If
End Sub
```

For another example of how to use SetTitleText( ), see the dynamic chart example for the AcChart class.

See also AcChart::CustomizeChart method
AcChart::GetTitleText method
AcChart::Localize method
AcReportComponent::OnRow method

AcChart::StartEmpty method

Call StartEmpty( ) to initialize a chart that you are creating dynamically. If you call this method, you must not call the chart's Start( ) method.

Syntax Sub StartEmpty( )

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

See also Class AcChart

AcChart::StartLayers method

Call the StartLayers( ) method to initialize the layers of a chart that you are creating dynamically. If you create a chart dynamically and do not use the standard chart building mechanism, you must call the StartLayers( ) method to initialize the chart's layers.

Syntax Sub StartLayers( )

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

See also Class AcChart

Class AcChartAxis

An axis within a chart layer. Figure 7-6 shows the class hierarchy of AcChartAxis.

Figure 7-6 AcChartAxis

Description Use the AcChartAxis class to represent a single axis within a chart layer. Do not create AcChartAxis objects explicitly from your own code. Instead, AcChartLayer objects create AcChartAxis objects automatically as necessary to build complete charts.

Use the methods of AcChartLayer to access a chart layer's axes. You can manipulate the appearance of a chart by calling methods on the chart's axes.

All types of base chart layers except pie chart layers have an *x*-axis and a *y*-axis. Pie chart layers do not have axes. Overlay and study chart layers except pie chart layers have *y*-axes only. Overlay and study chart layers do not have their own *x*-axes. Instead, they share the base chart layer's *x*-axis.

A chart axis must be either a category scale axis or a value scale axis.

About category scale axes

A category scale axis shows categories used to group multiple values. The *x*-axis of a bar chart layer is a category scale axis. A category scale axis simply shows all the categories in the chart data.

Category scale axes do not have lower and upper bounds. Points plotted against a category scale axis always line up exactly with the categories on that axis.

About value scale axes

A value scale axis shows a range of values. The *y*-axis of a bar chart layer is a value scale axis. Value scale axes have lower and upper bounds. Points plotted against a value scale axis do not have to line up with the ticks on that axis.

A value scale axis uses the values plotted against it to compute:

- The lower and upper bounds of the axis
- The interval between ticks and the number of ticks
- The label values

About the origin

The origin of a value scale axis determines how points in filled chart types such as area, bar, and step are drawn relative to the axis origin. For example, consider a bar chart layer's *y*-axis whose lower bound is 55 and whose upper bound is 105:

- If the axis's origin is 0, a point whose value is 75 is drawn as a bar between 55 and 75.
- If the axis's origin is 0, a point whose value is 50 is drawn as a bar between 75 and 105.

The origin of a value scale axis is calculated as follows:

- If series in the chart layer are not stacked, the origin is always the axis value where the opposite axis crosses.
- If series in the chart layer are stacked, the origin is the axis value that is closest to zero.

About outer margins

The upper bound of a value scale axis is normally slightly higher than the highest point plotted against that axis. The space between the highest point and the upper bound of the axis is the outer margin.

- A value scale axis that includes values below the value where the opposite axis crosses has an outer margin between its lower bound and the lowest point. This means that some value scale axes have two outer margins, one at the lower bound and one at the upper bound.
- A value scale axis that does not have any values above the value where the opposite axis crosses does not have an outer margin between its upper bound and the highest point.

About inner margins

By default, when all the values plotted against a value scale axis fall within a certain percentage of the highest value, the axis will not include zero. In this situation, the lower bound of the axis is normally slightly lower than the lowest point plotted against the axis. The distance between the lowest point and the lower bound of the axis is the inner margin.

- A value scale axis that includes zero does not have an inner margin.
- Value scale axes never have more than one inner margin.
- If the opposite axis crosses a value scale axis at its upper bound, the axis's inner margin will be between its upper bound and the highest point.

Example For an example of how to use this class to build a chart dynamically, see the dynamic chart example for the AcChart class.

See also Class AcChart
Class AcChartCategory
Class AcChartGridLine
Class AcChartLayer
Class AcChartPoint

Class AcChartPointStyle
Class AcChartSeries
Class AcChartSeriesStyle
Class AcChartTrendline

Methods for Class AcChartAxis

Methods defined in Class AcChartAxis

AddGridLine, ClearMajorTickInterval, ClearMaximumValue, ClearMinimumValue, ClearOtherAxisCrossesAt, ComputeScale, ForceMajorTickCount, GetAxisLetter, GetAxisLetterText, GetDataType, GetDefaultRangeRatio, GetGridLine, GetInnerMarginRatio, GetLabelFormat, GetLabelPlacement, GetLabelStyle, GetLabelText, GetLabelValue, GetLayer, GetLineStyle, GetMajorGridLineStyle, GetMajorTickCalculation, GetMajorTickCount, GetMajorTickInterval, GetMajorTickPlacement, GetMaximumDataValue, GetMaximumTrendlineValue, GetMaximumValue, GetMinimumDataValue, GetMinimumTrendlineValue, GetMinimumValue, GetMinorGridLineStyle, GetMinorTickCount, GetMinorTickPlacement, GetNoZeroRatio, GetNumberOfGridLines, GetNumberOfLabels, GetOriginValue, GetOtherAxisCrossesAt, GetOtherAxisPlacement, GetOuterMarginRatio, GetTitleStyle, GetTitleText, HasFixedMaximum, HasFixedMinimum, IgnoreTrendlines, InsertGridLine, IsCategoryScale, IsValueScale, IsXAxis, IsYAxis, IsZAxis, PlotCategoriesBetweenTicks, ResetMajorTickInterval, SetDataType, SetDefaultRangeRatio, SetForceMajorTickCount, SetIgnoreTrendlines, SetInnerMarginRatio, SetLabelFormat, SetLabelPlacement, SetLabelStyle, SetLabelValue, SetLineStyle, SetMajorGridLineStyle, SetMajorTickCalculation, SetMajorTickCount, SetMajorTickInterval, SetMajorTickPlacement, SetMaximumDataValue, SetMaximumValue, SetMinimumDataValue, SetMinimumValue, SetMinorGridLineStyle, SetMinorTickCount, SetMinorTickPlacement, SetNoZeroRatio, SetOtherAxisCrossesAt, SetOtherAxisPlacement, SetOuterMarginRatio, SetPlotCategoriesBetweenTicks, SetTitleStyle, SetTitleText

AcChartAxis::AddGridLine method

Call this method to add a grid line to the end of a chart axis's list of grid lines.

You can call this method only from:

- A chart's AdjustChart( ) method
- A chart's DrawOnChart( ) method
- Code that creates a chart dynamically

Syntax Function AddGridLine(value As Variant) As AcChartGridLine

Parameter **value**
The axis value at which the grid line is drawn. If the axis is a category scale axis, the first tick on the axis has the value 0, the second tick has the value 1, and so on.

Returns A handle to the new grid line object.

Example For an example of how to use this method, see the example for the AcChartGridLine class.

See also AcChart::AdjustChart method
AcChart::DrawOnChart method
AcChartAxis::GetGridLine method
AcChartAxis::GetNumberOfGridLines method
AcChartAxis::InsertGridLine method
Class AcChartGridLine

AcChartAxis::ClearMajorTickInterval method

Call the ClearMajorTickInterval( ) method to reset the major tick interval of a chart axis to its default setting and cause the axis to compute the major tick interval automatically.

You can call this method only on a value scale axis. You can call this method only from:

- A chart's CustomizeAxes( ) method
- A chart's AdjustChart( ) method

If you call ClearMajorTickInterval( ) from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub ClearMajorTickInterval( )

Example In the following example, you defined the *y*-axis of a chart's base layer in Chart Builder to have a fixed major tick interval. The example overrides the chart's CustomizeAxes( ) method to cancel the fixed interval, depending on the value of a Boolean parameter.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer )
   If parmAutoYAxisMajorInterval Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.ClearMajorTickInterval( )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ComputeScale method

AcChartAxis::GetMajorTickCalculation method
AcChartAxis::GetMajorTickInterval method
AcChartAxis::SetMajorTickCalculation method
AcChartAxis::SetMajorTickInterval method

AcChartAxis::ClearMaximumValue method

Call the ClearMaximumValue( ) method to remove a fixed maximum value from a chart axis.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub ClearMaximumValue( )

Example In the following example, you defined the *y*-axis of a chart's base layer in Chart Builder to have a fixed maximum value. The example overrides the chart's CustomizeAxes( ) method to cancel the fixed maximum value, depending on the value of a Boolean parameter.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAutoYAxisMaximum Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.ClearMaximumValue( )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ClearMinimumValue method
AcChartAxis::ComputeScale method
AcChartAxis::GetMaximumValue method
AcChartAxis::HasFixedMaximum method
AcChartAxis::SetMaximumValue method

AcChartAxis::ClearMinimumValue method

Call the ClearMinimumValue( ) method to remove a fixed minimum value from a chart axis. You can call this method only on a value scale axis. You can call this method only from:

- A chart's CustomizeAxes( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's X AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub ClearMinimumValue( )

Example In the following example, the *y*-axis of a chart's base layer was defined in Chart Builder to have a fixed minimum value. The example overrides the chart's CustomizeAxes( ) method to cancel the fixed minimum value, depending on the value of a Boolean parameter.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAutoYAxisMinimum Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.ClearMinimumValue( )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ClearMaximumValue method
AcChartAxis::ComputeScale method
AcChartAxis::GetMaximumValue method
AcChartAxis::HasFixedMinimum method
AcChartAxis::SetMinimumValue method

AcChartAxis::ClearOtherAxisCrossesAt method

Call the ClearOtherAxisCrossesAt( ) method to remove a fixed axis crossing point from a chart axis and cause the axis to compute the axis crossing point automatically.

You can call this method only on a value scale axis. You can call this method only from:

- A chart's CustomizeAxes( ) method
- A chart's AdjustChart( ) method

If you call ClearOtherAxisCrossesAt( ) from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub ClearOtherAxisCrossesAt( )

Example In the following example, you defined the *x*-axis of a chart's base layer in Chart Builder to cross the base layer's *y*-axis at a fixed y value. The example overrides the chart's CustomizeAxes( ) method to cancel the fixed axis crossing, depending on the value of a Boolean parameter.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAutoAxisCrossing Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.ClearOtherAxisCrossesAt( )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ComputeScale method
AcChartAxis::GetOtherAxisCrossesAt method
AcChartAxis::GetOtherAxisPlacement method
AcChartAxis::SetOtherAxisCrossesAt method
AcChartAxis::SetOtherAxisPlacement method

AcChartAxis::ComputeScale method

Call the ComputeScale( ) method to compute the scale for a chart axis. If you modify the scaling settings of a chart axis programmatically after the axis scale has been computed, you must call the ComputeScale( ) method to recompute the axis scale.

You can call this method only on a value scale axis.

You can call this method only from a chart's AdjustChart( ) method.

Syntax Sub ComputeScale( )

Example The following example adjusts the upper bound of a study layer's *y*-axis so that it is at least 100. Because this adjustment relies on the automatically computed upper bound, it can only be made in AdjustChart( ).

```
Sub AdjustChart( baseLayer As AcChartLayer,
+  overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = studyLayers(1).GetYAxis( )
   If (yAxis.GetMaximumValue( ) < 100) Then
      yAxis.SetMaximumValue( 100 )
      ' Recompute the ticks and labels.
      yAxis.ComputeScale( )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::ComputeScales method

AcChartAxis::ForceMajorTickCount method

Determines whether the number of major ticks on a chart axis is forced to be a specific value. You can call this method only on a value scale axis.

Syntax Function ForceMajorTickCount( ) As Boolean

Returns True if the number of major ticks on the chart axis is forced to be a specific value. False if the number of major ticks on the chart axis is computed from data plotted on the axis.

See also AcChartAxis::SetForceMajorTickCount method
AcChartAxis::SetMajorTickCount method

AcChartAxis::GetAxisLetter method

Returns an axis letter value that indicates which axis a chart axis is.

Syntax Function GetAxisLetter( ) As AcChartAxisLetter

Returns An axis letter value that indicates which axis the chart axis is.

See also AcChartAxis::GetAxisLetterText method
AcChartAxisLetter

AcChartAxis::GetAxisLetterText method

Returns a string that indicates whether the axis is an *x*-axis, a *y*-axis, or a *z*-axis.

Syntax Function GetAxisLetterText( ) As AcChartAxisLetter

Returns X if the axis is an *x*-axis.
Y if the axis is a *y*-axis.
Z if the axis is a *z*-axis.

See also AcChartAxis::GetAxisLetter method

AcChartAxis::GetDataType method

Returns the data type of the scale of a chart axis. You can call this method only on a value scale axis.

Syntax Function GetDataType( ) As AcDataType

Returns The data type of the scale of the chart axis. One of the following values:

- DataTypeDateTime
- DataTypeNumber

See also AcChartAxis::SetDataType method
AcDataType

AcChartAxis::GetDefaultRangeRatio method

Returns the ratio used to compute the range of a chart axis when all the values plotted on the axis lie on the axis's origin. The axis's origin value is multiplied by the default range ratio to give a range. That range is subtracted from the value to get the lower bound of the axis and added to the value to get the upper bound of the axis. For example, if all the points plotted on a chart axis have the value 100, and the default range ratio is 0.1, the lower bound of the axis will be 90 and the upper bound of the axis will be 110.

You can call GetDefaultRangeRatio( ) only on a value scale axis.

Syntax Function GetDefaultRangeRatio( ) As Double

Returns The ratio used to compute the range of the chart axis when all the values plotted on the axis lie on the axis's origin.

See also AcChartAxis::SetDefaultRangeRatio method

AcChartAxis::GetGridLine method

Returns a reference to the specified grid line within a chart axis. To determine the the chart axis's number of grid lines, call the GetNumberOfGridLines( ) method on the chart axis.

You can call this method only from:

- A chart's AdjustChart( ) method
- A chart's DrawOnChart( ) method

Syntax Function GetGridLine(index As Integer) As AcChartGridLine

Parameter **index**
An index into the chart axis's list of grid lines. The first grid line is index 1.

Returns A reference to the specified grid line within the chart axis.

See also AcChart::AdjustChart method
AcChart::DrawOnChart method
AcChartAxis::AddGridLine method
AcChartAxis::GetNumberOfGridLines method
AcChartAxis::InsertGridLine method
Class AcChartGridLine

AcChartAxis::GetInnerMarginRatio method

Returns the minimum ratio between the inner margin on a chart axis and the total range of that axis. For example, if this method returns 0.25 for a bar chart layer's *y*-axis, the shortest bar will be at least 25% of the total height of the axis from the bottom of the axis.

You can call this method only on a value scale axis.

Syntax Function GetInnerMarginRatio( ) As Double

Returns The minimum ratio between the inner margin on the chart axis and the total range of the axis.

See also AcChartAxis::GetOuterMarginRatio method
AcChartAxis::SetInnerMarginRatio method

AcChartAxis::GetLabelFormat method

Returns the format pattern used to format labels on a chart axis. Category labels are used as category scale axis labels. The value that this method returns for a category scale axis is exactly the same as the value that the GetCategoryLabelFormat( ) method returns of the parent chart layer of that axis.

Syntax Function GetLabelFormat( ) As String

Returns The format pattern used to format labels on a chart axis.

See also AcChartAxis::SetLabelFormat method
AcChartLayer::GetCategoryLabelFormat method

AcChartAxis::GetLabelPlacement method

Returns the placement of labels on a chart axis.

Syntax Function GetLabelPlacement( ) As AcChartAxisLabelPlacement

Returns The placement of labels on a chart axis.

See also AcChartAxisLabelPlacement
AcChartAxis::SetLabelPlacement method

AcChartAxis::GetLabelStyle method

Returns the style for labels on a chart axis. To change the style of labels on a chart axis, call this method to get the default settings.

Syntax Function GetLabelStyle( ) As AcDrawingTextStyle

Returns The style for labels on a chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to make axis labels on the *x*-axis of the chart's base layer italic, depending on the value of a Boolean parameter. GetLabelStyle( ) retrieves the default settings so that only the title style's Font member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
```

```
   If parmItalicXLabels Then
      Dim xAxis As AcChartAxis
      Set xAxis = baseLayer.GetXAxis( )
      Dim labelStyle As AcDrawingTextStyle
      labelStyle = xAxis.GetLabelStyle( )
      labelStyle.Font.Italic = True
      xAxis.SetLabelStyle( labelStyle )
   End If
End Sub
```

See also AcChartAxis::SetLabelStyle method
AcDrawingTextStyle

AcChartAxis::GetLabelText method

Returns the formatted text of the specified label on a chart axis. To retrieve the number of labels on a chart axis, call the axis's GetNumberOfLabels( ) method. Category labels are used as category scale axis labels. The value that this method returns for a category scale axis is exactly the same as the value that the GetLabelText( ) method returns of the corresponding chart category.

Syntax Function GetLabelText(index As Integer) As String

Parameter **index**
An index into the axis's list of labels. The first label is index 1.

Returns The formatted text of the specified label on a chart axis.
String label values are returned unformatted.

See also AcChartAxis::GetLabelValue method
AcChartAxis::GetNumberOfLabels method
AcChartCategory::GetLabelText method

AcChartAxis::GetLabelValue method

Returns the value of the specified label on a chart axis. To retrieve the number of labels on a chart axis, call the axis's GetNumberOfLabels( ) method. Category labels are used as category scale axis labels. The value that this method returns for a category scale axis is exactly the same as the value that the GetLabelValue( ) method returns of the corresponding chart category.

Syntax Function GetLabelValue(index As Integer) As Variant

Parameter **index**
An index into the axis's list of labels. The first label is index 1.

Returns The value of the specified label on a chart axis.

See also AcChartAxis::GetNumberOfLabels method
AcChartAxis::SetLabelValue method

AcChartCategory::GetLabelValue method

AcChartAxis::GetLayer method

Returns a reference to the parent chart layer of a chart axis.

Syntax Function GetLayer( ) As AcChartLayer

Returns A reference to the parent chart layer of a chart axis.

See also Class AcChartLayer

AcChartAxis::GetLineStyle method

Returns the line style used to draw a chart axis. To change the style of a chart axis line, call this method to get the default settings.

Syntax Function GetLineStyle( ) As AcDrawingLineStyle

Returns The line style used to draw a chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to change the thickness of all axes in the chart, depending on the value of a Boolean parameter. GetLineStyle( ) gets the default settings so that only the line style's Width member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmHeavyAxes Then
      Dim numberOfLayers As Integer
      numberOfLayers = GetNumberOfLayers( )
      Dim layerIndex As Integer
      For layerIndex = 1 To numberOfLayers
         Dim layer As AcChartLayer
         Set layer = GetLayer( layerIndex )
         Dim lineStyle As AcDrawingLineStyle
         Dim axis As AcChartAxis
         Set axis = layer.GetXAxis( )
         If Not axis Is Nothing Then
            lineStyle = axis.GetLineStyle( )
            lineStyle.Width = 2 * OnePoint
            axis.SetLineStyle( lineStyle )
         End If
         Set axis = layer.GetYAxis( )
         If Not axis Is Nothing Then
            lineStyle = axis.GetLineStyle( )
            lineStyle.Width = 2 * OnePoint
            axis.SetLineStyle( lineStyle )
```

```
        End If
     Next layerIndex
   End If
End Sub
```

See also AcChartAxis::SetLineStyle method
AcDrawingLineStyle

AcChartAxis::GetMajorGridLineStyle method

Returns the line style used to draw grid lines for the major ticks on a chart axis. To change the style of grid lines, call this method to get the default settings.

Syntax Function GetMajorGridLineStyle( ) As AcDrawingLineStyle

Returns The line style used to draw grid lines for the major ticks on the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to set dotted major grid lines for the *y*-axis of the chart's base layer, depending on the value of a Boolean parameter. GetMajorGridLineStyle( ) retrieves the default settings so that only the line style's Pen member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmDottedMajorGrid Then
     Dim yAxis As AcChartAxis
     Set yAxis = layer.GetYAxis( )
     Dim lineStyle As AcDrawingLineStyle
     lineStyle = yAxis.GetMajorGridLineStyle( )
     lineStyle.Pen = DrawingLinePenDot
     yAxis.SetMajorGridLineStyle( lineStyle )
   End If
End Sub
```

See also AcChartAxis::GetMinorGridLineStyle method
AcChartAxis::SetMajorGridLineStyle method
AcDrawingLineStyle

AcChartAxis::GetMajorTickCalculation method

Returns the type of calculation used to compute major ticks on a chart axis.

You can call this method only on a value scale axis.

In some cases, the value that this method returns is altered automatically when the axis's scale is computed, as follows:

- If the axis has fixed lower and upper bounds, and the major tick count is forced, the interval is also forced. In this case, the major tick calculation changes to ChartTickCalculationAuto.

- If the axis has fixed lower and upper bounds, the major tick count is not forced, and the major tick calculation is ChartTickCalculationExactInterval, the calculation changes to ChartTickCalculationMinimumInterval. In such a case, forcing an exact interval might violate the maximum major tick count.

Syntax Function GetMajorTickCalculation( ) As AcChartTickCalculation

Returns The type of calculation used to compute major ticks on a chart axis.

See also AcChartAxis::GetMajorTickInterval method
AcChartAxis::SetMajorTickCalculation method
AcChartTickCalculation

AcChartAxis::GetMajorTickCount method

Returns the exact or maximum number of major ticks on a chart axis.

You can call this method only on a value scale axis.

Syntax Function GetMajorTickCount( ) As Integer

Returns If the major tick count is forced, the exact number of major ticks on a chart axis. If the major tick count is not forced, the maximum number of major ticks on a chart axis.

See also AcChartAxis::ForceMajorTickCount method
AcChartAxis::GetMinorTickCount method
AcChartAxis::SetForceMajorTickCount method
AcChartAxis::SetMajorTickCount method

AcChartAxis::GetMajorTickInterval method

Returns the exact or minimum interval between major ticks on a chart axis. You can call this method only on a value scale axis.

If you call GetMajorTickInterval( ) before a chart axis's scale has been computed and the chart axis major tick calculation is not ChartTickCalculationAuto, it returns the value used to compute the interval between major ticks on the axis.

Syntax Function GetMajorTickInterval( ) As Double

Returns The interval between major ticks on a chart axis.

See also AcChartAxis::GetMajorTickCalculation method
AcChartAxis::SetMajorTickInterval method

AcChartAxis::GetMajorTickPlacement method

Returns the placement of major ticks on a chart axis.

Syntax Function GetMajorTickPlacement( ) As AcChartTickPlacement

Returns The placement of major ticks on a chart axis.

See also AcChartAxis::GetMinorTickPlacement method
AcChartAxis::SetMajorTickPlacement method
AcChartTickPlacement

AcChartAxis::GetMaximumDataValue method

Returns the highest value plotted against a chart axis.

Syntax Function GetMaximumDataValue( ) As Variant

Returns The highest value plotted against a chart axis.

See also AcChartAxis::GetMaximumTrendlineValue method
AcChartAxis::GetMinimumDataValue method
AcChartAxis::SetMaximumValue method

AcChartAxis::GetMaximumTrendlineValue method

Returns the maximum y value of all the trendlines in a chart axis.

You can only call this method after the chart has computed its trendlines. You can call this method from the following methods:

- AcChart::CustomizeAxes( )
- AcChart::AdjustChart( )

Syntax Function GetMaximumTrendlineValue( ) As Variant

Returns The maximum y value of all the trendlines in the chart axis.
Null if the chart axis does not contain any trendlines.

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::ComputeMinMaxDataValues method
AcChartAxis::GetMaximumDataValue method
AcChartAxis::GetMinimumTrendlineValue method
AcChartLayer::GetMaximumTrendlineYValue method
Class AcChartTrendline

AcChartAxis::GetMaximumValue method

Returns the upper bound of a chart axis.

You can call this method only on a value scale axis.

Syntax Function GetMaximumValue( ) As Variant

Returns The upper bound of a chart axis.

See also AcChartAxis::GetMaximumDataValue method
AcChartAxis::GetMaximumTrendlineValue method
AcChartAxis::GetMinimumValue method
AcChartAxis::HasFixedMaximum method
AcChartAxis::SetMaximumValue method

AcChartAxis::GetMinimumDataValue method

Returns the lowest value plotted against a chart axis.

Syntax Function GetMinimumDataValue( ) As Variant

Returns The lowest value plotted against a chart axis.

See also AcChartAxis::GetMaximumDataValue method
AcChartAxis::SetMinimumDataValue method

AcChartAxis::GetMinimumTrendlineValue method

Returns the minimum y value of all the trendlines in a chart axis.

You can only call this method after the chart has computed its trendlines.

You can call this method from the following methods:

- AcChart::CustomizeAxes( )
- AcChart::AdjustChart( )

Syntax Function GetMinimumTrendlineValue( ) As Variant

Returns The minimum y value of all the trendlines in the chart axis.
Null if the chart axis does not contain any trendlines.

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::ComputeMinMaxDataValues method
AcChartAxis::GetMaximumTrendlineValue method
AcChartAxis::GetMinimumDataValue method
AcChartLayer::GetMinimumTrendlineYValue method
Class AcChartTrendline

AcChartAxis::GetMinimumValue method

Returns the lower bound of a chart axis.

You can call this method only on a value scale axis.

Syntax Function GetMinimumValue( ) As Variant

Returns The lower bound of a chart axis.

See also AcChartAxis::GetMaximumValue method
AcChartAxis::GetMinimumDataValue method
AcChartAxis::GetMinimumTrendlineValue method
AcChartAxis::HasFixedMinimum method
AcChartAxis::SetMinimumValue method

AcChartAxis::GetMinorGridLineStyle method

Returns the line style used to draw grid lines for the minor ticks on a chart axis. To change the style of grid lines, call this method to get the default settings.

Syntax Function GetMinorGridLineStyle( ) As AcDrawingLineStyle

Returns The line style used to draw grid lines for the minor ticks on a chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to set dotted minor grid lines for the *y*-axis of the chart's base layer, depending on the value of a Boolean parameter. GetMinorGridLineStyle( ) retrieves the default settings so that only the line style's Pen member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmDottedMinorGrid Then
      Dim yAxis As AcChartAxis
      Set yAxis = layer.GetYAxis( )
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = yAxis.GetMinorGridLineStyle( )
      lineStyle.Pen = DrawingLinePenDot
      yAxis.SetMinorGridLineStyle( lineStyle )
   End If
End Sub
```

See also AcChartAxis::GetMajorGridLineStyle method
AcChartAxis::SetMinorGridLineStyle method
AcDrawingLineStyle

AcChartAxis::GetMinorTickCount method

Returns the number of minor ticks between major ticks on a chart axis.

You can call this method only on a value scale axis.

Syntax Function GetMinorTickCount( ) As Integer

Returns The number of minor ticks between major ticks on a chart axis.

See also AcChartAxis::GetMajorTickCount method
AcChartAxis::SetMinorTickCount method

AcChartAxis::GetMinorTickPlacement method

Returns the placement of minor ticks on a chart axis. You can call this method only on a value scale axis.

Syntax Function GetMinorTickPlacement( ) As AcChartAxisTickPlacement

Returns The placement of minor ticks on a chart axis.

See also AcChartAxis::GetMajorTickPlacement method
AcChartAxis::SetMinorTickPlacement method
AcChartAxisPlacement

AcChartAxis::GetNoZeroRatio method

Returns the minimum ratio between the lowest and highest values plotted on a chart axis that will cause zero to be suppressed on that axis.

For example, if the highest value plotted against a chart axis is 100, and the no zero ratio for that axis is 0.7:

- If the lowest value plotted against the axis is 70, zero will be suppressed.
- If the lowest value plotted against the axis is 69.9, zero will not be suppressed.

You can call this method only on a value scale axis.

Syntax Function GetNoZeroRatio( ) As Double

Returns The minimum ratio between the lowest and highest values plotted on the chart axis that will cause zero to be suppressed on the axis.

See also AcChartAxis::SetNoZeroRatio method

AcChartAxis::GetNumberOfGridLines method

Determines the number of grid lines in a chart axis. You can call this method only from:

- A chart’s AdjustChart( ) method
- A chart’s DrawOnChart( ) method

Syntax Function GetNumberOfGridLnes( ) As Integer

Returns The number of grid lines in the chart axis.

See also AcChart::AdjustChart method
AcChart::DrawOnChart method
AcChartAxis::AddGridLine method
AcChartAxis::GetGridLine method
AcChartAxis::InsertGridLine method
Class AcChartGridLine

AcChartAxis::GetNumberOfLabels method

Returns the number of labels on a chart axis. Category labels are used as category scale axis labels. The value that this method returns for a category scale axis is exactly the same as the value that the GetNumberOfCategories( ) method returns of the parent chart layer of that axis.

Syntax Function GetNumberOfLabels( ) As Integer

Returns The number of labels on a chart axis.

See also AcChartLayer::GetNumberOfCategories method

AcChartAxis::GetOriginValue method

Returns the origin of a chart axis.

You can call this method only on a value scale axis.

Syntax Function GetOriginValue( ) As Variant

Returns The origin of a chart axis.

See also AcChartAxis::GetOtherAxisCrossesAt method
AcChartAxis::GetOtherAxisPlacement method

AcChartAxis::GetOtherAxisCrossesAt method

Returns the value at which the opposite chart axis crosses a chart axis.

Syntax Function GetOtherAxisCrossesAt( ) As Variant

Returns The value at which the opposite chart axis crosses the chart axis.

If the chart axis is a category scale, this method returns the tick number on the axis at which the opposite axis crosses. The first tick is number 1.

See also AcChartAxis::GetOriginValue method
AcChartAxis::GetOtherAxisPlacement method
AcChartAxis::SetOtherAxisCrossesAt method

AcChartAxis::GetOtherAxisPlacement method

Returns the placement of the opposite axis relative to a chart axis.

Syntax Function GetOtherAxisPlacement( ) As AcChartAxisPlacement

Returns The placement of the opposite axis relative to a chart axis.

See also AcChartAxisPlacement
AcChartAxis::GetOriginValue method
AcChartAxis::GetOtherAxisCrossesAt method

AcChartAxis::SetOtherAxisPlacement method

AcChartAxis::GetOuterMarginRatio method

Returns the minimum ratio between the outer margin on a chart axis and the total range of that axis. For example, if this method returns 0.05 for a bar chart layer's *y*-axis, the longest bar will be no more than 95% of the total height of the axis from the top of the axis.

You can call this method only on a value scale axis.

Syntax Function GetOuterMarginRatio( ) As Double

Returns The minimum ratio between the outer margin on the chart axis and the total range of the axis.

See also AcChartAxis::GetOuterMarginRatio method
AcChartAxis::SetOuterMarginRatio method

AcChartAxis::GetTitleStyle method

Returns the style of the title of a chart axis. To change the style of the title of a chart axis, call this method to get the default settings.

Syntax Function GetTitleStyle( ) As AcDrawingTextStyle

Returns The style of the title of the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to make the title of the *y*-axis of the chart's overlay layer bold, depending on the value of a Boolean parameter. GetTitleStyle( ) retrieves the default settings so that only the title style's Font member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmBoldOverlayTitle Then
      Dim yAxis As AcChartAxis
      Set yAxis = overlayLayer.GetYAxis( )
      Dim TitleStyle As AcDrawingTextStyle
      TitleStyle = yAxis.GetTitleStyle( )
      TitleStyle.Font.Bold = True
      yAxis.SetTitleStyle( TitleStyle )
   End If
End Sub
```

See also AcChartAxis::SetTitleStyle method
AcDrawingTextStyle

AcChartAxis::GetTitleText method

Returns the text of the title of a chart axis.

Syntax Function GetTitleText( ) As AcDrawingTextStyle

Returns The text of the title of a chart axis.

See also AcChartAxis::SetTitleText method

AcChartAxis::HasFixedMaximum method

Determines whether a chart axis has a fixed upper bound. You can call this method only on a value scale axis.

Syntax Function HasFixedMaximum( ) As Boolean

Returns True if the chart axis has a fixed upper bound.
False if the upper bound of the chart axis is computed from data plotted on the axis.

See also AcChartAxis::GetMaximumValue method
AcChartAxis::SetMaximumValue method

AcChartAxis::HasFixedMinimum method

Determines whether a chart axis has a fixed lower bound. You can only call this method on a value scale axis.

Syntax Function HasFixedMinimum( ) As Boolean

Returns True if the chart axis has a fixed lower bound.
False if the lower bound of the chart axis is computed from data plotted on the axis.

See also AcChartAxis::GetMinimumValue method
AcChartAxis::SetMinimumValue method

AcChartAxis::IgnoreTrendlines method

Determines whether trendlines will be ignored when computing the scale for a chart axis.

Syntax Function IgnoreTrendlines( ) As Boolean

Returns True if trendlines will be ignored when computing the scale for the chart axis.
False if the chart axis's scale will be adjusted to fit trendlines.

See also AcChartAxis::SetIgnoreTrendlines method
Class AcChartTrendline

AcChartAxis::InsertGridLine method

Call this method to insert a grid line at a specific position within a chart axis's list of grid lines. When you insert a new grid line, the original grid line at the insertion point and all the grid lines above the insertion point move up one place.

You can call this method only from:

- A chart's AdjustChart( ) method
- A chart's DrawOnChart( ) method
- Code that creates a chart dynamically

Syntax Function InsertGridLine(index As Integer, value As Variant) As AcChartGridLine

Parameters **index**
The position in the chart axis's list of grid lines at which the new grid line will be inserted. The first grid line is index 1.

Must be greater than or equal to one. Must be less than or equal to the current number of grid lines in the chart axis plus one.

value
The axis value at which the grid line is drawn. If the axis is a category scale axis, the first tick on the axis has the value 0, the second tick has the value 1, and so on.

Returns A handle to the new grid line object.

See also AcChart::AdjustChart method
AcChart::DrawOnChart method
AcChartAxis::AddGridLine method
AcChartAxis::GetGridLine method
AcChartAxis::GetNumberOfGridLines method
Class AcChartGridLine

AcChartAxis::IsCategoryScale method

Determines whether a chart axis is a category scale axis.

Syntax Function IsCategoryScale( ) As Boolean

Returns True if the chart axis is a category scale axis.
False if the chart axis is not a category scale axis.

See also AcChartAxis::IsValueScale method

AcChartAxis::IsValueScale method

Determines whether a chart axis is a value scale axis.

Syntax Function IsValueScale( ) As Boolean

Returns True if the chart axis is a value scale axis.
False if the chart axis is not a value scale axis.

See also AcChartAxis::IsCategoryScale method

AcChartAxis::IsXAxis method

Determines whether a chart axis is the *x*-axis of its parent chart layer.

Syntax Function IsXAxis( ) As Boolean

Returns True if the chart axis is the *x*-axis of its parent chart layer.
False if the chart axis is not the *x*-axis of its parent chart layer.

See also AcChartAxis::IsYAxis method
AcChartAxis::IsZAxis method

AcChartAxis::IsYAxis method

Determines whether a chart axis is the *y*-axis of its parent chart layer.

Syntax Function IsYAxis( ) As Boolean

Returns True if the chart axis is the *y*-axis of its parent chart layer.
False if the chart axis is not the *y*-axis of its parent chart layer.

See also AcChartAxis::IsXAxis method
AcChartAxis::IsZAxis method

AcChartAxis::IsZAxis method

Determines whether a chart axis is the *z*-axis of its parent chart layer.

Syntax Function IsZAxis( ) As Boolean

Returns True if the chart axis is the *z*-axis of its parent chart layer.
False if the chart axis is not the *z*-axis of its parent chart layer.

See also AcChartAxis::IsXAxis method
AcChartAxis::IsYAxis method

AcChartAxis::PlotCategoriesBetweenTicks method

Determines whether categories are plotted between the ticks on a chart axis.

You can call this method only on a category scale axis.

Syntax Function PlotCategoriesBetweenTicks( ) As Boolean

Returns True if categories are plotted between the ticks on the chart axis.
False if categories are plotted on the ticks on the chart axis.

See also AcChartAxis::SetPlotCategoriesBetweenTicks method

AcChartAxis::ResetMajorTickInterval method

Call the ResetMajorTickInterval( ) method to reset the major tick interval of a chart axis to its default setting.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub ResetMajorTickInterval( )

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ComputeScale method
AcChartAxis::GetMajorTickCalculation method
AcChartAxis::GetMajorTickInterval method
AcChartAxis::SetMajorTickCalculation method
AcChartAxis::SetMajorTickInterval method

AcChartAxis::SetDataType method

Call the SetDataType( ) method to set the data type of the scale of a chart axis. A value scale axis determines its data type automatically from data plotted on the axis. If necessary, use type conversion functions in the data expressions in Chart Builder to give your chart data the data types you want.

To overrule the automatic data type setting for a chart axis, call the SetDataType( ) method to explicitly set the required data type.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetDataType(dataType As AcDataType)

Parameter **dataType**
The data type to set. Must be either DataTypeDateTime or DataTypeNumber.

See also AcChart::CustomizeAxes method

AcChartAxis::GetDataType method

AcChartAxis::SetDefaultRangeRatio method

Call the SetDefaultRangeRatio( ) method to set the default ratio used to scale a chart axis when all the values plotted on the axis lie on the axis's origin. You can call this method only on a value scale axis.

The axis's origin value is multiplied by the default range ratio to give a range. That range is subtracted from the value to get the lower bound of the axis, and added to the value to get the upper bound of the axis. For example, if all the points plotted on a chart axis have the value 100 and the default range ratio is 0.1, the lower bound of the axis is 90 and the upper bound of the axis is 110.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetDefaultRangeRatio(defaultRangeRatio As Double)

Parameter **defaultRangeRatio**
The ratio used to compute the range of the chart axis when all the values plotted on the axis lie on the axis's origin. Must be in the range 0.01 through 1.

Example The following example overrides a chart's CustomizeAxes( ) method to set the default range ratio for the *y*-axis of the chart's base layer to 1:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   yAxis.SetDefaultRangeRatio( 0.2 )
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetDefaultRangeRatio method

AcChartAxis::SetForceMajorTickCount method

Call the SetForceMajorTickCount( ) method to specify whether the number of major ticks on a chart axis is forced to be a specific value. You can set the number of major ticks an a chart axis by calling the axis's SetMajorTickCount( ) method. You can call SetForceMajorTickCount( ) only on a value scale axis.

You can call SetForceMajorTickCount( ) only from:

- A chart's CustomizeAxes( ) method

- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetForceMajorTickCount(forceMajorTickCount As Boolean)

Parameter **forceMajorTickCount**
True forces the number of major ticks on the chart axis to be a specific value. False allows the number of major ticks on the chart axis to be computed from data plotted on the axis.

Example The following example overrides a chart's CustomizeAxes( ) method to set the number of major ticks on the *y*-axis of the chart's base layer to a value specified by a parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If (parmNumberOfYMajorTicks > 0) Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.SetMajorTickCount( parmNumberOfYMajorTicks )
      yAxis.SetForceMajorTickCount( True )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::ForceMajorTickCount method
AcChartAxis::SetMajorTickCount method

AcChartAxis::SetIgnoreTrendlines method

Call this method to specify whether trendlines will be ignored when computing the scale for a chart axis.

Syntax Sub SetIgnoreTrendlines(ignoreTrendlines As Boolean)

Parameter **ignoreTrendlines**

True causes trendlines to be ignored when computing the scale for the chart axis. False cause the chart axis's scale to be adjusted to fit trendlines.

See also AcChartAxis::IgnoreTrendlines method
Class AcChartTrendline

AcChartAxis::SetInnerMarginRatio method

Call the SetInnerMarginRatio( ) method to set the minimum ratio between the inner margin on a chart axis and the total range of that axis. For example, if you call SetInnerMarginRatio( ) on a bar chart layer's *y*-axis with innerMarginRatio set to 0.25, the shortest bar will be at least 25% of the total height of the axis from the bottom of the axis.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetInnerMarginRatio(innerMarginRatio As Double)

Parameter **innerMarginRatio**
The minimum ratio between the inner margin on the chart axis and the total range of the axis. Must be in the range 0 through 0.5.

Example The following example overrides a chart's CustomizeAxes( ) method to make the inner margin on the *y*-axis of the chart's base layer at least 35% of the total height of the axis:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   yAxis.SetInnerMarginRatio( 0.35 )
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetInnerMarginRatio method
AcChartAxis::SetOuterMarginRatio method

AcChartAxis::SetLabelFormat method

Call the SetLabelFormat( ) method to set the format pattern used to format labels on a chart axis. The format pattern is ignored for string label values.

The recommended method from which to call SetLabelFormat( ) is a chart's CustomizeAxes( ) method.

To change the format pattern at view time, call this method from a chart's Localize( ) method.

Category labels are used as category scale axis labels. Setting a format pattern for a category scale axis with this method has the same effect as setting a format pattern with the SetCategoryLabelFormat( ) method of the parent chart layer of that axis.

Syntax Sub Format(labelFormat As String)

Parameter **labelFormat**
The format pattern.

Example The following example overrides a chart's CustomizeAxes( ) method to use a short or long date format for labels on the *x*-axis of the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim xAxis As AcChartAxis
   Set xAxis = baseLayer.GetXAxis( )
   If parmUseShortDateFormat Then
      xAxis.SetLabelFormat( "Short Date" )
   Else
      xAxis.SetLabelFormat( "Long Date" )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetLabelFormat method
AcChartLayer::SetCategoryLabelFormat method

AcChartAxis::SetLabelPlacement method

Call the SetLabelPlacement( ) method to specify the placement of labels on a chart axis.

The recommended method from which to call SetLabelPlacement( ) is a chart's CustomizeAxes( ) method.

Syntax Sub SetLabelPlacement(labelPlacement As AcChartAxisLabelPlacement)

Parameter **labelPlacement**
The placement of labels on the chart axis.

Set labelPlacement to ChartAxisLabelPlacementNone if you do not want to show labels on the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to disable labels on the *x*-axis of the chart's base layer if there is only one category:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim xAxis As AcChartAxis
   Set xAxis = baseLayer.GetXAxis( )
   If (baseLayer.GetNumberOfCategories( ) = 1) Then
      ' Disable labels on the x-axis.
      xAxis.SetLabelPlacement( ChartAxisLabelPlacementNone )
   End If
End Sub
```

See also AcChartAxisPlacement
AcChart::CustomizeAxes method
AcChartAxis::GetLabelPlacement method

AcChartAxis::SetLabelStyle method

Call the SetLabelStyle( ) method to set the style for labels on a chart axis.

The recommended method from which to call SetLabelStyle( ) is a chart's CustomizeAxes( ) method. To change the label style at view time, call SetLabelStyle( ) from a chart's Localize( ) method.

Syntax Sub SetLabelStyle(labelStyle As AcDrawingTextStyle)

Parameter **labelStyle**
The style for labels on the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to make axis labels on the *x*-axis of the chart's base layer italic, depending on the value of a Boolean parameter. GetLabelStyle( ) retrieves the default settings so that only the title style's Font member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmItalicXLabels Then
      Dim xAxis As AcChartAxis
      Set xAxis = baseLayer.GetXAxis( )
      Dim labelStyle As AcDrawingTextStyle
      labelStyle = xAxis.GetLabelStyle( )
      labelStyle.Font.Italic = True
      xAxis.SetLabelStyle( labelStyle )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetLabelStyle method
AcDrawingTextStyle

AcChartAxis::SetLabelValue method

Call the SetLabelValue( ) method to set the value of the specified label on a chart axis. The recommended method from which to call SetLabelValue( ) is a chart's AdjustChart( ) method.

To change label values at view time, call this method from a chart's Localize( ) method.

If you call this method from code that is creating a chart dynamically, you must call it after you call the chart's ComputeScales( ) method.

To determine the number of labels on a chart axis, call the axis's GetNumberOfLabels( ) method.

Category labels are used as category scale axis labels. Setting a label value for a category scale axis with this method has the same effect as setting a label value with the SetLabelValue( ) method of the corresponding chart category.

Syntax Function SetLabelValue(index As Integer, labelValue As Variant)

Parameters **index**
An index into the axis's list of labels. The first label is index 1.
Must be in the range 1 through the number of labels on the axis.

labelValue
The label value.

Examples The following example overrides a chart's AdjustChart( ) method to scale the *y*-axis labels on the chart's base layer and set the *y*-axis title to match. Because this adjustment relies on the automatically computed label values, it can only be made in AdjustChart( ).

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   Dim labelValue As Integer
   ' Assume axis starts at 0.
   ' Get the value of the first label above 0.
   labelValue = yAxis.GetLabelValue( 2 )
   If (labelValue > 1000) Then
      Dim numberOfLabels As Integer
      numberOfLabels = yAxis.GetNumberOfLabels( )
      Dim i As Integer
      For i = 2 To numberOfLabels
         labelValue = yAxis.GetLabelValue( i )
         yAxis.SetLabelValue( i, labelValue / 1000 )
      Next i
      yAxis.SetTitleText( "Sales ($K)" )
   End If
End Sub
```

The following example overrides a chart's Localize( ) method to translate labels on the chart's *x*-axis into French at view time if the viewing locale is French:

```
Sub Localize( baseLayer As AcChartLayer, overlayLayer As
   AcChartLayer, studyLayers( ) As AcChartLayer )
   If (GetLocaleName( ) = "fr_FR") Then
      Dim xAxis As AcChartAxis
      Set xAxis = baseLayer.GetXAxis( )
      Dim numberOfLabels As Integer
      numberOfLabels = xAxis.GetNumberOfLabels
      Dim labelIndex As Integer
      For labelIndex = 1 To numberOfLabels
         Select Case xAxis.GetLabelValue( labelIndex )
         Case "North"
            xAxis.SetLabelValue( labelIndex, "Nord" )
         Case "South"
            xAxis.SetLabelValue( labelIndex, "Sud" )
         Case "East"
            xAxis.SetLabelValue( labelIndex, "Est" )
         Case "West"
            xAxis.SetLabelValue( labelIndex, "Ouest" )
         End Select
      Next labelIndex
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartAxis::GetLabelValue method
AcChartAxis::GetNumberOfLabels method
AcChartAxis::SetLabelValue method

AcChartAxis::SetLineStyle method

Call the SetLineStyle( ) method to set the line style used to draw a chart axis.

The recommended method from which to call SetLineStyle( ) is a chart's AdjustChart( ) method.

Syntax Sub SetLineStyle(lineStyle As AcDrawingLineStyle)

Parameter **lineStyle**
The line style used to draw the chart axis.
Set the Pen member of lineStyle to DrawingLinePenNone to hide the axis.

Example The following example overrides a chart's CustomizeAxes( ) method to change the thickness of all axes in the chart, depending on the value of a Boolean parameter. GetLineStyle( ) retrieves the default settings so that only the line style's Width member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmHeavyAxes Then
      Dim numberOfLayers As Integer
      numberOfLayers = GetNumberOfLayers( )
      Dim layerIndex As Integer
      For layerIndex = 1 To numberOfLayers
         Dim layer As AcChartLayer
         Set layer = GetLayer( layerIndex )
```

```
            Dim lineStyle As AcDrawingLineStyle
            Dim axis As AcChartAxis
            Set axis = layer.GetXAxis( )
            If Not axis Is Nothing Then
               lineStyle = axis.GetLineStyle( )
               lineStyle.Width = 2 * OnePoint
               axis.SetLineStyle( lineStyle )
            End If
            Set axis = layer.GetYAxis( )
            If Not axis Is Nothing Then
               lineStyle = axis.GetLineStyle( )
               lineStyle.Width = 2 * OnePoint
               axis.SetLineStyle( lineStyle )
            End If
         Next layerIndex
      End If
   End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetLineStyle method
AcDrawingLineStyle

AcChartAxis::SetMajorGridLineStyle method

Call the SetMajorGridLineStyle( ) method to set the line style used to draw grid lines for the major ticks on a chart axis.

The recommended method from which to call SetMajorGridLineStyle( ) is a chart's AdjustChart( ) method.

Syntax Sub SetMajorGridLineStyle(majorGridLineStyle As AcDrawingLineStyle)

Parameter **majorGridLineStyle**
The line style used to draw grid lines for the major ticks on the chart axis.
If you do not want to display major grid lines on the chart axis, set the Pen member of majorGridLineStyle to DrawingLinePenNone.

Example The following example overrides a chart's CustomizeAxes( ) method to set dotted major grid lines for the *y*-axis of the chart's base layer, depending on the value of a Boolean parameter. GetMajorGridLineStyle( ) retrieves the default settings so that only the line style's Pen member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmDottedMajorGrid Then
      Dim yAxis As AcChartAxis
      Set yAxis = layer.GetYAxis( )
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = yAxis.GetMajorGridLineStyle( )
```

```
        lineStyle.Pen = DrawingLinePenDot
        yAxis.SetMajorGridLineStyle( lineStyle )
    End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetMajorGridLineStyle method
AcChartAxis::SetMinorGridLineStyle method
AcDrawingLineStyle

AcChartAxis::SetMajorTickCalculation method

Call the SetMajorTickCalculation( ) method to specify the type of calculation used to compute major ticks on a chart axis.

You can call this method only on a value scale axis. You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

If you call SetMajorTickCalculation( ) from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

In some cases, the value you set by calling SetMajorTickCalculation( ) is altered automatically when the axis's scale is computed, as follows:

- If the axis has fixed lower and upper bounds and the major tick count is forced, the interval is also forced. In this case, the major tick calculation is changed to ChartTickCalculationAuto.
- If the axis has fixed lower and upper bounds, the major tick count is not forced, and the major tick calculation is ChartTickCalculationExactInterval, the calculation is changed to ChartTickCalculationMinimumInterval. This change is done because forcing an exact interval might violate the maximum major tick count.

Setting an exact major tick interval on an axis can cause some points to be clipped.

Syntax Sub SetMajorTickCalculation(majorTickCalculation As AcChartTickCalculation)

Parameter **majorTickCalculation**
The type of calculation used to compute major ticks on the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to force the minimum interval between major ticks on the *y*-axis of the chart's base layer to a value specified by a parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If (parmMinimumYMajorTickInterval > 0) Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.SetMajorTickCalculation(
         ChartTickCalculationMinimumInterval )
      yAxis.SetMajorTickInterval(
         parmMinimumYMajorTickInterval )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ComputeScale method
AcChartAxis::GetMajorTickCalculation method
AcChartAxis::SetMajorTickInterval method
AcChartTickCalculation

AcChartAxis::SetMajorTickCount method

Call the SetMajorTickCount( ) method to set the exact or maximum number of major ticks on a chart axis. You can call this method only on a value scale axis.

If the major tick count is forced, the axis will have the exact number of major ticks set by this method. If the major tick count is not forced, the axis will have no more than the number of major ticks this method sets.

You can call SetMajorTickCount( ) only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

If you call SetMajorTickCount( ) from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub SetMajorTickCount(majorTickCount As Integer)

Parameter **majorTickCount**
The exact or maximum number of major ticks on the chart axis. Must be greater than or equal to 2.

Example The following example overrides a chart's CustomizeAxes( ) method to set the number of major ticks on the *y*-axis of the chart's base layer to a value specified by a parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If (parmNumberOfYMajorTicks > 0) Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.SetMajorTickCount( parmNumberOfYMajorTicks )
      yAxis.SetForceMajorTickCount( True )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ComputeScale method
AcChartAxis::ForceMajorTickCount method
AcChartAxis::GetMajorTickCount method
AcChartAxis::SetForceMajorTickCount method
AcChartAxis::SetMinorTickCount method

AcChartAxis::SetMajorTickInterval method

Call the SetMajorTickInterval( ) method to set the exact or minimum interval between major ticks on a chart axis. If the major tick calculation is ChartTickCalculationAuto, then SetMajorTickInterval( ) automatically changes the major tick calculation to ChartTickCalculationExactInterval.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub SetMajorTickInterval(majorTickInterval As Double)

Parameter **majorTickInterval**
The interval between major ticks on the chart axis. Must be greater than zero.

Example The following example overrides a chart's CustomizeAxes( ) method to force the minimum interval between major ticks on the *y*-axis of the chart's base layer to a value specified by a parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
```

```
   If (parmMinimumYMajorTickInterval > 0) Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.SetMajorTickCalculation(
         ChartTickCalculationMinimumInterval )
      yAxis.SetMajorTickInterval( parmMinimumYMajorTickInterval )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ComputeScale method
AcChartAxis::GetMajorTickInterval method
AcChartAxis::SetMajorTickCalculation method

AcChartAxis::SetMajorTickPlacement method

Call the SetMajorTickPlacement( ) method to specify the placement of major ticks on a chart axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

Syntax Sub SetMajorTickPlacement(majorTickPlacement As AcChartTickPlacement)

Parameter **majorTickPlacement**
The placement of major ticks on the chart axis.

Set majorTickPlacement to ChartTickPlacementNone if you do not want to show major ticks on the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to disable major ticks on the *x*-axis of the chart's base layer if only one category exists:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim xAxis As AcChartAxis
   Set xAxis = baseLayer.GetXAxis( )
   If (baseLayer.GetNumberOfCategories( ) = 1) Then
      ' Disable major ticks on the x-axis.
      xAxis.SetMajorTickPlacement( ChartTickPlacementNone )
   End If
End Sub
```

See also AcChart::CustomizeAxes method

AcChartAxis::ComputeScale method
AcChartAxis::GetMajorTickPlacement method
AcChartAxis::SetMinorTickPlacement method
AcChartTickPlacement

AcChartAxis::SetMaximumDataValue method

Call the SetMaximumDataValue( ) method to use a specific value as if it were the highest value plotted against a chart axis. This method supports using the standard automatic scaling mechanism with values that do not exist as points.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub SetMaximumDataValue(maximumDataValue As Variant)

Parameter **maximumDataValue**
The value to use as if it were the lowest value plotted against the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to make the *y*-axis of the chart's base layer symmetrical above and below zero. Using SetMaximumDataValue( ) and SetMinimumDataValue( ) allows the automatic scaling mechanism to maintain the correct outer margins.

```
Sub CustomizeAxes( baseLayer As AcChartLayer, overlayLayer As
   AcChartLayer,
+  studyLayers( ) As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   Dim minimumDataValue As Variant
   minimumDataValue = yAxis.GetMinimumDataValue( )
   Dim maximumDataValue As Variant
   maximumDataValue = yAxis.GetMaximumDataValue( )
   If (-minimumDataValue > maximumDataValue) Then
      yAxis.SetMaximumDataValue( -minimumDataValue )
   Else
      yAxis.SetMinimumDataValue( -maximumDataValue )
   End If
   ' Recompute the axis scale.
   yAxis.ComputeScale( )
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ComputeScale method
AcChartAxis::GetMaximumDataValue method
AcChartAxis::SetMaximumValue method
AcChartAxis::SetMinimumDataValue method

AcChartAxis::SetMaximumValue method

Call the SetMaximumValue( ) method to set a fixed upper bound on a chart axis.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Setting a fixed upper bound on an axis can cause some points to be clipped.

Syntax Sub SetMaximumValue(maximumValue As Variant)

Parameter **maximumValue**
The upper bound of the chart axis.

Example The following example adjusts the upper bound of a study layer's *y*-axis so that it is at least 100. Because this adjustment relies on the automatically computed upper bound, it can only be made in AdjustChart( ).

```
Sub AdjustChart( baseLayer As AcChartLayer,
+  overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = studyLayers(1).GetYAxis( )
   If (yAxis.GetMaximumValue( ) < 100) Then
      yAxis.SetMaximumValue( 100 )
      ' Recompute the ticks and labels.
      yAxis.ComputeScale( )
   End If
End Sub
```

See also AcChart::AdjustChart method

AcChart::CustomizeAxes method
AcChartAxis::ClearMaximumValue method
AcChartAxis::ComputeScale method
AcChartAxis::GetMaximumValue method
AcChartAxis::HasFixedMaximum method
AcChartAxis::SetMaximumDataValue method
AcChartAxis::SetMinimumValue method

AcChartAxis::SetMinimumDataValue method

Call the SetMinimumDataValue( ) method to use a specific value as if it were the lowest value plotted against a chart axis. This allows you to use the standard automatic scaling mechanism with values that do not exist as points.

You can call this method only on a value scale axis.

You can call this method only from:

- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's CustomizeAxes( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub SetMinimumDataValue(minimumDataValue As Variant)

Parameter **minimumDataValue**
The value to use as if it were the lowest value plotted against the chart axis.

Example For an example of how to use this method, see the example for the AcChartAxis::SetMaximumDataValue( ) method.

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ComputeScale method
AcChartAxis::GetMinimumDataValue method
AcChartAxis::SetMaximumDataValue method
AcChartAxis::SetMinimumValue method

AcChartAxis::SetMinimumValue method

Call the SetMinimumValue( ) method to set a fixed lower bound on a chart axis. You can only call this method on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method

- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Setting a fixed lower bound on an axis might cause some points to be clipped.

Syntax Sub SetMinimumValue(minimumValue As Variant)

Parameter **minimumValue**
The lower bound of the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to set the lower bound of the *y*-axis of a chart's base layer to zero, depending on the value of a Boolean parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmYAxisClipsAtZero Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.SetMinimumValue( 0 )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ClearMinimumValue method
AcChartAxis::ComputeScale method
AcChartAxis::GetMinimumValue method
AcChartAxis::HasFixedMinimum method
AcChartAxis::SetMaximumValue method
AcChartAxis::SetMinimumValue method

AcChartAxis::SetMinorGridLineStyle method

Call the SetMinorGridLineStyle( ) method to set the line style used to draw grid lines for the minor ticks on a chart axis.

The recommended method from which to call SetMinorGridLineStyle( ) is a chart's CustomizeAxes( ) method.

Syntax Sub SetMinorGridLineStyle(minorGridLineStyle As AcDrawingLineStyle)

Parameter **minorGridLineStyle**
The line style used to draw grid lines for the minor ticks on the chart axis.
If you do not want to display minor grid lines on the chart axis, set the Pen member of minorGridLineStyle to DrawingLinePenNone.

Example The following example overrides a chart's CustomizeAxes( ) method to set dotted minor grid lines for the *y*-axis of the chart's base layer, depending on the value of a Boolean parameter. GetMinorGridLineStyle( ) retrieves the default settings so that only the line style's Pen member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmDottedMinorGrid Then
      Dim yAxis As AcChartAxis
      Set yAxis = layer.GetYAxis( )
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = yAxis.GetMinorGridLineStyle( )
      lineStyle.Pen = DrawingLinePenDot
      yAxis.SetMinorGridLineStyle( lineStyle )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetMinorGridLineStyle method
AcChartAxis::SetMajorGridLineStyle method
AcDrawingLineStyle

AcChartAxis::SetMinorTickCount method

Call the SetMinorTickCount( ) method to set the number of minor ticks between major ticks on a chart axis. You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

Syntax Sub SetMinorTickCount(minorTickCount As Integer)

Parameter **minorTickCount**
The number of minor ticks between major ticks on the chart axis. Must be greater than or equal to 1.

Example The following example overrides a chart's AdjustChart( ) method to set the number of minor ticks depending on the interval between major ticks:

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   Dim interval As Integer
   interval = yAxis.GetMajorTickInterval( )
   Do
      Select Case interval
      Case 1, 5
         yAxis.SetMinorTickCount( 4 )
         Exit Do
      Case 2
         yAxis.SetMinorTickCount( 1 )
         Exit Do
      Case 0
         ' Didn't find the interval - disable minor ticks.
         yAxis.SetMinorTickPlacement( ChartTickPlacementNone )
         Exit Do
      End Select
      interval = interval \ 10
   Loop
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::GetMinorTickCount method

AcChartAxis::SetMinorTickPlacement method

Call the SetMinorTickPlacement( ) method to specify the placement of minor ticks on a chart axis.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

Syntax Sub SetMinorTickPlacement(minorTickPlacement As AcChartTickPlacement)

Parameter **minorTickPlacement**
The placement of minor ticks on the chart axis.
Set minorTickPlacement to ChartTickPlacementNone if you do not want to show minor ticks on the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to enable minor ticks on the *y*-axis of the chart's overlay layer, depending on the value of a Boolean parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = overlayLayer.GetYAxis( )
   If parmShowMinorTicks Then
      yAxis.SetMinorTickPlacement( ChartTickPlacementOutside )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetMinorTickPlacement method
AcChartAxis::SetMajorTickPlacement method
AcChartTickPlacement

AcChartAxis::SetNoZeroRatio method

Call the SetNoZeroRatio( ) method to set the minimum ratio between the lowest and highest values plotted on a chart axis that will cause zero to be suppressed on that axis. For example, if the highest value plotted against a chart axis is 100, and the no zero ratio for that axis is 0.7:

- If the lowest value plotted against the axis is 70, zero will be suppressed.
- If the lowest value plotted against the axis is 69.9, zero will not be suppressed.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetNoZeroRatio(noZeroRatio As Double)

Parameter **noZeroRatio**
The minimum ratio between the lowest and highest values plotted on a chart axis that will cause zero to be suppressed on that axis. Must be in the range 0 through 1. 0 means that zero can be suppressed as long as all the values plotted on the axis have the same sign. 1 means that zero can never be suppressed.

Example The following example overrides a chart's CustomizeAxes( ) method to prevent zero from being suppressed on the *y*-axis of the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmNoZeroSuppression Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.SetNoZeroRatio( 1 )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetNoZeroRatio method

AcChartAxis::SetOtherAxisCrossesAt method

Call the SetOtherAxisCrossesAt( ) method to set the value at which the opposite axis crosses a chart axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

SetOtherAxisCrossesAt( ) automatically forces the other axis placement setting to ChartAxisPlacementCustom.

Syntax Sub SetOtherAxisCrossesAt(otherAxisCrossesAt As Variant)

Parameter **otherAxisCrossesAt**
The value at which the opposite chart axis crosses the chart axis.
If the chart axis is a category scale:

- otherAxisCrossesAt specifies the tick number on the axis at which the opposite axis crosses. The first tick is number 1.
- otherAxisCrossesAt must be an integer in the range 1 through the number of ticks on the axis.

Example The following example overrides a chart's CustomizeAxes( ) method to set the value at which the chart's base layer's *x*-axis crosses its *y*-axis to the value of the first point in the first series:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   Dim point As AcChartPoint
   Set point = series.GetPoint( 1 )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   yAxis.SetOtherAxisCrossesAt( point.GetYValue( ) )
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChartAxis::ClearOtherAxisCrossesAt method

AcChartAxis::GetOtherAxisCrossesAt method
AcChartAxis::SetOtherAxisPlacement method

AcChartAxis::SetOtherAxisPlacement method

Call the SetOtherAxisPlacement( ) method to specify the placement of the opposite axis relative to a chart axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method
- A chart's AdjustChart( ) method

If you call this method from a chart's AdjustChart( ) method, you must also call ComputeScale( ) on the chart axis to recompute the axis scale.

Syntax Sub SetOtherAxisPlacement(otherAxisPlacement As AcChartAxisPlacement)

Parameter **otherAxisPlacement**
The placement of the opposite axis relative to the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to change the placement of the *x*-axis relative to the *y*-axis of the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmXAxisCrossesAtTop Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.SetOtherAxisPlacement(
         ChartAxisPlacementRightOrTop )
   End If
End Sub
```

See also AcChartAxisPlacement
AcChartAxis::GetOtherAxisPlacement method
AcChartAxis::SetOtherAxisCrossesAt method

AcChartAxis::SetOuterMarginRatio method

Call the SetOuterMarginRatio( ) method to set the minimum ratio between the outer margin on a chart axis and the total range of that axis. For example, if you call this method on a bar chart layer's *y*-axis with outerMarginRatio set to 0.05, the longest bar is no more than 95% of the total height of the axis from the top of the axis.

You can call this method only on a value scale axis.

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetOuterMarginRatio(outerMarginRatio As Double)

Parameter **outerMarginRatio**
The minimum ratio between the outer margin on the chart axis and the total range of the axis. Must be in the range 0 through 0.25.

Example The following example overrides a chart's CustomizeAxes( ) method to make the outer margin on the *y*-axis of the chart's base layer at least 10% of the total height of the axis:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   yAxis.SetOuterMarginRatio( 0.1 )
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::GetOuterMarginRatio method
AcChartAxis::SetInnerMarginRatio method

AcChartAxis::SetPlotCategoriesBetweenTicks method

Call the SetPlotCategoriesBetweenTicks( ) method to specify whether categories are plotted between the ticks on a chart axis. You can call this method only on a category scale axis.

You can call this method only on an axis in a chart if all of the layers in the chart have either of the following chart types:

- Area
- Line

You can call this method only from:

- A chart's CustomizeAxes( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetPlotCategoriesBetweenTicks(plotCategoriesBetweenTicks As Boolean)

Parameter **plotCategoriesBetweenTicks**
True causes categories to be plotted between the ticks on the chart axis.
False causes categories to be plotted on the ticks on the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to plot categories between ticks on the *x*-axis of the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim xAxis As AcChartAxis
   Set xAxis = baseLayer.GetXAxis( )
   xAxis.SetPlotCategoriesBetweenTicks(
      parmCategoriesBetweenTicks )
End Sub
```

See also AcChart::CustomizeAxes method
AcChartAxis::PlotCategoriesBetweenTicks method

AcChartAxis::SetTitleStyle method

Call SetTitleStyle( ) to set the style of the title of a chart axis. The recommended method from which to call SetTitleStyle( ) is a chart's CustomizeAxes( ) method. To change the title style at view time, call SetTitleStyle( ) from a chart's Localize( ) method.

Syntax Sub SetTitleStyle(titleStyle As AcDrawingTextStyle)

Parameter **titleStyle**
The style of the title of the chart axis.

Example The following example overrides a chart's CustomizeAxes( ) method to make the axis title of the *x*-axis of a chart italic, depending on the value of a Boolean parameter. GetTitleStyle( ) retrieves the default settings so that only the title style's Font member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmItalicTitle Then
      Dim xAxis As AcCHart
      Dim titleStyle As AcDrawingTextStyle
      titleStyle = GetTitleStyle( )
      titleStyle.Font.Italic = True
      SetTitleStyle( titleStyle )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChart::Localize method

AcChartAxis::GetTitleStyle method
AcDrawingTextStyle

AcChartAxis::SetTitleText method

Call the SetTitleText( ) method to set the text of the title of a chart axis.

The recommended methods from which to call SetTitleText( ) are:

- A chart's CustomizeAxes( ) method
- A chart's AdjustChart( ) method

To change the title text at view time, call SetTitleText( ) from a chart's Localize( ) method.

Syntax Sub SetTitleText(titleText As String)

Parameter **titleText**
The text of the title of the chart axis. Set this parameter to "" if you do not want a title.

Examples The following example overrides a chart's AdjustChart( ) method to scale the *y*-axis labels on the chart's base layer and set the *y*-axis title to match. Because this adjustment relies on the automatically computed label values, it can only be made in AdjustChart( ).

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   Dim labelValue As Integer
   ' Assume axis starts at 0, get the value of the first label
   ' above 0.
   labelValue = yAxis.GetLabelValue( 2 )
   If (labelValue > 1000) Then
      Dim numberOfLabels As Integer
      numberOfLabels = yAxis.GetNumberOfLabels( )
      Dim i As Integer
      For i = 2 To numberOfLabels
         labelValue = yAxis.GetLabelValue( i )
         yAxis.SetLabelValue( i, labelValue / 1000 )
      Next i
      yAxis.SetTitleText( "Sales ($K)" )
   End If
End Sub
```

For another example of how to use this method, see the dynamic chart example for the AcChart class.

See also Class AcChart

AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::Localize method
AcChartAxis::GetTitleText method

Class AcChartCategory

A category within a chart layer. Figure 7-7 shows the class hierarchy of AcChartCategory.

Figure 7-7 AcChartCategory

Description Use AcChartCategory to represent a single category within a chart layer. Do not create AcChartCategory objects explicitly from your own code. Instead, AcChartLayer objects create AcChartCategory objects as necessary to build complete charts.

To access a chart layer's categories, you must use AcChartLayer's methods. You can manipulate the appearance of a chart by calling methods on the chart's categories. All types of chart layer except scatter chart layers have at least one category. Scatter chart layers do not have categories.

Example For an example of how to use this class to build a chart dynamically, see the dynamic chart example for the AcChart class.

See also Class AcChart
Class AcChartAxis
Class AcChartGridLine
Class AcChartLayer
Class AcChartPoint
Class AcChartPointStyle
Class AcChartSeries
Class AcChartSeriesStyle
Class AcChartTrendline

Methods for Class AcChartCategory

Methods defined in Class AcChartCategory

GetIndex, GetKeyValue, GetLabelText, GetLabelValue, GetLayer, GetSumOfPointValues, SetKeyValue, SetLabelValue

AcChartCategory::GetIndex method

Returns the index of a chart category within its parent chart layer's list of categories. The first category in a layer is index 1.

Syntax Function GetIndex( ) As Integer

Returns The index of the chart category within its parent chart layer's list of categories.

AcChartCategory::GetKeyValue method

Returns the unique key value for a chart category.

Syntax Function GetKeyValue( ) As Variant

Returns The unique key value for the chart category.

See also AcChartCategory::GetLabelValue method
AcChartCategory::SetKeyValue method

AcChartCategory::GetLabelText method

Returns the formatted label text for a chart category. String label values are returned unformatted.

Syntax Function GetLabelText( ) As String

Returns The formatted label text for a chart category.

See also AcChartCategory::GetKeyValue method
AcChartCategory::GetLabelValue method
AcChartCategory::SetLabelValue method

AcChartCategory::GetLabelValue method

Returns the label value for a chart category.

Syntax Function GetLabelValue( ) As Variant

Returns The label value for a chart category.

See also AcChartCategory::GetKeyValue method
AcChartCategory::GetLabelText method
AcChartCategory::SetLabelValue method

AcChartCategory::GetLayer method

Returns a reference to the parent chart layer of a chart category.

Syntax Function GetLayer( ) As AcChartLayer

Returns A reference to the parent chart layer of the chart category.

See also Class AcChartLayer

AcChartCategory::GetSumOfPointValues method

Returns the sum of the y values of all the points in a chart category.

Syntax Function GetSumOfPointValues( ) As Variant

Returns The sum of the y values of all the points in the category.

See also AcChartPoint::GetYValue method
AcChartSeries::GetSumOfPointValues method

AcChartCategory::SetKeyValue method

Call the SetKeyValue( ) method to set the unique key value for a chart category. A chart category's initial key value is set when the category is created. This method changes that value. Changing a category's key value has no effect on the order in which categories appear on the *x*-axis.

You can call this method only from a chart's CustomizeCategoriesAndSeries( ) method.

Syntax Sub SetKeyValue(keyValue As Variant)

Parameter **keyValue**
The unique key value for the chart category.

See also AcChart::CustomizeCategoriesAndSeries method
AcChartCategory::GetKeyValue method
AcChartCategory::GetLabelValue method

AcChartCategory::SetLabelValue method

Call the SetLabelValue( ) method to set the label value for a chart category. A chart category's initial label value is set when the category is created. This method changes that value. Changing a category's label value has no effect on the order in which categories appear on the *x*-axis.

The label value does not have to be a string. Label values are formatted into label text when the chart is viewed. This allows locale-specific formatting. For example, if you set labelValue to 1.5, when the chart is viewed in the US English locale, the label text will be "1.5", but when the chart is viewed in the French locale the label text will be "1,5".

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- A chart's Localize( ) method

Syntax Sub SetLabelValue(labelValue As Variant)

Parameter **labelValue**
The label value for the chart category.

Example The following example overrides a chart's Localize( ) method to translate category labels in the chart's base layer into French at view time if the viewing locale is French:

```
Sub Localize( baseLayer As AcChartLayer, overlayLayer As
   AcChartLayer, studyLayers( ) As AcChartLayer )
   If (GetLocaleName( ) = "fr_FR") Then
      Dim numberOfCategories As Integer
      numberOfCategories = baseLayer.GetNumberOfCategories( )
      Dim categoryIndex As Integer
      For categoryIndex = 1 To numberOfCategories
         Dim category As AcChartCategory
         Set category = baseLayer.GetCategory( categoryIndex )
         Select Case category.GetLabelValue( labelIndex )
         Case "North"
            category.SetLabelValue( "Nord" )
         Case "South"
            category.SetLabelValue( "Sud" )
         Case "East"
            category.SetLabelValue( "Est" )
         Case "West"
            category.SetLabelValue( "Ouest" )
         End Select
      Next categoryIndex
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChart::Localize method
AcChartCategory::GetKeyValue method
AcChartCategory::GetLabelValue method

Class AcChartGridLine

A grid line in a chart. Figure 7-8 shows the class hierarchy of AcChartGridLine.

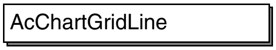


Figure 7-8 AcChartGridLine

Description AcChartGridLine represents a grid line in a chart. A grid line is a horizontal or vertical line across the plot area of a chart layer.

To add grid lines to a chart, use the AcChartAxis::AddGridLine( ) or AcChartAxis::InsertGridLine( ) methods. You cannot use the New keyword or the NewInstance( ) or NewPersistentInstance( ) functions to create AcChartGridLine objects.

To define the appearance of a grid line, call methods on the corresponding AcChartGridLine object.

Example In the following example, a chart's DrawOnChart( ) method has been overridden to add a horizontal grid line to a chart. The grid line indicates the average value of all the data points in the first series in the base layer.

```
Sub DrawOnChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )

   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )

   Dim numberOfPoints As Integer
   numberOfPoints = series.GetNumberOfPoints( )

   Dim averageValue As Double
   Dim pointIndex As Integer
   For pointIndex = 1 To numberOfPoints
      Dim pointValue As Double
      pointValue = series.GetPoint( pointIndex ).GetYValue( )
      If Not IsNull( pointValue ) Then
         averageValue = averageValue + pointValue
      End If
   Next pointIndex
   averageValue = averageValue / numberOfPoints

   Dim yAxis As AcChartAxis
   Set yAxis = baseLayer.GetYAxis( )
   Dim gridLine As AcChartGridLine
   Set gridLine = yAxis.AddGridLine( averageValue )
   gridLine.SetLabelText( "Average" )
   gridLine.SetDrawInFrontOfPoints( True )
   Dim lineStyle As AcDrawingLineStyle
   lineStyle = gridLine.GetLineStyle( )
   lineStyle.Color = Blue
   lineStyle.Pen = DrawingLinePenDot
   lineStyle.Width = 1.5 * OnePoint
   gridLine.SetLineStyle( lineStyle )
End Sub
```

See also Class AcChart
Class AcChartAxis
Class AcChartCategory
Class AcChartLayer
Class AcChartPoint
Class AcChartPointStyle
Class AcChartSeries
Class AcChartSeriesStyle
Class AcChartTrendline

Methods for Class AcChartGridLine

Methods defined in Class AcChartGridLine

DrawInFrontOfPoints, GetAxis, GetIndex, GetLabelText, GetLineStyle, GetValue, SetDrawInFrontOfPoints, SetLabelText, SetLineStyle, SetValue

AcChartGridLine::DrawInFrontOfPoints method

Determines whether a grid line is drawn in front of the data points within a chart.

Syntax Function DrawInFrontOfPoints( ) As Boolean

Returns True if the grid line is drawn in front of the data points.
False if the grid line is drawn behind the data points.

See also AcChartGridLine::SetDrawInFrontOfPoints method

AcChartGridLine::GetAxis method

Returns a reference to the parent chart axis of a grid line.

Syntax Function GetAxis( ) As AcChartAxis

Returns A reference to the parent chart axis of the grid line.

See also Class AcChartAxis

AcChartGridLine::GetIndex method

Returns the index of a grid line within its parent axis's list of grid lines.

Syntax Function GetIndex( ) As Integer

Returns The index of the grid line within its parent axis's list of grid lines. The first grid line for an axis is index 1.

AcChartGridLine::GetLabelText method

Returns the label text for a grid line. The label text appears in the chart legend.

Syntax Function GetLabelText( ) As String

Returns The label text for the grid line.

See also AcChartGridLine::SetLabelText method

AcChartGridLine::GetLineStyle method

Returns the line style used to draw a grid line. Call this method to retrieve the default settings before changing a grid line's line style.

Syntax Function GetLineStyle( ) As AcDrawingLineStyle

Returns The line style used to draw the grid line.

Example For an example of how to use this method, see the example for the AcChartGridLine::SetLineStyle method.

See also AcChartGridLine::SetLineStyle method
AcDrawingLineStyle

AcChartGridLine::GetValue method

Returns the axis value at which a grid line is drawn.

Syntax Function GetValue( ) As Variant

Returns The axis value at which the grid line is drawn. If the grid line's parent axis is a category scale axis, the first tick on the axis has the value 0, the second tick has the value 1, and so on.

See also AcChartGridLine::SetValue method

AcChartGridLine::SetDrawInFrontOfPoints method

Defines whether a grid line is drawn in front of the data points within a chart.

The recommended place from which to call SetDrawInFrontOfPoints( ) is a chart's DrawOnChart( ) method.

Syntax Sub SetDrawInFrontOfPoints(drawInFrontOfPoints As Boolean)

Parameter **drawInFrontOfPoints**
True causes the grid line to be drawn in front of the data points. False causes the grid line to be drawn behind the data points.

Example For an example of how to use this method, see the example for the AcChartGridLine class.

See also Class AcChartGridLine
AcChart::DrawOnChart method
AcChartGridLine::DrawInFrontOfPoints method

AcChartGridLine::SetLabelText method

Sets the label text for a grid line. The label text appears in the chart legend. If the label text is "" or Null, the grid line will not be listed in the legend.

The recommended place from which to call SetLabelText( ) is a chart's DrawOnChart( ) method.

Syntax Function SetlLabelText(labelText As String)

Parameter **labelText**
Text that will be shown in the chart legend. Null or "" if you do not want the grid line to be listed in the chart's legend.

Example For an example of how to use this method, see the example for the AcChartGridLine class.

See also Class AcChartGridLine
AcChart::DrawOnChart method
AcChartGridLine::GetLabelText method

AcChartGridLine::SetLineStyle method

Sets the line style used to draw a grid line.

The recommended place from which to call SetLineStyle( ) is a chart's DrawOnChart( ) method.

Syntax Function SetLineStyle(lineStyle As AcDrawingLineStyle)

Parameter **lineStyle**
The line style used to draw the grid line.

Example For an example of how to use this method, see the example for the AcChartGridLine class.

See also Class AcChartGridLine
AcChart::DrawOnChart method
AcChartGridLine::GetLineStyle method

AcChartGridLine::SetValue method

Sets the axis value at which a grid line is drawn.

The recommended place from which to call SetValue( ) is a chart's DrawOnChart( ) method.

Syntax Function SetValue(value As Variant)

Parameter **value**
The axis value at which the grid line is drawn. If the grid line's parent axis is a category scale axis, the first tick on the axis has the value 0, the second tick has the value 1, and so on.

See also AcChart::DrawOnChart method
AcChartAxis::AddGridLine method
AcChartAxis::InsertGridLine method
AcChartGridLine::GetValue method

Class AcChartLayer

Defines a layer in a chart. Figure 7-9 shows the class hierarchy of AcChartLayer.

Figure 7-9 AcChartLayer

Description A chart layer is a set of points plotted against a single *y*-axis, or a set of points plotted as a pie. Every chart contains one or more chart layers. All charts contain a base layer. In addition to the base layer, some charts contain an overlay layer and one or more study layers.

The points in an overlay layer are plotted in the same area as the points in the base layer, but using a second *y*-axis opposite the base layer's *y*-axis. For example, an overlay line chart layer might be plotted on a base bar chart layer.

A study layer is drawn below a chart's base layer, using its own *y*-axis and a duplicate of the base layer's *x*-axis. An example of a study layer is volume bars drawn below a candlestick stock chart. If a chart contains multiple study layers, the study layers are arranged from top to bottom. Study layers do not overlay each other.

Use the AcChartLayer class to represent a single chart layer. Do not create AcChartLayer objects explicitly from your own code. Instead, AcChart objects create AcChartLayer objects automatically as necessary to build complete charts.

Use AcChart's methods to access a chart's layers. You can manipulate the content and appearance of a chart by calling methods on the chart's layers.

Example For an example of how to use this class to build a chart dynamically, see the dynamic chart example for the AcChart class.

See also Class AcChart
Class AcChartAxis
Class AcChartCategory
Class AcChartGridLine
Class AcChartPoint
Class AcChartPointStyle
Class AcChartSeries
Class AcChartSeriesStyle
Class AcChartTrendline

Methods for Class AcChartLayer

Methods defined in Class AcChartLayer

AddCategory, AddSeries, ChartTypeIsStackable, GetBarShape, GetBubbleSize, GetCategory, GetCategoryGapRatio, GetCategoryGrouping, GetCategoryLabelFormat, GetChart, GetChartType, GetDownBarBorderStyle, GetDownBarFillStyle, GetDropLineStyle, GetHighLowLineStyle, GetIndex, GetLayerType, GetLineWidth, GetMarkerSize, GetMaximumDataXValue, GetMaximumDataYValue, GetMaximumNumberOfPoints, GetMaximumNumberOfPointsPerSeries, GetMaximumNumberOfSeries, GetMaximumTrendlineYValue, GetMinimumDataXValue, GetMinimumDataYValue, GetMinimumTrendlineYValue, GetMissingPoints, GetNumberOfCategories, GetNumberOfSeries, GetPieCenter, GetPieExplosion, GetPieExplosionAmount, GetPieExplosionTestOperator, GetPieExplosionTestValue, GetPieRadius, GetPlotAreaBorderStyle, GetPlotAreaFillStyle, GetPlotAreaPosition, GetPlotAreaSize, GetPointBorderStyle, GetPointLabelFormat, GetPointLabelLineStyle, GetPointLabelPlacement, GetPointLabelSource, GetPointLabelStyle, GetSeries, GetSeriesGrouping, GetSeriesLabelFormat, GetSeriesOverlapRatio, GetSeriesPlacement, GetSeriesStyle, GetStartAngle, GetStudyHeightRatio, GetThreeDBackWallFillStyle, GetThreeDFloorFillStyle, GetThreeDSideWallFillStyle, GetUpBarBorderStyle, GetUpBarFillStyle, GetXAxis, GetYAxis, HasCategoryScaleXAxis, HasValueScaleXAxis, HasXAxis, HasYAxis, InsertCategory, InsertSeries, IsBaseLayer, IsOverlayLayer, IsStacked, IsStudyLayer, PieExplosionTestValueIsPercentage, PlotBarsAsLines, PlotLinesBetweenPoints, PlotMarkersAtPoints, PlotUpDownBars, RemoveCategory, RemoveSeries, SetBarShape, SetBubbleSize, SetCategoryGapRatio, SetCategoryLabelFormat, SetChartType, SetDownBarBorderStyle, SetDownBarFillStyle, SetDropLineStyle, SetHighLowLineStyle, SetLineWidth, SetMarkerSize, SetMaximumNumberOfPoints, SetMaximumNumberOfPointsPerSeries, SetMaximumNumberOfSeries, SetMissingPoints, SetPieExplosion, SetPieExplosionAmount, SetPieExplosionTestOperator, SetPieExplosionTestValue, SetPieExplosionTestValueIsPercentage, SetPlotAreaBackgroundColor, SetPlotAreaBorderStyle, SetPlotAreaFillStyle, SetPlotBarsAsLines, SetPlotHighLowLines, SetPlotLinesBetweenPoints, SetPlotMarkersAtPoints, SetPlotUpDownBars, SetPointBorderStyle, SetPointLabelFormat, SetPointLabelLineStyle, SetPointLabelPlacement, SetPointLabelSource, SetPointLabelStyle, SetSeriesLabelFormat, SetSeriesOverlapRatio, SetSeriesPlacement, SetStartAngle, SetStockHasClose, SetStockHasOpen, SetStudyHeightRatio, SetThreeDFloorFillStyle, SetThreeDWallFillStyle, SetUpBarBorderStyle, SetUpBarFillStyle, StockHasClose, StockHasOpen

AcChartLayer::AddCategory method

Call AddCategory( ) to append a new category at the end of a chart layer's list of categories. When you add a category to a chart layer that already has a series, corresponding empty points are added to each of the chart layer's series.

You can call this method only on a chart's base layer.

All the layers in a chart share the same *x*-axis. This means that all the layers in a chart must have the same set of categories.

If a chart has an overlay layer, when you call AddCategory on the chart's base layer the new category is automatically duplicated in the chart's overlay layer.

If a chart has study layers, when you call AddCategory on the chart's base layer the new category is automatically duplicated in all the chart's study layers.

You cannot call this method on a scatter chart layer. Scatter chart layers do not have categories.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, after you have set the chart's status to ChartStatusBuilding

If you are adding categories and series to an empty chart layer, you must add at least one category before you add any series.

If you add categories to a chart layer using AddCategory( ), the categories appear on the chart in the order in which you added them. Categories you add using AddCategory( ) are not automatically sorted in any way.

The categoryLabelValue does not have to be a string. Label values are formatted into label text when the chart is viewed. This formatting allows locale-specific formatting. For example, if you set categoryLabelValue to 1.5, when the chart is viewed in the US English locale the label text is 1.5 but the label text is 1,5 when the chart is viewed in the French locale.

Syntax Function AddCategory(categoryKeyValue As Variant) As AcChartCategory

Function AddCategory(categoryKeyValue As Variant, categoryLabelValue As Variant) As AcChartCategory

Parameters **categoryKeyValue**
A unique identifying key value for the new category.

categoryLabelValue
A value to display as the label for the new category.

If this parameter is omitted, the category label value is the same as the category key value.

Returns A reference to the new category.

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

See also Class AcChart
Class AcChartCategory
AcChart::CustomizeCategoriesAndSeries method

AcChartLayer::AddSeries method
AcChartLayer::InsertCategory method
AcChartLayer::RemoveCategory method

AcChartLayer::AddSeries method

Call this method to append a new series to the end of a chart layer's list of series.

You can call AddSeries( ) on any layer in a chart.

All the layers in a chart share the same *x*-axis. All the layers in a chart do not necessarily have the same set of series.

You cannot call this method on a pie chart layer. Such layers only have one series.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, after you have set the chart's status to ChartStatusBuilding

If you are adding categories and series to an empty chart layer, you must add at least one category before you add any series.

If you add series to a chart layer using AddSeries( ), you must also populate those series with points. Points are not created automatically when you call AddSeries( ).

If you add series to a chart layer using AddSeries( ), the series appear on the chart in the order in which you added them. Series you add using AddSeries( ) will not be sorted automatically in any way.

The seriesLabelValue does not have to be a string. Label values are formatted into label text when the chart is viewed. This allows locale-specific formatting. For example, if you set seriesLabelValue to 1.5, when the chart is viewed in the US English locale the label text is 1.5 but the label text is 1,5 when the chart is viewed in the French locale.

Syntax

```
Function AddSeries( seriesKeyValue As Variant ) As AcChartSeries

Function AddSeries( seriesKeyValue As Variant, seriesLabelValue As
    Variant ) As AcChartSeries
```

Parameters

seriesKeyValue
A unique identifying key value for the new series.

seriesLabelValue
A value to be displayed as the label for the new series.
If this parameter is omitted, the series label value is the same as the series key value.

Returns A reference to the new series.

Example For an example of how to use this method, see the dynamic chart example for the AcChart class.

See also AcChart::CustomizeCategoriesAndSeries method
AcChartLayer::AddCategory method
AcChartLayer::InsertSeries method
AcChartLayer::RemoveSeries method
Class AcChart
Class AcChartSeries

AcChartLayer::ChartTypeIsStackable method

Specifies whether a chart layer's chart type supports stacked series. A typical stacked series chart type is a stacked bar chart. In a stacked bar chart layer, the points for all the series in the chart layer are stacked into a single bar in each category. The height of a bar is the sum of the values for all series in a category.

The following chart types support stacked series:

- Area
- Bar
- Line
- Step

Syntax Function ChartTypeIsStackable( ) As Boolean

Returns True if the chart layer's chart type supports stacked series.
False if the chart layer's chart type does not support stacked series.

See also AcChartLayer::SetChartType method
AcChartLayer::SetSeriesPlacement method

AcChartLayer::GetBarShape method

Returns the shape of bars in a three-dimensional bar chart layer. You can call this method only on a three-dimensional bar chart layer.

Syntax Function GetBarShape( ) As AcChartBarShape

Returns The shape of the bars in the chart layer.

See also AcChartBarShape
AcChartLayer::SetBarShape method

AcChartLayer::GetBubbleSize method

Returns the size of the largest bubble in a bubble chart as a percentage of the length of the shorter of the chart layer's two axes.

Syntax Function GetBubbleSize( ) As Double

Returns The size of the largest bubble.

See also AcChartLayer::SetBubbleSize method

AcChartLayer::GetCategory method

Returns a reference to the specified category in a chart layer. You can call this method only on a chart's base layer. Do not call this method on a scatter chart layer. Scatter chart layers do not have categories.

To determine the number of categories in a chart layer, call the chart layer's GetNumberOfCategories( ) method.

Syntax Function GetCategory(index As Integer) As AcChartCategory

Parameter **index**
An index into the chart layer's list of categories. The first category is index 1.

Returns A reference to the specified category in the chart layer.

See also Class AcChartCategory
AcChartLayer::GetNumberOfCategories method
AcChartLayer::GetSeries method

AcChartLayer::GetCategoryGapRatio method

Returns the size of the gap between categories in a bar chart layer, relative to the width of a single bar.

The size of the gap is defined relative to the width of a single bar. If the size of the gap is 1, it is the same width as a single bar. If the size of the gap is 2, it is twice the width of a single bar. If the size of the gap is 0.5, it is half the width of a single bar.

You can call this method only on a bar chart layer.

Syntax Function GetCategoryGapRatio( ) As Double

Returns The size of the gap between categories in the bar chart layer.

See also AcChartLayer::GetSeriesOverlapRatio method
AcChartLayer::SetCategoryGapRatio method

AcChartLayer::GetCategoryGrouping method

Returns a reference to the data grouping definition used to control how data is grouped into categories in a chart. You can then call methods on the data grouping definition object to change the way data is grouped into categories.

The category data grouping mechanism only works in charts that have a category key defined in Chart Builder.

You can change the way data is grouped only from within a chart's CustomizeLayers( ) method.

You cannot call this method on a scatter chart layer. Scatter chart layers do not have categories.

You can call this method only on a chart's base layer.

All the layers in a chart must have the same set of categories, so there is only one category grouping definition object in a chart. All the layers in a chart share the category grouping definition.

Syntax Function GetCategoryGrouping( ) As AcDataGrouping

Returns A reference to the data grouping definition used to group data into categories in the chart layer.

Example In the following example, a chart has a base stock chart layer. You defined the category in Chart Builder as a date. The chart's CustomizeLayers( ) method has been overridden to group category key values into weeks.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim categoryGrouping As AcDataGrouping
   Set categoryGrouping = baseLayer.GetCategoryGrouping( )
   categoryGrouping.SetUnit( DataGroupingUnitWeek )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetCategoryGrouping method

AcChartLayer::GetCategoryLabelFormat method

Returns the format pattern used to format category labels in a chart layer. You can call this method only on a chart's base layer.

Category labels are used as category scale axis labels. The value that this method returns is exactly the same as the value that the GetLabelFormat( ) method returns of a category scale axis.

Syntax Function GetCategoryLabelFormat( ) As String

Returns The format pattern used to format category labels in the chart layer.

See also AcChartAxis::GetLabelFormat method
AcChartLayer::GetPointLabelFormat method
AcChartLayer::GetSeriesLabelFormat method
AcChartLayer::SetCategoryLabelFormat method

AcChartLayer::GetChart method

Returns a reference to a chart layer's parent chart.

Syntax Function GetChart( ) As AcChart

Returns A reference to the chart layer's parent chart.

See also Class AcChart

AcChartLayer::GetChartType method

Returns the chart type of a chart layer.

Syntax Function GetChartType( ) As AcChartType

Returns The chart type of the chart layer.

See also AcChartType

AcChartLayer::GetDownBarBorderStyle method

Returns the style of the border around down bars in a chart layer. To change the border around down bars, call this method to retrieve the default settings.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

Syntax Function GetDownBarBorderStyle( ) As AcDrawingBorderStyle

Returns The style of the border around a down bar in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the border around down bars in the chart's base layer, depending on the value of a Boolean parameter. GetDownBarBorderStyle( ) retrieves the default settings so that only the border style's Color member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmRedOutlinedDownBars Then
      Dim borderStyle As AcDrawingBorderStyle
      borderStyle = baseLayer.GetDownBarBorderStyle( )
      borderStyle.Color = Red
      baseLayer.SetDownBarBorderStyle( borderStyle )
   End If
End Sub
```

See also AcChartLayer::GetDownBarFillStyle method
AcChartLayer::GetUpBarBorderStyle method
AcChartLayer::SetDownBarBorderStyle method
AcChartLayer::SetPlotUpDownBars method
AcDrawingBorderStyle

AcChartLayer::GetDownBarFillStyle method

Returns the fill style for down bars in a chart layer. To change the fill for down bars, call this method to retrieve the default settings.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

Syntax Function GetDownBarFillStyle( ) As AcDrawingFillStyle

Returns The fill style for a down bar in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of down bars in the chart's base layer, depending on the value of a Boolean parameter. GetDownBarFillStyle( ) retrieves the default settings so that only the fill style's Color1 member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmRedFilledDownBars Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetDownBarFillStyle( )
      fillStyle.Color = Red
      baseLayer.SetDownBarFillStyle( fillStyle )
   End If
End Sub
```

See also AcChartLayer::GetDownBarBorderStyle method
AcChartLayer::GetUpBarFillStyle method
AcChartLayer::SetDownBarFillStyle method
AcChartLayer::SetPlotUpDownBars method
AcDrawingFillStyle

AcChartLayer::GetDropLineStyle method

Returns the line style used to draw drop lines in a chart layer. To change the style of drop lines, call this method to retrieve the default settings.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

Syntax Function GetDropLineStyle( ) As AcDrawingLineStyle

Returns The line style used to draw drop lines in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the pattern used to draw drop lines in the chart's base layer, depending on the value of a Boolean parameter. GetDropLineStyle( ) retrieves the default settings so that only the line style's Pen member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmDottedDropLines Then
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = baseLayer.GetDropLineStyle( )
      lineStyle.Pen = DrawingLinePenDot
      baseLayer.SetDropLineStyle( lineStyle )
   End If
End Sub
```

See also AcChartLayer::SetDropLineStyle method
AcDrawingLineStyle

AcChartLayer::GetHighLowLineStyle method

Returns the line style used to draw high-low lines in a chart layer. To change the style of high-low lines, call this method to retrieve the default settings.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

Syntax Function GetHighLowLineStyle( ) As AcDrawingLineStyle

Returns The line style used to draw high-low lines in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the thickness of high-low lines in the chart's base layer, depending on the value of a Boolean parameter. GetHighLowLineStyle( ) retrieves the default settings so that only the line style's Width member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmThickHighLowLines Then
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = baseLayer.GetHighLowLineStyle( )
      lineStyle.Width = 2 * OnePoint
      baseLayer.SetHighLowLineStyle( lineStyle )
   End If
End Sub
```

See also AcChartLayer::SetHighLowLineStyle method
AcDrawingLineStyle

AcChartLayer::GetIndex method

Returns the index of a chart layer within its parent chart's list of layers. The first layer in a chart is index 1.

Syntax Function GetIndex( ) As Integer

Returns The index of the chart layer within its parent chart's list of layers.

AcChartLayer::GetLayerType method

Returns the chart layer type of a chart layer.

Syntax Function GetLayerType( ) As AcChartLayerType

Returns The chart layer type of the chart layer.

See also AcChartLayerType

AcChartLayer::GetLineWidth method

Returns the default width of the lines joining points within each series in a chart layer. You can call this method only on chart layers of the following chart types:

- Stacked bar
- Line
- Scatter

The line width this method returns might not apply to all the series in a chart layer. You can retrieve the width of the line for an individual series by calling the corresponding series style's GetLineStyle( ) method.

Syntax Function GetLineWidth( ) As AcTwips

Returns The default width of the lines joining points within each series in the chart layer.

See also AcChartLayer::PlotLinesBetweenPoints method
AcChartLayer::SetLineWidth method
AcChartSeriesStyle::GetLineStyle method
AcTwips

AcChartLayer::GetMarkerSize method

Returns the default size for markers within a chart layer. You can call this method only on chart layers with the following chart types:

- Line
- Scatter
- Stock

The marker size this method returns might not apply to all the points in a chart layer. To retrieve the default size for markers in an individual series, call the corresponding series style's GetMarkerSize( ) method. To retrieve the size of the marker for an individual point, call the corresponding point style's GetMarkerSize( ) method.

Syntax Function GetMarkerSize( ) As AcTwips

Returns The default size for markers within the chart layer.

See also AcChartLayer::SetMarkerSize method
AcChartPointStyle::GetMarkerSize method
AcTwips

AcChartLayer::GetMaximumDataXValue method

Returns the maximum x value of all the points in a chart layer. You can call this method only on scatter chart layers.

You can call this method only after the chart has computed its minimum and maximum data values. You can call GetMaximumDataXValue( ) from the following methods:

- AcChart::CustomizeAxes( )
- AcChart::AdjustChart( )

Syntax Function GetMaximumDataXValue( ) As Variant

Returns The maximum x value of all the points in a chart layer.

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::ComputeMinMaxDataValues method
AcChartLayer::GetMaximumDataYValue method
AcChartLayer::GetMinimumDataXValue method

AcChartLayer::GetMaximumDataYValue method

Returns the maximum y value of all the points in a chart layer. In a pie chart layer, the y values are the slice values.

You can call this method only after the chart has computed its minimum and maximum data values. You can call GetMaximumDataYValue( ) from the following methods:

- AcChart::CustomizeAxes( )
- AcChart::AdjustChart( )

Syntax Function GetMaximumDataYValue( ) As Variant

Returns The maximum y value of all the points in the chart layer.

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::ComputeMinMaxDataValues method
AcChartLayer::GetMaximumDataXValue method
AcChartLayer::GetMaximumTrendlineYValue method
AcChartLayer::GetMinimumDataYValue method

AcChartLayer::GetMaximumNumberOfPoints method

Returns the maximum number of points permitted in a chart layer. This number is a safety limit used to prevent charts from growing excessively large due to programming errors or unexpected data. If you exceed this limit, a run-time error occurs.

Syntax Function GetMaximumNumberOfPoints( ) As Integer

Returns The maximum number of points permitted in the chart layer.

See also AcChartLayer::GetMaximumNumberOfPointsPer Series method
AcChartLayer::GetMaximumNumberOfSeries method
AcChartLayer::SetMaximumNumberOfPoints method

AcChartLayer::GetMaximumNumberOfPointsPer Series method

Returns the maximum number of points permitted in a single series in a chart layer. This number is a safety limit used to prevent charts from growing excessively large due to programming errors or unexpected data. If you exceed this limit, a run-time error occurs.

In charts with a category axis, the maximum number of points per series is equivalent to the maximum number of categories.

Syntax Function GetMaximumNumberOfPointsPerSeries( ) As Integer

Returns The maximum number of points permitted in a single series in the chart layer.

See also AcChartLayer::GetMaximumNumberOfPoints method
AcChartLayer::GetMaximumNumberOfSeries method
AcChartLayer::SetMaximumNumberOfPointsPer Series method

AcChartLayer::GetMaximumNumberOfSeries method

Returns the maximum number of series permitted in a chart layer. This number is a safety limit used to prevent charts from growing excessively large due to programming errors or unexpected data. If you exceed this limit, a run-time error occurs.

Syntax Function GetMaximumNumberOfSeries( ) As Integer

Returns The maximum number of series permitted in the chart layer.

See also AcChartLayer::GetMaximumNumberOfPoints method
AcChartLayer::GetMaximumNumberOfPointsPer Series method
AcChartLayer::SetMaximumNumberOfSeries method

AcChartLayer::GetMaximumTrendlineYValue method

Returns the maximum y value of all the trendlines in a chart layer.

You can only call this method after the chart has computed its trendlines.

You can call this method from the following methods:

- AcChart::CustomizeAxes( )
- AcChart::AdjustChart( )

Syntax Function GetMaximumTrendlineYValue( ) As Variant

Returns The maximum y value of all the trendlines in the chart layer.
Null if the chart layer does not contain any trendlines.

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::ComputeMinMaxDataValues method
AcChartAxis::GetMaximumTrendlineValue method
AcChartLayer::GetMaximumDataYValue method
AcChartLayer::GetMinimumTrendlineYValue method
Class AcChartTrendline

AcChartLayer::GetMinimumDataXValue method

Returns the minimum x value of all the points in a chart layer. You can only call this method on scatter chart layers.

You can only call this method after the chart has computed its minimum and minimum data values. You can call GetMaximumDataYValue( ) from the following methods:

- AcChart::CustomizeAxes( )
- AcChart::AdjustChart( )

Syntax Function GetMinimumDataXValue( ) As Variant

Returns The minimum x value of all the points in the chart layer.

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::ComputeMinMaxDataValues method

AcChartLayer::GetMaximumDataXValue method
AcChartLayer::GetMinimumDataYValue method

AcChartLayer::GetMinimumDataYValue method

Returns the minimum y value of all the points in a chart layer. In a pie chart layer, the y values are the slice values.

You can only call this method after the chart has computed its minimum and minimum data values.

You can call GetMinimumDataYValue( ) from the following methods:

- AcChart::CustomizeAxes( )
- AcChart::AdjustChart( )

Syntax Function GetMinimumDataYValue( ) As Variant

Returns The minimum y value of all the points in the chart layer.

See also AcChart::AdjustChart method
AcChart::ComputeMinMaxDataValues method
AcChartLayer::GetMaximumDataYValue method
AcChartLayer::GetMaximumTrendlineYValue method
AcChartLayer::GetMinimumDataXValue method
AcChartLayer::GetMinimumTrendlineYValue method

AcChartLayer::GetMinimumTrendlineYValue method

Returns the minimum y value of all the trendlines in a chart layer.

You can only call this method after the chart has computed its trendlines. You can call this method from the following methods:

- AcChart::CustomizeAxes( )
- AcChart::AdjustChart( )

Syntax Function GetMinimumTrendlineYValue( ) As Variant

Returns The minimum y value of all the trendlines in the chart layer.
Null if the chart layer does not contain any trendlines.

See also AcChart::AdjustChart method
AcChart::CustomizeAxes method
AcChart::ComputeMinMaxDataValues method
AcChartAxis::GetMinimumTrendlineValue method
AcChartLayer::GetMinimumDataYValue method
AcChartLayer::GetMaximumTrendlineYValue method
AcChartLayer::GetMinimumDataYValue method

Class AcChartTrendline

AcChartLayer::GetMissingPoints method

Returns the way that missing points are plotted in a chart layer.

Syntax Function GetMissingPoints( ) As AcChartMissingPoints

Returns The way that missing points are plotted in the chart layer.

See also AcChartLayer::SetMissingPoints method
AcChartMissingPoints

AcChartLayer::GetNumberOfCategories method

Returns the number of categories in a chart layer.

You cannot call this method on a scatter chart layer. Scatter chart layers do not have categories.

You can only call this method on a chart's base layer.

Syntax Function GetNumberOfCategories( ) As Integer

Returns The number of categories in the chart layer.

See also AcChartLayer::GetCategory method
AcChartLayer::GetNumberOfSeries method

AcChartLayer::GetNumberOfSeries method

Returns the number of series in a chart layer.

Syntax Function GetNumberOfSeries( ) As Integer

Returns The number of series in the chart layer.

See also AcChartLayer::GetNumberOfCategories method
AcChartLayer::GetSeries method

AcChartLayer::GetPieCenter method

Returns the position of the center of a pie chart relative to the top left corner of its parent chart's chart drawing plane.

You can use this method only for two-dimensional pie charts.

You can call this method only from the AcChart::DrawOnChart( ) method. You must call the AcChartDescribeLayout( ) method before calling this method.

Syntax Function GetPieCenter( ) As AcPoint

Returns The position of the center of the pie chart relative to the top left corner of its parent chart's chart drawing plane.

See also AcChart::DescribeLayout method
AcChart::DrawOnChart method
AcChartLayer::GetPieRadius method
AcChartLayer::GetPlotAreaPosition method
AcChartLayer::GetPlotAreaSize method
Class AcDrawingChartPlane
AcRectangle

AcChartLayer::GetPieExplosion method

Returns the circumstances in which pie slices will be exploded in a pie chart layer.

You can call this method only on a pie chart layer.

Syntax Function GetPieExplosion( ) As AcChartPieExplode

Returns The circumstances in which pie slices will be exploded in the pie chart layer.

See also AcChartLayer::GetPieExplosionAmount method
AcChartLayer::GetPieExplosionTestOperator method
AcChartLayer::GetPieExplosionTestValue method
AcChartLayer::PieExplosionTestValueIsPercentage method
AcChartLayer::SetPieExplosion method
AcChartPieExplode

AcChartLayer::GetPieExplosionAmount method

Returns the amount that pie slices will be exploded in a pie chart layer. The amount is a proportion of the radius of the pie. If the amount is 0.25, exploded slices will be moved outwards from the center of the pie by one quarter of the radius of the pie.

You can only call this method on a pie chart layer.

Syntax Function GetPieExplosionAmount( ) As Double

Returns The amount that pie slices will be exploded in the pie chart layer.

See also AcChartLayer::GetPieExplosion method
AcChartLayer::GetPieExplosionTestOperator method
AcChartLayer::GetPieExplosionTestValue method
AcChartLayer::PieExplosionTestValueIsPercentage method
AcChartLayer::SetPieExplosionAmount method

AcChartLayer::GetPieExplosionTestOperator method

Returns the operator used to test whether a pie slice will be exploded in a pie chart layer.

You can only call this method on a pie chart layer.

Syntax Function GetPieExplosionTestOperator( ) As AcChartComparisonOperator

Returns The operator used to test whether a pie slice will be exploded in a pie chart layer.

See also AcChartComparisonOperator
AcChartLayer::GetPieExplosion method
AcChartLayer::GetPieExplosionAmount method
AcChartLayer::GetPieExplosionTestValue method
AcChartLayer::PieExplosionTestValueIsPercentage method
AcChartLayer::SetPieExplosionTestOperator method

AcChartLayer::GetPieExplosionTestValue method

Returns the value used to test whether a pie slice will be exploded in a pie chart layer.

You can only call this method on a pie chart layer.

Syntax Function GetPieExplosionTestValue( ) As Variant

Returns The value used to test whether to explode a pie slice in the pie chart layer.

See also AcChartLayer::GetPieExplosion method
AcChartLayer::GetPieExplosionAmount method
AcChartLayer::GetPieExplosionTestOperator method
AcChartLayer::PieExplosionTestValueIsPercentage method
AcChartLayer::SetPieExplosionTestValue method

AcChartLayer::GetPieRadius method

Returns the radius of a pie chart.

You can use this method only for two-dimensional pie charts.

You can call this method only from the AcChart::DrawOnChart( ) method.

You must call the AcChartDescribeLayout( ) method before calling this method.

Syntax Function GetPieRadius( ) As AcTwips

Returns The radius of the pie chart.

See also AcChart::DescribeLayout method
AcChart::DrawOnChart method
AcChartLayer::GetPieCenter method

AcChartLayer::GetPlotAreaPosition method
AcChartLayer::GetPlotAreaSize method
Class AcDrawingChartPlane
AcTwips

AcChartLayer::GetPlotAreaBorderStyle method

Returns the style of the border around a chart layer's plot area. To change the border around a chart layer's plot area, call this method to retrieve the default settings.

You can only call this method on a chart's base layer.

All the layers in a chart are drawn with the same plot area border style as the base layer. You cannot change the plot area border style on individual layers.

You cannot call this method on a three-dimensional chart layer. A three-dimensional chart layer does not have a plot area border.

You cannot call this method on a pie chart layer. A pie chart layer does not have a plot area border.

Syntax Function GetPlotAreaBorderStyle( ) As AcDrawingBorderStyle

Returns The style of the border around the chart layer's plot area.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the border around the chart's base layer's plot area, based on the value of a parameter. GetPlotAreaBorderStyle( ) retrieves the default settings so that only the border style's Color member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim borderStyle As AcDrawingBorderStyle
   borderStyle = baseLayer.GetPlotAreaBorderStyle( )
   borderStyle.Color = parmPlotAreaBorderColor
   baseLayer.SetPlotAreaBorderStyle( borderStyle )
End Sub
```

See also AcChart::GetBorderStyle method
AcChartLayer::SetPlotAreaBorderStyle method
AcDrawingBorderStyle

AcChartLayer::GetPlotAreaFillStyle method

Returns the background fill style for a chart layer's plot area. To change the background of a chart layer's plot area, call this method to retrieve the default settings.

You can only call this method on a chart's base layer.

All the layers in a chart are drawn with the same plot area fill style as the base layer. You cannot change the plot area fill style on individual layers.

You cannot call this method on a three-dimensional chart layer. A three-dimensional chart layer has separate fill styles for its walls and its floor instead of a plot area fill style.

You cannot call this method on a pie chart layer. A pie chart layer does not have a plot area fill style.

Syntax Function GetPlotAreaFillStyle( ) As AcDrawingFillStyle

Returns The background fill style for the chart layer's plot area.

Example The following example overrides a chart's CustomizeLayers( ) method to create a patterned plot area background, depending on the value of a Boolean parameter. GetPlotAreaFillStyle( ) retrieves the default settings so that only the fill style's Pattern member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAddBackgroundPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetPlotAreaFillStyle( )
      fillStyle.Pattern = DrawingFillPattern05Percent
      baseLayer.SetPlotAreaFillStyle( fillStyle )
   End If
End Sub
```

See also AcChart::GetFillStyle method
AcChartLayer::GetThreeDBackWallFillStyle method
AcChartLayer::GetThreeDFloorFillStyle method
AcChartLayer::GetThreeDSideWallFillStyle method
AcChartLayer::SetPlotAreaBackgroundColor method
AcChartLayer::SetPlotAreaFillStyle method
AcDrawingFillStyle

AcChartLayer::GetPlotAreaPosition method

Returns the position of a chart layer's plot area relative to the top left corner of its parent chart's chart drawing plane. You can use this method only for two-dimensional charts that are not pie charts. You can call this method only from the AcChart::DrawOnChart( ) method. You must call the AcChartDescribeLayout( ) method before calling this method.

Syntax Function GetPlotAreaPosition( ) As AcPoint

Returns The position of the chart layer's plot area relative to the top left corner of its parent chart's chart drawing plane.

Example For an example of how to use this method, see the example for the AcChart::DrawOnChart( ) method.

See also AcChart::DescribeLayout method
AcChart::DrawOnChart method
AcChartLayer::GetPieCenter method
AcChartLayer::GetPieRadius method
AcChartLayer::GetPlotAreaSize method
Class AcDrawingChartPlane
AcRectangle

AcChartLayer::GetPlotAreaSize method

Returns the size of a chart layer's plot area. You can use this method only for two-dimensional charts that are not pie charts. You can call this method only from the AcChart::DrawOnChart( ) method. You must call the AcChartDescribeLayout( ) method before calling this method.

Syntax Function GetPlotAreaSize( ) As AcSize

Returns The size of the chart layer's plot area.

Example For an example of how to use this method, see the example for the AcChart::DrawOnChart( ) method.

See also AcChart::DescribeLayout method
AcChart::DrawOnChart method
AcChartLayer::GetPieCenter method
AcChartLayer::GetPieRadius method
AcChartLayer::GetPlotAreaPosition method
Class AcDrawingChartPlane
AcSize

AcChartLayer::GetPointBorderStyle method

Returns the default style for the borders around points in a chart layer. To change the border around a chart layer's points, call this method to retrieve the default settings.

You can only call this method on chart layers with these chart types:

- Area
- Bar
- Pie
- Step

The border style that this method returns might not apply to all the points in a chart layer. To retrieve the default border style for the points within an individual series, call the corresponding series style's GetBorderStyle( ) method. To retrieve the border style for an individual point, call the corresponding point style's GetBorderStyle( ) method.

Syntax Function GetPointBorderStyle( ) As AcDrawingBorderStyle

Returns The default style for the borders around points in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the border around points in the chart's base layer, based on the value of a parameter. GetPointBorderStyle( ) retrieves the default settings so that only the border style's Color member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim borderStyle As AcDrawingBorderStyle
   borderStyle = baseLayer.GetPointBorderStyle( )
   borderStyle.Color = parmPointBorderColor
   baseLayer.SetPointBorderStyle( borderStyle )
End Sub
```

See also AcChartLayer::SetPointBorderStyle method
AcChartPointStyle::GetBorderStyle method
AcDrawingBorderStyle

AcChartLayer::GetPointLabelFormat method

Returns the default format pattern used to format point labels in a chart layer. The format pattern that this method returns might not apply to all the points in a chart layer. To retrieve the point label format pattern for an individual series, call the corresponding series style's GetPointLabelFormat( ) method. To retrieve the point label format pattern for an individual point, call the point's GetCustomLabelFormat( ) method.

Syntax Function GetPointLabelFormat( ) As String

Returns The default format pattern used to format point labels in the chart layer.

See also AcChartLayer::GetCategoryLabelFormat method
AcChartLayer::GetSeriesLabelFormat method
AcChartLayer::GetPointLabelFormat method
AcChartPoint::GetCustomLabelFormat method
AcChartPoint::HasCustomLabelFormat method
AcChartSeriesStyle::GetPointLabelFormat method

AcChartLayer::GetPointLabelLineStyle method

Returns the line style used to draw point label lines in a chart layer. To change line style used to draw point label lines in a chart layer, call this method to retrieve the default settings. You can only call this method on pie chart layers.

Syntax Function GetPointLabelLineStyle( ) As AcDrawingLineStyle

Returns The line style used to draw point label lines in a chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the pattern used to draw the point label lines in the chart's base layer, depending on the value of a Boolean parameter. GetPointLabelLineStyle( ) retrieves the default settings so that only the line style's Pen member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmDottedPointLabelLines Then
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = baseLayer.GetPointLabelLineStyle( )
      lineStyle.Pen = DrawingLinePenDot
      baseLayer.SetPointLabelLineStyle( lineStyle )
   End If
End Sub
```

See also AcChartLayer::SetPointLabelStyle method
AcDrawingLineStyle

AcChartLayer::GetPointLabelPlacement method

Returns the default placement of point labels in a chart layer. The placement that this method returns might not apply to all the points in a chart layer. To retrieve the default point label placement for the points within an individual series, call the corresponding series style's GetPointLabelPlacement( ) method. To retrieve the point label placement for an individual point, call the corresponding point style's GetPointLabelPlacement( ) method.

Syntax Function GetPointLabelPlacement( ) As AcChartPointLabelPlacement

Returns The default placement of point labels in a chart layer.

See also AcChartPointLabelPlacement
AcChartPointStyle::SetPointLabelPlacement method

AcChartLayer::GetPointLabelSource method

Returns the default source for point label values in a chart layer. The source that this method returns might not apply to all the points in a chart layer. To retrieve the point label source for an individual series, call the corresponding series style's GetPointLabelSource( ) method. To retrieve the point label value for an individual point, call the point's GetCustomLabelValue( ) method.

Syntax Function GetPointLabelSource( ) As AcChartPointLabelSource

Returns The default source for point label values in the chart layer.

See also AcChartLayer::SetPointLabelSource method

AcChartPoint::GetCustomLabelValue method
AcChartPoint::HasCustomLabelFormat method
AcChartPointLabelSource
AcChartSeriesStyle::GetPointLabelSource method

AcChartLayer::GetPointLabelStyle method

Returns the default style for point labels in a chart layer. To change the default style of a chart layer's point labels, call this method to retrieve the default settings.

The style that this method returns might not apply to all the points in a chart layer. To retrieve the default point label style for the points within an individual series, call the corresponding series style's GetPointLabelStyle( ) method. To retrieve the point label style for an individual point, call the corresponding point style's GetPointLabelStyle( ) method.

Syntax Function GetPointLabelStyle( ) As AcDrawingTextStyle

Returns The default style for point labels in a chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to make point labels italic in the chart's base layer, depending on the value of a Boolean parameter. GetPointLabelStyle( ) retrieves the default settings so that only the text style's Font member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim textStyle As AcDrawingTextStyle
   textStyle = baseLayer.GetPointLabelStyle( )
   textStyle.Font.Italic = parmItalicPointLabels
   baseLayer.SetPointLabelStyle( textStyle )
End Sub
```

See also AcChartLayer::SetPointLabelStyle method
AcChartPointStyle::GetPointLabelStyle method
AcDrawingTextStyle

AcChartLayer::GetSeries method

Returns a reference to the specified series in a chart layer. To determine the number of series in a chart layer, call the chart layer's GetNumberOfSeries( ) method.

Syntax Function GetSeries(index As Integer) As AcChartSeries

Parameter **index**
An index into the chart layer's list of series. The first series is index 1.

Returns A reference to the specified series in the chart layer.

See also AcChartLayer::GetCategory method
AcChartLayer::GetNumberOfSeries method
Class AcChartSeries

AcChartLayer::GetSeriesGrouping method

Returns a reference to the data grouping definition used to control how data is grouped into series in a chart layer. You can then call methods on the data grouping definition object to change the way data is grouped into series.

The series data grouping mechanism only works in chart layers that have a series key defined in Chart Builder.

You can change the way data is grouped only from within a chart's CustomizeLayers( ) method.

Syntax Function GetSeriesGrouping( ) As AcDataGrouping

Returns A reference to the data grouping definition used to group data into series in the chart layer.

Example In the following example, you defined the series key for a chart's base layer in Chart Builder as a date. The example overrides the chart's CustomizeLayers( ) method to group series key values into calendar quarters.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim seriesGrouping As AcDataGrouping
   Set seriesGrouping = baseLayer.GetSeriesGrouping( )
   ' Enable grouping.
   seriesGrouping.Mode = DataGroupingModeInterval
   ' Group into calendar quarters.
   seriesGrouping.SetUnit( DataGroupingUnitQuarter )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetCategoryGrouping method

AcChartLayer::GetSeriesLabelFormat method

Returns the format pattern used to format series labels in a chart layer.

Syntax Function GetSeriesLabelFormat( ) As String

Returns The format pattern used to format series labels in the chart layer.

See also AcChartLayer::GetCategoryLabelFormat method
AcChartLayer::GetPointLabelFormat method

AcChartLayer::GetSeriesOverlapRatio method

Returns the amount by which adjacent series in a bar chart can overlap, relative to the width of a single bar.

The amount of overlap is defined relative to the width of a single bar. If the amount of overlap is 0.5, adjacent bars overlap by half the width of a single bar.

Negative overlaps are permitted. If the amount of overlap is -0.5, there is a gap half the width of a single bar between adjacent bars.

You can only call this method on a two-dimensional bar chart layer.

Syntax Function GetSeriesOverlapRatio( ) As Double

Returns The amount that adjacent series in the bar chart layer will overlap, relative to the width of a single bar.
Negative values mean there is a gap instead of an overlap.

See also AcChartLayer::GetCategoryGapRatio method
AcChartLayer::SetSeriesOverlapRatio method

AcChartLayer::GetSeriesPlacement method

Returns the relative placement of points for multiple series within a category in a chart layer. You cannot call this method on chart layers with the following chart types:

- Pie
- Scatter
- Stock

Syntax Function GetSeriesPlacement( ) As AcChartSeriesPlacement

Returns The relative placement of points for multiple series within a category in a chart layer.

See also AcChartLayer::IsStacked method
AcChartLayer::SetChartType method
AcChartLayer::SetSeriesPlacement method
AcChartSeriesPlacement

AcChartLayer::GetSeriesStyle method

Returns a reference to the specified series style in a chart layer. You can then call methods on the series style object to change the appearance of the corresponding series in the chart layer.

To determine the number of series styles in a chart layer that is not a pie chart layer, call the chart layer’s GetNumberOfSeries( ) method.

A pie chart layer has only one series. Each slice in a pie corresponds to a category, not a series. For pie chart layers, series styles are used for pie slices. To determine the number of series styles in a pie chart layer, call the chart layer's GetNumberOfCategories( ) method.

The recommended method in which to modify series styles is a chart's CustomizeSeriesStyles( ) method.

Syntax Function GetSeriesStyle(index As Integer) As AcChartSeriesStyle

Parameter **index**
An index into the chart layer's list of series styles. The first series style is index 1.

Returns A reference to the specified series style in the chart layer.

Example The following example overrides a chart's CustomizeSeriesStyles( ) method to change the fill patterns of all the series in the chart's base layer, depending on the value of a Boolean parameter. Each series styles' GetFillStyle( ) method retrieves the default settings for that series so that only the fill style's Pattern member needs to change.

```
Sub CustomizeSeriesStyles( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim numberOfSeries As Integer
   numberOfSeries = baseLayer.GetNumberOfSeries( )
   Dim seriesIndex As Integer
   For seriesIndex = 1 To numberOfSeries
      Dim seriesStyle As AcChartSeriesStyle
      Set seriesStyle = baseLayer.GetSeriesStyle( seriesIndex )
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = seriesStyle.GetFillStyle( )
      fillStyle.Pattern = DrawingFillPattern05Percent
      seriesStyle.SetFillStyle( fillStyle )
   Next seriesIndex
End Sub
```

See also AcChart::CustomizeSeriesStyles method
AcChartLayer::GetNumberOfCategories method
AcChartLayer::GetNumberOfSeries method
Class AcChartSeriesStyle

AcChartLayer::GetStartAngle method

Returns the angle at which the first slice in a pie chart layer is drawn. The angle is measured in degrees clockwise from vertical.

You can only call this method on a pie chart layer.

Syntax Function GetStartAngle( ) As AcAngle

Returns The angle at which the first slice in the pie chart layer is drawn.

See also AcChartLayer::SetStartAngle method

AcChartLayer::GetStudyHeightRatio method

Returns the ratio of the height of a study layer to the height of its parent chart's base layer. For example, if the study layer is half the height of the base layer, this method returns 0.5.

You can only call this method on a study layer.

Syntax Function GetStudyHeightRatio( ) As Double

Returns The ratio of the height of the study layer to the height of its parent chart's base layer.

See also AcChartLayer::SetStudyHeightRatio method

AcChartLayer::GetThreeDBackWallFillStyle method

Returns the background fill style for a three-dimensional chart's back wall. To change the background of a three-dimensional chart's walls, call this method to get the default settings.

You can only call this method on a chart's base layer.

You can only call this method on a three-dimensional chart layer.

You cannot call this method on a three-dimensional pie chart layer. A three-dimensional pie chart layer does not have walls or a floor.

The back wall and side wall of a three-dimensional chart layer always have the same fill styles.

Syntax Function GetThreeDBackWallFillStyle( ) As AcDrawingFillStyle

Returns The background fill style of the chart layer's back wall.

Example The following example overrides a chart's CustomizeLayers( ) method to create patterned walls, depending on the value of a Boolean parameter. GetThreeDBackWallFillStyle( ) retrieves the default settings so that only the fill style's Pattern member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAddBackgroundPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetThreeDBackWallFillStyle( )
      fillStyle.Pattern = DrawingFillPattern20Percent
      baseLayer.SetThreeDWallFillStyle( fillStyle )
   End If
End Sub
```

See also AcChartLayer::GetPlotAreaFillStyle method
AcChartLayer::GetThreeDFloorFillStyle method
AcChartLayer::GetThreeDSideWallFillStyle method
AcChartLayer::SetThreeDWallFillStyle method
AcDrawingFillStyle

AcChartLayer::GetThreeDFloorFillStyle method

Returns the background fill style for a three-dimensional chart's floor. To change the background of a three-dimensional chart's floor, call this method to get the default settings.

You can only call this method on a chart's base layer.

You can only call this method on a three-dimensional chart layer.

You cannot call this method on a three-dimensional pie chart layer. A three-dimensional pie chart layer does not have walls or a floor.

Syntax Function GetThreeDFloorFillStyle( ) As AcDrawingFillStyle

Returns The background fill style of the chart layer's floor.

Example The following example overrides a chart's CustomizeLayers( ) method to create a patterned floor, depending on the value of a Boolean parameter. GetThreeDFloorFillStyle( ) retrieves the default settings so that only the fill style's Pattern member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAddFloorPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetThreeDFloorFillStyle( )
      fillStyle.Pattern = DrawingFillPatternBrickHorizontal
      baseLayer.SetThreeDFloorFillStyle( fillStyle )
   End If
End Sub
```

See also AcChartLayer::GetPlotAreaFillStyle method
AcChartLayer::GetThreeDBackWallFillStyle method
AcChartLayer::GetThreeDSideWallFillStyle method
AcChartLayer::SetThreeDFloorFillStyle method
AcDrawingFillStyle

AcChartLayer::GetThreeDSideWallFillStyle method

Returns the background fill style for a three-dimensional chart's side wall. To change the background of a three-dimensional chart's walls, call this method to get the default settings.

You can only call this method on a chart's base layer.

You can only call this method on a three-dimensional chart layer.

You cannot call this method on a three-dimensional pie chart layer. A three-dimensional pie chart layer does not have walls or a floor.

The back wall and side wall of a three-dimensional chart layer always have the same fill styles.

Syntax Function GetThreeDSideWallFillStyle( ) As AcDrawingFillStyle

Returns The background fill style of the chart layer's side wall.

Example The following example overrides a chart's CustomizeLayers( ) method to create patterned walls, depending on the value of a Boolean parameter. GetThreeDSideWallFillStyle( ) retrieves the default settings so that only the fill style's Pattern member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAddWallPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetThreeDSideWallFillStyle( )
      fillStyle.Pattern = DrawingFillPattern20Percent
      baseLayer.SetThreeDWallFillStyle( fillStyle )
   End If
End Sub
```

See also AcChartLayer::GetPlotAreaFillStyle method
AcChartLayer::GetThreeDBackWallFillStyle method
AcChartLayer::GetThreeDFloorFillStyle method
AcChartLayer::SetThreeDWallFillStyle method
AcDrawingFillStyle

AcChartLayer::GetUpBarBorderStyle method

Returns the style of the border around an up bar in a chart layer. To change the border around up bars, call this method to get the default settings.

You can only call this method on chart layers with the following chart types:

- Line
- Stock

Syntax Function GetUpBarBorderStyle( ) As AcDrawingBorderStyle

Returns The style of the border around an up bar in the chart.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the border around the chart's base layer's up bars, depending on the value of a Boolean parameter. GetUpBarBorderStyle( ) retrieves the default settings so that only the border style's Color member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmGreenOutlinedUpBars Then
      Dim borderStyle As AcDrawingBorderStyle
      borderStyle = baseLayer.GetUpBarBorderStyle( )
      borderStyle.Color = Green
      baseLayer.SetUpBarBorderStyle( borderStyle )
   End If
End Sub
```

See also AcChartLayer::GetDownBarBorderStyle method
AcChartLayer::GetUpBarFillStyle method
AcChartLayer::SetUpBarBorderStyle method
AcChartLayer::SetPlotUpDownBars method
AcDrawingBorderStyle

AcChartLayer::GetUpBarFillStyle method

Returns the fill style for an up bar in an chart layer. To change the fill for up bars, call this method to retrieve the default settings. You can only call this method on chart layers with the following chart types:

- Line
- Stock

Syntax Function GetUpBarFillStyle( ) As AcDrawingFillStyle

Returns The fill style for an up bar in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the chart's base layer's up bars, depending on the value of a Boolean parameter. GetUpBarFillStyle( ) retrieves the default settings so that only the fill style's Color1 member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmGreenFilledUpBars Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetUpBarFillStyle( )
      fillStyle.Color = Green
      baseLayer.SetUpBarFillStyle( fillStyle )
   End If
End Sub
```

See also AcChartLayer::GetDownBarFillStyle method
AcChartLayer::GetUpBarBorderStyle method
AcChartLayer::SetUpBarFillStyle method

AcChartLayer::SetPlotUpDownBars method
AcDrawingFillStyle

AcChartLayer::GetXAxis method

Returns a reference to a chart layer's *x*-axis. You can then call methods on the axis object to change the behavior and appearance of the axis. The recommended method in which to modify axes is a chart's CustomizeAxes( ) method.

You can only call this method on a chart's base layer. Only the base layer of a chart has an *x*-axis. All the other layers in a chart use the base layer's *x*-axis.

Pie chart layers do not have any axes.

Syntax Function GetXAxis( ) As AcChartAxis

Returns A reference to the chart layer's *x*-axis.
Nothing if the chart layer does not have an *x*-axis.

Example The following example overrides a chart's CustomizeAxes( ) method to change the thickness of the chart's base layer's *x*-axis, depending on the value of a Boolean parameter. The *x*-axis GetLineStyle( ) method retrieves the default settings so that only the line style's Width member needs to change.

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmThickXAxis Then
      Dim xAxis As AcChartAxis
      Set xAxis = baseLayer.GetXAxis( )
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = xAxis.GetLineStyle( )
      lineStyle.Width = 2 * OnePoint
      xAxis.SetLineStyle( lineStyle )
   End If
End Sub
```

See also AcChart::CustomizeAxes method
AcChartLayer::GetYAxis method
AcChartLayer::HasXAxis method
Class AcChartAxis

AcChartLayer::GetYAxis method

Returns a reference to a chart layer's *y*-axis. You can then call methods on the axis object to change the behavior and appearance of the axis. The recommended method in which to modify axes is a chart's CustomizeAxes( ) method.

Pie chart layers do not have axes.

Syntax Function GetYAxis( ) As AcChartAxis

Returns A reference to the chart layer's *y*-axis.
Nothing if the chart layer does not have a *y*-axis.

Examples The following example overrides a chart's CustomizeAxes( ) method to show minor ticks on the chart's base layer's *y*-axis, depending on the value of a Boolean parameter:

```
Sub CustomizeAxes( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmShowMinorTicks Then
      Dim yAxis As AcChartAxis
      Set yAxis = baseLayer.GetYAxis( )
      yAxis.SetMinorTickPlacement( ChartTickPllacementOutside )
   End If
End Sub
```

For another example of how to use this method, see the dynamic chart example for the AcChart class.

See also AcChart::CustomizeAxes method
AcChartLayer::GetXAxis method
AcChartLayer::HasYAxis method
Class AcChartAxis

AcChartLayer::HasCategoryScaleXAxis method

Determines whether a chart layer's *x*-axis is a category scale axis. You can only call this method on a chart's base layer. Only the base layer of a chart has an *x*-axis. All the other layers in a chart use the base layer's *x*-axis.

You cannot call this method on a pie chart layer. Pie chart layers do not have axes.

Syntax Function HasCategoryScaleXAxis( ) As Boolean

Returns True if the chart layer's *x*-axis is a category scale axis.
False if the chart layer's *x*-axis is not a category scale axis.

See also AcChartLayer::HasValueScaleXAxis method

AcChartLayer::HasValueScaleXAxis method

Determines whether a chart layer's *x*-axis is a value scale axis. You can only call this method on a chart's base layer. Only the base layer of a chart has an *x*-axis. All the other layers in a chart use the base layer's *x*-axis.

You cannot call this method on a pie chart layer. Pie chart layers do not have axes.

Syntax Function HasValueScaleXAxis( ) As Boolean

Returns True if the chart layer's *x*-axis is a value scale axis.
False if the chart layer's *x*-axis is not a value scale axis.

See also AcChartLayer::HasCategoryScaleXAxis method

AcChartLayer::HasXAxis method

Determines whether a chart layer has an *x*-axis.

Syntax Function HasXAxis( ) As Boolean

Returns True if the chart layer has an *x*-axis.
False if the chart layer does not have an *x*-axis.

See also AcChartLayer::HasYAxis method

AcChartLayer::HasYAxis method

Determines whether a chart layer has a *y*-axis.

Syntax Function HasYAxis( ) As Boolean

Returns True if the chart layer has a *y*-axis.
False if the chart layer does not have a *y*-axis.

See also AcChartLayer::HasXAxis method

AcChartLayer::InsertCategory method

Call the InsertCategory( ) method to insert a new category at a specific position in a chart layer's list of categories. When you insert a category, the original category at the insertion point and all the categories above the insertion point move up one place.

When you add a category to a chart layer that already has a series, corresponding empty points are added automatically to each of the chart layer's series.

You can only call this method on a chart's base layer.

All the layers in a chart share the same *x*-axis. This means that all the layers in a chart must have the same set of categories.

If a chart has an overlay layer, when you call InsertCategory on the chart's base layer the new category is automatically duplicated in the chart's overlay layer.

If a chart has study layers, when you call AddCategory on the chart's base layer the new category is automatically duplicated in all the chart's study layers.

You cannot call this method on a scatter chart layer. Scatter chart layers do not have categories.

You can only call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method

- Code that is creating a chart dynamically, after you have set the chart's status to ChartStatusBuilding

If you are adding categories and series to an empty chart layer, you must add at least one category before you add any series.

If you add categories to a chart layer using InsertCategory( ), the categories appear on the chart in the order in which they occur in the chart layer's list of categories. Categories you add using InsertCategory( ) are not automatically sorted in any way.

The categoryLabelValue need not be a string. Label values are formatted into text when the chart is viewed to support locale-specific formatting. For example, if you set categoryLabelValue to 1.5, when the chart is viewed in US English locale the label text is 1.5 but the text is 1,5 when the chart is viewed in the French locale.

Syntaxes Function InsertCategory(index As Integer, categoryKeyValue As Variant) As AcChartCategory

Function InsertCategory(index As Integer, categoryKeyValue As Variant, categoryLabelValue As Variant) As AcChartCategory

Parameters **index**
The position in the chart layer's list of categories at which the new category will be inserted. The first category is index 1. Must be greater than or equal to one. Must be less than or equal to the current number of categories in the chart layer plus one.

categoryKeyValue
A unique identifying key value for the new category.

categoryLabelValue
A value to be displayed as the label for the new category. If this parameter is omitted, the category label value is the same as the category key value.

Returns A reference to the new category.

Example The following example overrides a chart's CustomizeCategoriesAndSeries( ) method to insert a new category. The new category appears as the first category on the *x*-axis. The points for each series in the new category are populated with the mean value of the other points in that series.

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' Insert a new category.
   Dim newCategory As AcChartCategory
   Set newCategory = baseLayer.InsertCategory( 1, "Mean" )
   ' Loop through all the series.
   Dim numberOfSeries As Integer
   numberOfSeries = baseLayer.GetNumberOfSeries( )
   Dim seriesIndex As Integer
   For seriesIndex = 1 To numberOfSeries
```

```
            Dim series As AcChartSeries
            Set series = baseLayer.GetSeries( seriesIndex )
            ' Get the mean value of the points in the series.
            Dim numberOfPoints As Integer
            numberOfPoints = series.GetNumberOfPoints( )
            Dim point As AcChartPoint
            Dim pointIndex As Integer
            Dim total As Double
            total = 0
            Dim count As Integer
            count = 0
            ' Ignore the first point in each series, because
            ' that point belongs to the new category.
            For pointIndex = 2 To numberOfPoints
               Set point = series.GetPoint( pointIndex )
               ' Ignore missing values.
               If Not point.IsMissing( ) Then
                  total = total + point.GetYValue( )
                  count = count + 1
               End If
            Next pointIndex
            ' Put the mean value into the point for the new category.
            Set point = series.GetPoint( 1 )
            point.SetYValue( total / count )
         Next seriesIndex
      End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartLayer::AddCategory method
AcChartLayer::InsertSeries method
AcChartLayer::RemoveCategory method
Class AcChartCategory

AcChartLayer::InsertSeries method

Call the InsertSeries( ) method to insert a new series at a specific position in a chart layer's list of series. When you insert a series, the original series at the insertion point and all the series above the insertion point move up one place.

You can call InsertSeries( ) on any layer in a chart except a pie chart layer. Pie chart layers have only one series.

All the layers in a chart share the same *x*-axis. This does not mean, though, that all the layers in a chart must have the same set of series.

You can only call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method

- Code that is creating a chart dynamically, after you have set the chart's status to ChartStatusBuilding

If you are adding categories and series to an empty chart layer, you must add at least one category before you add any series.

If you add series to a chart layer using InsertSeries( ), you must also populate those series with points. Points are not created automatically when you call AddSeries( ).

If you add series to a chart layer using InsertSeries( ), the series appear on the chart in the order in which they occur in the chart layer's list of series. Series you add using InsertSeries( ) are sorted automatically in any way.

The seriesLabelValue need not be a string. Label values are formatted into text when the chart is viewed to support locale-specific formatting. For example, if you set seriesLabelValue to 1.5, when the chart is viewed in the US English locale the label text is 1.5 but the text is 1,5 when the chart is viewed in the French locale.

Syntaxes

Function InsertSeries(index As Integer, seriesKeyValue As Variant) As AcChartSeries

Function InsertSeries(index As Integer, seriesKeyValue As Variant, seriesLabelValue As Variant) As AcChartSeries

Parameters

index
The position in the chart layer's list of series at which the new series will be inserted. The first series is index 1. Must be greater than or equal to one. Must be less than or equal to the current number of series in the chart layer plus one.

seriesKeyValue
A unique identifying key value for the new series.

seriesLabelValue
A value to display as the label for the new series.
If this parameter is omitted, the series label value is the same as the series key value.

Returns

A reference to the new series.

Example

The following example overrides a chart's CustomizeCategoriesAndSeries( ) method to insert a new series into the chart's base layer. The new series appear as the first series on the chart's *x*-axis. Each point in the new series is populated with the mean value of the points in the same category for all the other series.

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' Insert a new series.
   Dim newSeries As AcChartSeries
   Set newSeries = baseLayer.InsertSeries( 1, "Mean" )
   ' Loop through all the categories.
   Dim numberOfCategories As Integer
```

```
      numberOfCategories = baseLayer.GetNumberOfCategories( )
      Dim categoryIndex As Integer
      For categoryIndex = 1 To numberOfCategories
         ' Get the mean value of all the points in this category.
         Dim point As AcChartPoint
         Dim total As Double
         total = 0
         Dim count As Integer
         count = 0
         Dim numberOfSeries As Integer
         numberOfSeries = baseLayer.GetNumberOfSeries( )
         Dim seriesIndex As Integer
         ' Ignore the first series, because that is the new series.
         For seriesIndex = 2 To numberOfSeries
            Dim series As AcChartSeries
            Set series = baseLayer.GetSeries( seriesIndex )
            Set point = series.GetPoint( categoryIndex )
            ' Ignore missing values.
            If Not point.IsMissing( ) Then
               total = total + point.GetYValue( )
               count = count + 1
            End If
         Next seriesIndex
         ' Put the mean value into a new point in the new series.
         Set point = newSeries.InsertPoint(categoryIndex, total/count)
      Next categoryIndex
   End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartLayer::AddSeries method
AcChartLayer::InsertCategory method
AcChartLayer::RemoveSeries method
Class AcChartSeries

AcChartLayer::IsBaseLayer method

Determines whether a chart layer is the base layer of its parent chart.

Syntax Function IsBaseLayer( ) As Boolean

Returns True if the chart layer is the base layer of its parent chart.
False if the chart layer is not the base layer of its parent chart.

See also AcChartLayer::IsOverlayLayer method
AcChartLayer::IsStudyLayer method

AcChartLayer::IsOverlayLayer method

Determines whether a chart layer is the overlay layer of its parent chart.

Syntax Function IsOverlayLayer( ) As Boolean

Returns True if the chart layer is the overlay layer of its parent chart.
False if the chart layer is not the overlay layer of its parent chart.

See also AcChartLayer::IsBaseLayer method
AcChartLayer::IsStudyLayer method

AcChartLayer::IsStacked method

Determines whether the series in a chart layer are stacked. A typical example of a stacked series chart layer is a stacked bar chart layer. In a stacked bar chart layer, the points for all the series in the chart layer are stacked into a single bar in each category, with the total height of a bar showing the sum of the values for all the series in a category.

You cannot call this method on a pie chart layer.

Syntax Function IsStacked( ) As Boolean

Returns True if the series in a chart layer are stacked.
False if the series in a chart layer are not stacked.

See also AcChartLayer::ChartTypeIsStackable method
AcChartLayer::GetSeriesPlacement method

AcChartLayer::IsStudyLayer method

Determines whether a chart layer is a study layer of its parent chart.

Syntax Function IsStudyLayer( ) As Boolean

Returns True if the chart layer is a study layer of its parent chart.
False if the chart layer is not a study layer of its parent chart.

See also AcChartLayer::IsBaseLayer method
AcChartLayer::IsOverlayLayer method

AcChartLayer::PieExplosionTestValueIsPercentage method

Determines whether the pie explosion test value in a pie chart layer is treated as a percentage of the total pie.

You can only call this method on a pie chart layer.

Syntax Function PieExplosionTestValueIsPercentage( ) As Boolean

Returns True if the pie explosion test value in the pie chart layer is treated as a percentage of the total pie.
False if the pie explosion test value in the pie chart layer is treated as a value.

See also AcChartLayer::GetPieExplosion method
AcChartLayer::GetPieExplosionAmount method
AcChartLayer::GetPieExplosionTestOperator method
AcChartLayer::GetPieExplosionTestValue method
AcChartLayer::SetPieExplosionTestValuesIs Percentage method

AcChartLayer::PlotBarsAsLines method

Determines whether points in a bar chart layer are plotted as lines instead of bars.

You can call this method only on a two-dimensional bar chart layer.

The value that this method returns might not apply to all the series in a chart layer. To retrieve the setting for an individual series, call the corresponding series style's PlotBarsAsLines( ) method.

Syntax Function PlotBarsAsLines( ) As Boolean

Returns True if points in the bar chart layer are plotted as lines instead of bars.
False if points in the bar chart layer are plotted as bars.

See also AcChartLayer::SetPlotBarsAsLines method

AcChartLayer::PlotLinesBetweenPoints method

Determines whether the default setting for series in a chart layer is that lines will be drawn between the points within each series.

You can only call this method on layers with the following chart types:

- Stacked bar
- Line
- Scatter

The value that this method returns might not apply to all the series in a chart layer. To retrieve the setting for an individual series, call the corresponding series style's PlotLinesBetweenPoints( ) method.

Syntax Function PlotLinesBetweenPoints( ) As Boolean

Returns True if the default setting for series in the chart layer is that lines will be drawn between the points within each series.
False if the default setting for series in the chart layer is that lines will not be drawn between the points within each series.

See also AcChartLayer::SetPlotLinesBetweenPoints method

AcChartLayer::PlotMarkersAtPoints method

Determines whether the default setting for series within a chart layer is to draw markers at points.

You can only call this method on layers with the following chart types:

- Line
- Scatter
- Stock

The value that this method returns might not apply to all the points in a chart layer. To retrieve the default setting for points within an individual series, call the corresponding series style's PlotMarkersAtPoints( ) method. To retrieve the marker shape for an individual point, call the corresponding point style's GetMarkerShape( ) method.

Syntax Function PlotMarkersAtPoints( ) As Boolean

Returns True if the default setting for series within the chart layer is that markers will be drawn at points.
False if the default setting for series within the chart layer is that markers will not be drawn at points.

See also AcChartLayer::SetPlotMarkersAtPoints method
AcChartPointStyle::GetMarkerShape method
AcChartSeriesStyle::PlotMarkersAtPoints method

AcChartLayer::PlotUpDownBars method

Determines whether up and down bars will be drawn between points within each category in a chart layer.

You can only call this method on chart layers with the following chart types:

- Line
- Stock

Syntax Function PlotUpDownBars( ) As Boolean

Returns True to draw up and down bars between points within each category in the chart layer.
False if up and down bars are not drawn between points within each category in the chart layer.

See also AcChartLayer::SetPlotUpDownBars method

AcChartLayer::RemoveCategory method

Call the RemoveCategory( ) method to remove a category from a chart layer. When you remove a category from a chart layer that already has series, the corresponding points are removed automatically from all of the chart layer's series.

You can call this method only on a chart's base layer.

All the layers in a chart share the same *x*-axis. This means that all the layers in a chart must have the same set of categories.

If a chart has an overlay layer, when you call RemoveCategory( ) on the chart's base layer the corresponding category is automatically removed from the chart's overlay layer.

If a chart has study layers, when you call RemoveCategory( ) on the chart's base layer the corresponding category is automatically removed from all the chart's study layers.

You cannot call this method on a scatter chart layer. Scatter chart layers do not have categories.

You can only call this method from a chart's CustomizeCategoriesAndSeries( ) method.

Syntax Sub RemoveCategory(index As Integer)

Parameter **index**
The position in the chart layer's list of categories from to remove the category. The first category is index 1.

Example The following example overrides a chart's CustomizeCategoriesAndSeries( ) method to remove any category where the sum of the values of the points in that category is less than 10:

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' Loop through all the categories.
   Dim numberOfCategories As Integer
   numberOfCategories = baseLayer.GetNumberOfCategories( )
   Dim categoryIndex As Integer
   ' Use reverse order so that deleting categories
   ' does not invalidate the current index.
   For categoryIndex = numberOfCategories To 1 Step -1
      ' Add up all the values in the category.
      Dim total As Double
      total = 0
      Dim numberOfSeries As Integer
      numberOfSeries = baseLayer.GetNumberOfSeries( )
      Dim seriesIndex As Integer
      For seriesIndex = 1 To numberOfSeries
         Dim series As AcChartSeries
         Set series = baseLayer.GetSeries( seriesIndex )
         Dim point As AcChartPoint
         Set point = series.GetPoint( categoryIndex )
         ' Ignore missing values.
         If Not point.IsMissing( ) Then
            total = total + point.GetYValue( )
         End If
      Next seriesIndex
      ' Remove categories whose values total less than 10.
      If (total < 10) Then
         baseLayer.RemoveCategory( categoryIndex )
      End If
   Next categoryIndex
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartLayer::AddCategory method
AcChartLayer::InsertCategory method
AcChartLayer::InsertSeries method
Class AcChartCategory

AcChartLayer::RemoveSeries method

Call the RemoveSeries( ) method to remove a series from a chart layer. When you remove a series, all the points in the series are automatically deleted.

When you remove a series, all the series above that one move down one place.

You can call RemoveSeries( ) on any layer in a chart.

All the layers in a chart share the same *x*-axis. This does not mean that all the layers in a chart must have the same set of series, though.

You cannot call this method on a pie chart layer. Pie chart layers only have one series.

You can only call this method from a chart's CustomizeCategoriesAndSeries( ) method.

Syntax Function RemoveSeries(index As Integer)

Parameter **index**
The position in the chart layer's list of series from which to remove the series. The first series is index 1.

Example The following example overrides a chart's CustomizeCategoriesAndSeries( ) method to remove any series where the sum of the values of the points in that series is less than 10:

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim numberOfSeries As Integer
   numberOfSeries = baseLayer.GetNumberOfSeries( )
   Dim seriesIndex As Integer
   ' Use reverse order so that deleting series
   ' does not invalidate the current index.
   For seriesIndex = numberOfSeries To 1 Step -1
      Dim series As AcChartSeries
      Set series = baseLayer.GetSeries( seriesIndex )
      ' Add up all the values in the series.
      Dim total As Double
      total = 0
      Dim numberOfPoints As Integer
      numberOfPoints = series.GetNumberOfPoints( )
      Dim pointIndex As Integer
      For pointIndex = 1 To numberOfPoints
         Dim point As AcChartPoint
         Set point = series.GetPoint( pointIndex )
         ' Ignore missing values.
         If Not point.IsMissing( ) Then
            total = total + point.GetYValue( )
         End If
      Next pointIndex
      ' Remove series whose values total less than 10.
      If (total < 10) Then
         baseLayer.RemoveSeries( seriesIndex )
      End If
   Next seriesIndex
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartLayer::AddSeries method
AcChartLayer::InsertSeries method
AcChartLayer::RemoveCategory method
Class AcChartSeries

AcChartLayer::SetBarShape method

Call the SetBarShape( ) method to set the shape of bars in a three-dimensional bar chart layer. You can call this method only on a three-dimensional bar chart layer.

The recommended methods from which to call SetBarShape( ) are:

- A chart's CustomizeLayers( ) method
- A chart's AdjustChart( ) method

Syntax Sub SetBarShape(barShape As AcChartBarShape)

Parameter **barShape**
The bar shape.

Example The following example overrides a chart's CustomizeLayers( ) method to set the shape of bars in the chart's base three-dimensional bar chart layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmHexagonalBars Then
      baseLayer.SetBarShape( ChartBarShapeHexagonal )
   End If
End Sub
```

See also AcChartBarShape
AcChart::AdjustChart method
AcChart::CustomizeLayers method
AcChartLayer::GetBarShape method

AcChartLayer::SetBubbleSize method

Sets the size of the largest bubble in a bubble chart as a percentage of the length of the shorter of the chart layer's two axes.

You can call this method only on a bubble chart layer.

The recommended methods from which to call SetBubbleSize( ) are:

- A chart's CustomizeLayers( ) method
- A chart's AdjustChart( ) method

Syntax Sub SetBubbleSize(bubbleSize As Double)

Parameter **bubbleSize**
The size of the largest bubble in the bubble chart, as a percentage of the length of the shorter of the chart layer's two axes. Must be in the range 0 through 0.75.

Example The following example overrides a chart's CustomizeLayers( ) method to set the size of bubbles in the chart's base bubble chart layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmSmallBubbles Then
      baseLayer.SetBubbleSize( 0.15 )
   End If
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeLayers method
AcChartLayer::GetBubbleSize method

AcChartLayer::SetCategoryGapRatio method

Call the SetCategoryGapRatio( ) method to set the size of the gap between categories in a bar chart layer, relative to the width of a single bar. The size of the gap is defined relative to the width of a single bar. If the size of the gap is 1, it is the same width as a single bar. If the size of the gap is 2, it is twice the width of a single bar. If the size of the gap is 0.5, it is half the width of a single bar.

You can call this method only on a bar chart layer.

The recommended methods from which to call SetCategoryGapRatio( ) are:

- A chart's CustomizeLayers( ) method
- A chart's AdjustChart( ) method

Syntax Sub SetCategoryGapRatio(categoryGapRatio As Double)

Parameter **categoryGapRatio**
The size of the gap between categories, relative to the width of a single bar. Must be in the range 0 through 5.

Example In the following example, all a chart's layers are bar chart layers. The example overrides the chart's AdjustChart( ) method to adjust the gaps between categories in all its layers so that the total width of the bars in each category is the same in each layer.

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim numberOfLayers As Integer
   numberOfLayers = GetNumberOfLayers( )
   Dim layerIndex As Integer
   For layerIndex = 1 To numberOfLayers
      Dim layer As AcChartLayer
      Set layer = GetLayer( layerIndex )
      ' Adjust the gap between categories so that
      ' all layers' bars take up the same space.
      Dim gapRatio As Integer
      gapRatio = layer.GetNumberOfSeries( )
      ' The maximum permitted gap ratio is 5.
      If (gapRatio > 5) Then
         gapRatio = 5
      End If
      layer.SetCategoryGapRatio( gapRatio )
   Next layerIndex
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeLayers method
AcChartLayer::GetCategoryGapRatio method
AcChartLayer::SetSeriesOverlapRatio method

AcChartLayer::SetCategoryLabelFormat method

Call the SetCategoryLabelFormat( ) method to set the format pattern used to format category labels in a chart layer. Category labels appear on a chart layer's *x*-axis.

You can call this method only on a chart's base layer. You cannot call this method on a scatter chart layer. Scatter chart layers do not have categories.

The format pattern is ignored for string label values.

The recommended method from which to call SetCategoryLabelFormat( ) is a chart's CustomizeLayers( ) method. You can also call SetCategoryLabelFormat( ) from the following methods:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method

Category labels are used as category scale *x*-axis labels. Setting a format pattern with this method has exactly the same effect as setting a format pattern with the SetCategoryLabelFormat( ) method of a category scale *x*-axis.

Syntax Sub Format(categoryLabelFormat As String)

Parameter **categoryLabelFormat**
The format pattern.

Examples The following example overrides a chart's CustomizeLayers( ) method to use a short or long date format for labels on the *x*-axis of the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer )
   If parmUseShortDateFormat Then
      baseLayer.SetCategoryLabelFormat( "Short Date" )
   Else
      baseLayer.SetCategoryLabelFormat( "Long Date" )
   End If
End Sub
```

For another example of how to use this method, see the dynamic chart example for the AcChart class.

See also AcChart::AdjustChart method
AcChart::CustomizeLayers method
AcChart::Localize method
AcChartAxis::SetLabelFormat method
AcChartLayer::SetCategoryLabelFormat method
AcChartLayer::GetCategoryLabelFormat method
AcChartLayer::SetPointLabelFormat method
AcChartLayer::SetSeriesLabelFormat method

AcChartLayer::SetChartType method

Call the SetChartType( ) method to set the chart type of a chart layer.

You can only call this method from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

If you call this method on a chart layer, you must do so before calling any other methods on that chart layer.

Table 7-8 lists the default chart layer settings for each chart type.

Table 7-8 Default settings for chart types

Chart type	Default settings
Area	Missing points: interpolate Series placement: stacked
Bar	Lines between points: off Missing points: do not plot Series placement: side by side
Line	High-low lines: off Lines between points: on Markers at points: off Marker size: 5 pt Missing points: do not plot Series placement: side by side Up/down bars: off
Pie	Not applicable
Scatter	Lines between points: off Markers at points: on Marker size: 5 pt
Step	Missing points: interpolate Series placement: stacked
Stock	High-low lines: on Markers at points: off Marker size: 6 pt Missing points: do not plot Up/down bars: on

SetChartType( ) always resets the plot area background color to LightGray for two-dimensional chart types other than pie.

SetChartType( ) always resets the plot area border pen to DrawingLinePenSolid for two-dimensional chart types other than pie.

SetChartType( ) might change the chart layer's parent chart from two-dimensional presentation to three-dimensional presentation or the reverse automatically to match the new chart type and series placement. If the chart layer's parent chart has an overlay layer or study layers and SetChartType( ) attempts to change the chart to three-dimensional presentation, SetChartType( ) throws a run-time error.

Syntax

Sub SetChartType(chartType As AcChartType)

Sub SetChartType(chartType As AcChartType, seriesPlacement As AcChartSeriesPlacement)

Parameters

chartType
The chart type.

seriesPlacement
The relative placement of points for multiple series within a category. If you do not specify this parameter, the series placement is selected automatically, based on the chart type. This parameter is ignored for the following chart types:

- Pie
- Scatter
- Stock

Do not set this parameter to ChartSeriesPlacementOnZAxis if the chart type is step.

Examples

The following example overrides a chart's CustomizeLayers( ) method to make the chart's base layer a pie chart layer or a bar chart layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers( ) As AcChartLayer )
   If parmPieChart Then
      ' Series placement does not apply to a pie.
      baseLayer.SetChartType( ChartTypePie )
   Else
      ' Use default series placement.
      baseLayer.SetChartType( ChartTypeBar )
   End If
End Sub
```

The following example overrides a chart's CustomizeLayers( ) method to make the chart's base layer a line chart layer or a bar chart layer, depending on the value of a Boolean parameter. A second Boolean parameter controls whether the series are stacked.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim seriesPlacement As AcChartSeriesPlacement
   If parmStackSeries Then
      seriesPlacement = ChartSeriesPlacementStacked
   Else
      seriesPlacement = ChartSeriesPlacementSideBySide
   End If
   If parmLineChart Then
      baseLayer.SetChartType( ChartTypeLine, seriesPlacement )
   Else
      baseLayer.SetChartType( ChartTypeBar, seriesPlacement )
   End If
End Sub
```

For another example of how to use this method, see the dynamic chart example for the AcChart class.

See also AcChart::CustomizeLayers method
AcChartLayer::GetChartType method
AcChartLayer::SetSeriesPlacement method
AcChartSeriesPlacement
AcChartType
Class AcChart

AcChartLayer::SetDownBarBorderStyle method

Call the SetDownBarBorderStyle( ) method to set the style of the borders around down bars in a chart layer.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

To turn off borders around down bars, set the border style's Pen member to DrawingLinePenNone.

Syntax Sub SetDownBarBorderStyle(downBarBorderStyle As AcDrawingBorderStyle)

Parameter **downBarBorderStyle**
The style for borders around down bars in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the border around down bars in the chart's base layer, depending on the value of a Boolean parameter. GetDownBarBorderStyle( ) retrieves the default settings so that only the border style's Color member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmRedOutlinedDownBars Then
      Dim borderStyle As AcDrawingBorderStyle
      borderStyle = baseLayer.GetDownBarBorderStyle( )
      borderStyle.Color = Red
      baseLayer.SetDownBarBorderStyle( borderStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetDownBarBorderStyle method
AcChartLayer::SetDownBarFillStyle method
AcChartLayer::SetPlotUpDownBars method
AcChartLayer::SetUpBarBorderStyle method
AcDrawingBorderStyle

AcChartLayer::SetDownBarFillStyle method

Call the SetDownBarFillStyle( ) method to set the fill style for down bars in a chart layer.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetDownBarFillStyle(downBarFillStyle As AcDrawingFillStyle)

Parameter **downBarFillStyle**
The fill style for down bars in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of down bars in the chart's base layer, depending on the value of a Boolean parameter. GetDownBarFillStyle( ) retrieves the default settings so that only the fill style's Color1 member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmRedFilledDownBars Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetDownBarFillStyle( )
      fillStyle.Color = Red
      baseLayer.SetDownBarFillStyle( fillStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetDownBarBorderStyle method
AcChartLayer::GetDownBarFillStyle method
AcChartLayer::SetPlotUpDownBars method
AcChartLayer::SetUpBarFillStyle method
AcDrawingFillStyle

AcChartLayer::SetDropLineStyle method

Call the SetDropLineStyle( ) method to set the line style used to draw drop lines in a chart layer.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

To turn off drop lines, set the line style's Pen member to DrawingLinePenNone.

Syntax Sub SetDropLineStyle(dropLineStyle As AcDrawingLineStyle)

Parameter **dropLineStyle**
The line style used to draw drop lines in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the pattern used to draw drop lines in the chart's base layer, depending on the value of a Boolean parameter. GetDropLineStyle( ) retrieves the default settings so that only the line style's Pen member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmDottedDropLines Then
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = baseLayer.GetDropLineStyle( )
      lineStyle.Pen = DrawingLinePenDot
      baseLayer.SetDropLineStyle( lineStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetDropLineStyle method
AcDrawingLineStyle

AcChartLayer::SetHighLowLineStyle method

Call the SetHighLowLineStyle( ) method to set the line style used to draw high-low lines in a chart layer.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetHighLowLineStyle(highLowLineStyle As AcDrawingLineStyle)

Parameter **highLowLineStyle**
The line style used to draw high-low lines in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the thickness of high-low lines in the chart's base layer, depending on the value of a Boolean parameter. GetHighLowLineStyle( ) retrieves the default settings so that only the line style's Width member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmThickHighLowLines Then
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = baseLayer.GetHighLowLineStyle( )
      lineStyle.Width = 2 * OnePoint
      baseLayer.SetHighLowLineStyle( lineStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetHighLowLineStyle method
AcChartLayer::SetPlotHighLowLines method
AcDrawingLineStyle

AcChartLayer::SetLineWidth method

Call the SetLineWidth( ) method to set the default width of the lines joining the points within each series in a chart layer.

You can call this method only on chart layers with the following chart types:

- Stacked bar
- Line
- Scatter

To set the width of the line for an individual series, call the corresponding series style's SetLineStyle( ) method.

To enable or disable lines between points, call a chart layer's SetPlotLinesBetweenPoints( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetLineWidth(lineWidth As AcTwips)

Parameter **lineWidth**
The default line width of the lines joining the points within each series in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the thickness of lines in the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmThickLines Then
      baseLayer.SetLineWidth( 2 * OnePoint )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetLineWidth method
AcChartLayer::SetPlotLinesBetweenPoints method
AcChartSeriesStyle::SetLineStyle method
Class AcChartSeriesStyle
AcTwips

AcChartLayer::SetMarkerSize method

Call the SetMarkerSize( ) method to set the default size for markers within a chart layer.

You can call this method only on chart layers with the following chart types:

- Stacked bar
- Line
- Scatter

To set the default size for markers in an individual series, call the corresponding series style's SetMarkerSize( ) method. To set the size of the marker for an individual point, call the corresponding point style's SetMarkerSize( ) method.

To enable or disable markers, call a chart layer's SetPlotMarkersAtPoints( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method.
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method.

Syntax Sub SetMarkerSize(markerSize As AcTwips)

Parameter **markerSize**
The default size for markers within the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to draw large markers in the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmPlotBigMarkers Then
      baseLayer.SetPlotMarkersAtPoints( True )
      baseLayer.SetMarkerSize( 8 * OnePoint )
   Else
      baseLayer.SetPlotMarkersAtPoints( False )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetMarkerSize method
AcChartPointStyle::SetMarkerSize method
AcChartLayer::SetPlotMarkersAtPoints method
Class AcChartPointStyle
Class AcChartSeriesStyle
AcTwips

AcChartLayer::SetMaximumNumberOfPoints method

Call the SetMaximumNumberOfPoints( ) method to set the maximum number of points permitted in a chart layer. This value is a safety limit used to prevent charts from growing excessively large due to programming errors or unexpected data. If you exceed this limit, a run-time error occurs.

You can call this method only from a chart's CustomizeLayers( ) method.

Syntax Sub SetMaximumNumberOfPoints(maximumNumberOfPoints As Integer)

Parameter **maximumNumberOfPoints**
The maximum number of points permitted in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to set the maximum number of points permitted in the chart's base layer to 100:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   baseLayer.SetMaximumNumberOfPoints( 100 )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetMaximumNumberOfPoints method
AcChartLayer::SetMaximumNumberOfPointsPer Series method
AcChartLayer::SetMaximumNumberOfSeries method

AcChartLayer::SetMaximumNumberOfPointsPer Series method

Call the SetMaximumNumberOfPointsPerSeries( ) method to set the maximum number of points permitted in a single series in a chart layer. This value is a safety limit used to prevent charts from growing excessively large due to programming errors or unexpected data. If you exceed this limit, a run-time error occurs.

You can call this method only from a chart's CustomizeLayers( ) method.

Syntax Sub SetMaximumNumberOfPointsPerSeries(
   maximumNumberOfPointsPerSeries As Integer)

Parameter **maximumNumberOfPointsPerSeries**
The maximum number of points permitted in a single series in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to set the maximum number of points permitted in a single series in the chart's first study layer to 100:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   studyLayers(1).SetMaximumNumberOfPointsPerSeries( 100 )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetMaximumNumberOfPointsPer Series method
AcChartLayer::SetMaximumNumberOfPoints method
AcChartLayer::SetMaximumNumberOfSeries method

AcChartLayer::SetMaximumNumberOfSeries method

Call the SetMaximumNumberOfSeries( ) method to set the maximum number of series permitted in a chart layer. This value is a safety limit used to prevent charts from growing excessively large due to programming errors or unexpected data. If you exceed this limit, a run-time error occurs.

You can call this method only from a chart's CustomizeLayers( ) method.

Syntax Sub SetMaximumNumberOfSeries(maximumNumberOfSeries As Integer)

Parameter **maximumNumberOfSeries**
The maximum number of series permitted in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to set the maximum number of series permitted in the chart's overlay layer to 10:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   overlayLayer.SetMaximumNumberOfSeries( 10 )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetMaximumNumberOfSeries method
AcChartLayer::SetMaximumNumberOfPoints method
AcChartLayer::SetMaximumNumberOfPointsPer Series method

AcChartLayer::SetMissingPoints method

Call the SetMissingPoints( ) method to specify how missing points are plotted in a chart layer.

You can call this method only on chart layers with the following chart types:

- Area
- Bar
- Line
- Scatter
- Step

You can call this method only from:

- A chart's CustomizeLayers( ) method

- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetMissingPoints(missingPoints As AcChartMissingPoints)

Parameters **missingPoints**
The way that missing points are plotted in the chart layer.

Table 7-9 lists the valid values for missingPoints for each chart type.

Table 7-9 Valid values for chart types

Chart Type	Valid Values
Area	ChartMissingPointsPlotAsZero ChartMissingPointsInterpolate
Bar	ChartMissingPointsDoNotPlot ChartMissingPointsPlotAsZero
Line	ChartMissingPointsDoNotPlot ChartMissingPointsPlotAsZero ChartMissingPointsInterpolate
Scatter	ChartMissingPointsDoNotPlot ChartMissingPointsPlotAsZero ChartMissingPointsInterpolate
Step	ChartMissingPointsPlotAsZero ChartMissingPointsInterpolate

Example The following example overrides a chart's CustomizeLayers( ) method to select the way that missing points are plotted in the chart's bar base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmSkipMissingPoints Then
      baseLayer.SetMissingPoints( ChartMissingPointsDoNotPlot )
   Else
      baseLayer.SetMissingPoints( ChartMissingPointsPlotAsZero )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetMissingPoints method
AcChartMissingPoints

AcChartLayer::SetPieExplosion method

Call the SetPieExplosion( ) method to specify which pie slices are exploded in a pie chart layer. You can specify all pie slices, only specific pie slices, or none.

You can call this method only on a pie chart layer.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPieExplosion(pieExplosion As AcChartPieExplode)

Parameter **pieExplosion**
The pie slices to explode in the pie chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to explode pie slices in the chart's base layer. Slices are exploded if their values are greater than a certain percentage of the total pie value. The percentage is specified as a parameter value.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If (parmTestValue > 0) Then
      baseLayer.SetPieExplosion( ChartPieExplodeSpecificSlices )
      baseLayer.SetPieExplosionTestOperator(
         ChartComparisonOperatorGT )
      baseLayer.SetPieExplosionTestValue( parmTestValue )
      baseLayer.SetPieExplosionTestValueIsPercentage( True )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetPieExplosion method
AcChartLayer::SetPieExplosionAmount method
AcChartLayer::SetPieExplosionTestOperator method
AcChartLayer::SetPieExplosionTestValue method
AcChartLayer::SetPieExplosionTestValuesIs Percentage method
AcChartPieExplode

AcChartLayer::SetPieExplosionAmount method

Call the SetPieExplosionAmount( ) method to set the amount that pie slices are exploded in a pie chart layer. The amount is relative to the radius of the pie. If the amount is 0.25, exploded slices are moved outward from the center of the pie by one quarter of the radius of the pie.

You can call this method only on a pie chart layer.

You can call this method only from:

- A chart's CustomizeLayers( ) method

- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPieExplosionAmount(pieExplosionAmount As Double)

Parameter **pieExplosionAmount**
The amount that pie slices are exploded in the pie chart layer. Must be in the range 0 through 0.4.

Example The following example overrides a chart's CustomizeLayers( ) method to increase the amount that pie slices are exploded in the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmBigExplosion Then
      baseLayer.SetPieExplosionAmount( 0.4 )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetPieExplosionAmount method
AcChartLayer::SetPieExplosion method
AcChartLayer::SetPieExplosionTestOperator method
AcChartLayer::SetPieExplosionTestValue method
AcChartLayer::SetPieExplosionTestValuesIs Percentage method

AcChartLayer::SetPieExplosionTestOperator method

Call the SetPieExplosionTestOperator( ) method to set the operator used to test whether a pie slice is exploded in a pie chart layer. You can call this method only on a pie chart layer.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPieExplosionTestOperator(pieExplosionTestOperator As AcChartComparisonOperator)

Parameter **pieExplosionTestOperator**
The operator used to test whether a pie slice is exploded in a pie chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to explode pie slices in the chart's base layer. Slices are exploded if their values are less than or equal to a parameter value.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
```

```
   If (parmTestValue > 0) Then
      baseLayer.SetPieExplosion( ChartPieExplodeSpecificSlices )
      baseLayer.SetPieExplosionTestOperator(
         ChartComparisonOperatorLE )
      baseLayer.SetPieExplosionTestValue( parmTestValue )
      baseLayer.SetPieExplosionTestValueIsPercentage( False )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetPieExplosionTestOperator method
AcChartLayer::SetPieExplosion method
AcChartLayer::SetPieExplosionAmount method
AcChartLayer::SetPieExplosionTestValue method
AcChartLayer::SetPieExplosionTestValuesIs Percentage method
AcChartPieExplode

AcChartLayer::SetPieExplosionTestValue method

Call the SetPieExplosionTestValue( ) method to set the value used to test whether a pie slice is exploded in a pie chart layer.

You can call this method only on a pie chart layer.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPieExplosionTestValue(pieExplosionTestValue As Variant)

Parameter **pieExplosionTestValue**
The value used to test whether a pie slice is exploded in a pie chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to explode pie slices in the chart's base layer. Slices are exploded if their values are less than a certain percentage of the total pie value. The percentage is specified as a parameter value.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If (parmTestValue > 0) Then
      baseLayer.SetPieExplosion( ChartPieExplodeSpecificSlices )
      baseLayer.SetPieExplosionTestOperator(
         ChartComparisonOperatorLT )
      baseLayer.SetPieExplosionTestValue( parmTestValue )
      baseLayer.SetPieExplosionTestValueIsPercentage( True )
   End If
End Sub
```

See also

AcChart::CustomizeLayers method
AcChartLayer::GetPieExplosionTestValue method
AcChartLayer::SetPieExplosion method
AcChartLayer::SetPieExplosionAmount method
AcChartLayer::SetPieExplosionTestOperator method
AcChartLayer::SetPieExplosionTestValuesIs Percentage method
AcChartPieExplode

AcChartLayer::SetPieExplosionTestValuesIs Percentage method

Call the SetPieExplosionTestValueIsPercentage( ) method to specify whether the pie explosion test value in a pie chart layer is treated as a percentage of the total pie.

You can call this method only on a pie chart layer.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax

```
Sub SetPieExplosionTestValueIsPercentage(
   pieExplosionTestValueIsPercentage As Boolean )
```

Parameter

pieExplosionTestValueIsPercentage
True causes the pie explosion test value in the pie chart layer to be treated as a percentage of the total pie. False causes the pie explosion test value in the pie chart layer to be treated as a value.

Example

The following example overrides a chart's CustomizeLayers( ) method to explode pie slices in the chart's base layer. Slices are exploded if their values are greater than or equal to a parameter value.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If (parmTestValue > 0) Then
      baseLayer.SetPieExplosion( ChartPieExplodeSpecificSlices )
      baseLayer.SetPieExplosionTestOperator(
         ChartComparisonOperatorGE )
      baseLayer.SetPieExplosionTestValue( parmTestValue )
      baseLayer.SetPieExplosionTestValueIsPercentage( False )
   End If
End Sub
```

See also

AcChart::CustomizeLayers method
AcChartLayer::PieExplosionTestValueIsPercentage method
AcChartLayer::SetPieExplosion method

AcChartLayer::SetPieExplosionAmount method
AcChartLayer::SetPieExplosionTestOperator method
AcChartLayer::SetPieExplosionTestValue method
AcChartPieExplode

AcChartLayer::SetPlotAreaBackgroundColor method

Call the SetPlotAreaBackground( ) method to set the background color of a chart layer's plot area. This sets a chart layer's plot area fill style to a single solid color. This method sets a chart layer's plot area fill style members as follows:

- The Color1 member is set to the specified background color.
- The Color2 member is not affected.
- The Pattern member is set to DrawingFillPatternSolid.

You can call this method only on a chart's base layer.

All the layers in a chart are drawn with the same plot area fill style as the base layer. You cannot change the plot area fill style on individual layers.

You cannot call this method on a three-dimensional chart layer. A three-dimensional chart layer has separate fill styles for its walls and its floor instead of a plot area fill style.

You cannot call this method on a pie chart layer. A pie chart layer does not have a plot area fill style.

The recommended method from which to call SetPlotAreaBackgroundColor( ) is a chart's CustomizeLayers( ) method.

Syntax Sub SetPlotAreaBackgroundColor(plotAreaBackgroundColor As AcColor)

Parameter **plotAreaBackgroundColor**
The background color for the chart layer's plot area.

Example The following example overrides a chart's CustomizeLayers( ) method to set the background color of the chart's base layer to the value of a parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   baseLayer.SetPlotAreaBackgroundColor( parmPlotAreaColor )
End Sub
```

See also AcChart::CustomizeLayers method
AcChart::SetBackgroundColor method
AcChart::SetFillStyle method
AcChartLayer::SetPlotAreaFillStyle method
AcChartLayer::SetThreeDFloorFillStyle method
AcChartLayer::SetThreeDWallFillStyle method
AcDrawingFillStyle

AcChartLayer::SetPlotAreaBorderStyle method

Call the SetPlotAreaBorderStyle( ) method to set the style of the border around a chart layer's plot area. To turn off the border around a chart layer's plot area, set the border style's Pen member to DrawingLinePenNone.

You can call this method only on a chart's base layer.

All the layers in a chart are drawn with the same plot area border style as the base layer. You cannot change the plot area border style on individual layers.

You cannot call this method on a three-dimensional chart layer. A three-dimensional chart layer does not have a plot area border.

You cannot call this method on a pie chart layer. A pie chart layer does not have a plot area border.

The recommended method from which to call SetPlotAreaBorderStyle( ) is a chart's CustomizeLayers( ) method.

Syntax Sub SetPlotAreaBorderStyle(PlotAreaBorderStyle As AcDrawingBorderStyle)

Parameter **plotAreaBorderStyle**
The border style for the chart layer's plot area.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the border around the chart's base layer's plot area, based on the value of a parameter. GetPlotAreaBorderStyle( ) retrieves the default settings so that only the border style's Color member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim borderStyle As AcDrawingBorderStyle
   borderStyle = baseLayer.GetPlotAreaBorderStyle( )
   borderStyle.Color = parmPlotAreaBorderColor
   baseLayer.SetPlotAreaBorderStyle( borderStyle )
End Sub
```

See also AcChart::CustomizeLayers method
AcChart::SetBorderStyle method
AcChartLayer::GetPlotAreaBorderStyle method
AcDrawingBorderStyle

AcChartLayer::SetPlotAreaFillStyle method

Call the SetPlotAreaFillStyle( ) method to set the background fill style for a chart layer's plot area. You can call this method only on a chart's base layer.

All the layers in a chart are drawn with the same plot area fill style as the base layer. You cannot change the plot area fill style on individual layers.

You cannot call this method on a three-dimensional chart layer. A three-dimensional chart layer has separate fill styles for its walls and its floor instead of a plot area fill style.

You cannot call this method on a pie chart layer. A pie chart layer does not have a plot area fill style.

The recommended method from which to call SetPlotAreaFillStyle( ) is a chart's CustomizeLayers( ) method.

Syntax Sub SetPlotAreaFillStyle(plotAreaFillStyle As AcDrawingFillStyle)

Parameter **plotAreaFillStyle**
The background fill style for the chart layer's plot area.

Example The following example overrides a chart's CustomizeLayers( ) method to create a patterned plot area background, depending on the value of a Boolean parameter. GetPlotAreaFillStyle( ) retrieves the default settings so that only the fill style's Pattern member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAddBackgroundPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetPlotAreaFillStyle( )
      fillStyle.Pattern = DrawingFillPattern05Percent
      baseLayer.SetPlotAreaFillStyle( fillStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChart::SetBackgroundColor method
AcChart::SetFillStyle method
AcChartLayer::GetPlotAreaFillStyle method
AcChartLayer::SetPlotAreaBackgroundColor method
AcChartLayer::SetThreeDFloorFillStyle method
AcChartLayer::SetThreeDWallFillStyle method
AcDrawingFillStyle

AcChartLayer::SetPlotBarsAsLines method

Call the SetPlotBarsAsLines( ) method to specify whether points in a bar chart layer are plotted as lines instead of bars. You can call this method only on a two-dimensional bar chart layer. In some cases, the value that this method sets does not apply to all the series in a chart layer. To set the value for an individual series, call the corresponding series style's SetPlotBarsAsLines( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method

- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPlotBarsAsLines(plotBarsAsLines As Boolean)

Parameter **plotBarsAsLines**
True causes points in the bar chart layer to be plotted as lines instead of bars. False causes points in the bar chart layer to be plotted as bars.

Example The following example overrides a chart's CustomizeLayers( ) method to plot bars as lines in the chart's overlay layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   overlayLayer.SetPlotBarsAsLines( parmOverlayBarsAsLines )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::PlotBarsAsLines method
AcChartSeriesStyle::PlotBarsAsLines method

AcChartLayer::SetPlotHighLowLines method

Call the SetPlotHighLowLines( ) method to specify whether high-low lines are plotted in a chart layer. This method is a simple way to set a chart layer's high-low line style to sensible default values. The high-low line style settings depend on the value of the plotHighLowLines parameter, as shown in Table 7-10.

Table 7-10 Setting high-low line styles

plotHighLowLines	High-low line style

True	Color = Black
	Pen = DrawingLinePenSolid
	Width = 1 pt

False	Color = not changed
	Pen = DrawingLinePenNone
	Width = not changed

You can call this method only on chart layers with the following chart types:

- Line
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPlotHighLowLines(plotHighLowLines As Boolean)

Parameter **plotHighLowLines**
True turns on plotting high-low lines in the chart layer. False turns off plotting high-low lines in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to plot high-low lines in the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   baseLayer.SetPlotHighLowLines( parmPlotHighLowLines )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::SetHighLowLineStyle method
AcDrawingLineStyle

AcChartLayer::SetPlotLinesBetweenPoints method

Call the SetPlotLinesBetweenPoints( ) method to specify whether the default setting for series in a chart layer is that lines are drawn between the points within each series.

You can call this method only on layers with the following chart types:

- Stacked bar
- Line
- Scatter

The value that this method sets might not apply to all the series in a chart layer. To set the value for an individual series, call the corresponding series style's SetPlotLinesBetweenPoints( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPlotLinesBetweenPoints(plotLinesBetweenPoints As Boolean)

Parameter **plotLinesBetweenPoints**
True sets the default to be that lines are drawn between the points within each series in the chart layer. False sets the default to be that lines are drawn between the points within each series in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to plot lines between points in the chart's overlay layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   overlayLayer.SetPlotLinesBetweenPoints(
      parmPlotOverlayLines )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::PlotLinesBetweenPoints method
AcChartSeriesStyle::SetPlotLinesBetweenPoints method

AcChartLayer::SetPlotMarkersAtPoints method

Call the SetPlotMarkersAtPoints( ) method to specify whether the default setting for series within a chart layer is that markers are drawn at points.

You can call this method only on layers with the following chart types:

- Line
- Scatter
- Stock

The value that this method sets might not apply to all the points in a chart layer. To set the default value for points within an individual series, call the corresponding series style's PlotMarkersAtPoints( ) method. To set the marker shape for an individual point, call the corresponding point style's SetMarkerShape( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPlotMarkersAtPoints(plotMarkersAtPoints As Boolean)

Parameter **plotMarkersAtPoints**
True sets the default to be that markers are drawn at points in the chart layer. False sets the default to be that markers are not drawn at points in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to plot markers at points in the chart's first study layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   studyLayers(1).SetPlotMarkersAtPoints( parmPlotStudyMarkers )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::PlotMarkersAtPoints method
AcChartPointStyle::SetMarkerShape method
AcChartSeriesStyle::SetPlotMarkersAtPoints method

AcChartLayer::SetPlotUpDownBars method

Call the SetPlotUpDownBars( ) method to specify whether up and down bars are drawn between points within each category in a chart layer.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPlotUpDownBars(plotUpDownBars As Boolean)

Parameter **plotUpDownBars**
True turns on drawing up and down bars in the chart layer. False turns off drawing up and down bars in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to draw up and down bars in the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   baseLayer.SetPlotUpDownBars( parmPlotUpDownBars )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::SetPlotUpDownBars method

AcChartLayer::SetPointBorderStyle method

Call the SetPointBorderStyle( ) method to set the default style for the borders around points in a chart layer. To turn off borders around points, set the border style's Pen member to DrawingLinePenNone.

You can call this method only on chart layers with the following chart types:

- Area
- Bar
- Pie
- Step

In some cases, the border style that this method sets does not apply to all the points in a chart layer. To set the default border style for points within an individual series, call the corresponding series style's SetBorderStyle( ) method. To set the border style for a particular point, call the corresponding point style's SetBorderStyle( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPointBorderStyle(pointBorderStyle As AcDrawingBorderStyle)

Parameter **pointBorderStyle**
The default style for the borders around points in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the border around points in the chart's base layer, based on the value of a parameter. GetPointBorderStyle( ) retrieves the default settings so that only the border style's Color member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim borderStyle As AcDrawingBorderStyle
   borderStyle = baseLayer.GetPointBorderStyle( )
   borderStyle.Color = parmPointBorderColor
   baseLayer.SetPointBorderStyle( borderStyle )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetPointBorderStyle method
AcChartPointStyle::GetBorderStyle method
AcDrawingBorderStyle

AcChartLayer::SetPointLabelFormat method

Call the SetPointLabelFormat( ) method to set the default format pattern used to format point labels in a chart layer. The format pattern is ignored for string label values.

In some cases, the format pattern that this method sets does not apply to all points in a chart layer. To set the point label format pattern for an individual series, call the corresponding series style's SetPointLabelFormat( ) method. To set the point label format pattern for an individual point, call the point's SetCustomLabelFormat( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method
- A chart's Localize( ) method

Syntax Sub SetPointLabelFormat(pointLabelFormat As String)

Parameter **pointLabelFormat**
The default format pattern used to format point labels in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the number of decimal places shown in point labels in the chart's base layer, based on the value of a parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim pointLabelFormat As String
   If (parmPointLabelDP > 0) Then
      pointLabelFormat = "." & String( parmPointLabelDP, "0" )
   End If
   pointLabelFormat = "#,##0" & pointLabelFormat
   baseLayer.SetPointLabelFormat( pointLabelFormat )
End Sub
```

See also AcChart::CustomizeLayers method
AcChart::Localize method
AcChartLayer::GetPointLabelFormat method
AcChartLayer::SetCategoryLabelFormat method
AcChartLayer::SetSeriesLabelFormat method
AcChartPoint::SetCustomLabelFormat method
AcChartSeriesStyle::SetPointLabelFormat method

AcChartLayer::SetPointLabelLineStyle method

Call the SetPointLableLineStyle( ) method to set the line style used to draw point label lines in a chart layer. To disable point label lines, set the line style's Pen member to DrawingLinePenNone.

You can call this method only on pie chart layers.

The recommended method from which to call SetPointLabelLineStyle( ) is a chart's CustomizeLayers( ) method.

Syntax Sub SetPointLabelLineStyle(pointLabelLineStyle As AcDrawingLineStyle)

Parameter **pointLabelLineStyle**
The line style used to draw point label lines in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the pattern used to draw the point label lines in the chart's base layer, depending on the value of a Boolean parameter. GetPointLabelLineStyle( ) retrieves the default settings so that only the line style's Pen member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmDottedPointLabelLines Then
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = baseLayer.GetPointLabelLineStyle( )
      lineStyle.Pen = DrawingLinePenDot
      baseLayer.SetPointLabelLineStyle( lineStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetPointLabelLineStyle method
AcDrawingLineStyle

AcChartLayer::SetPointLabelPlacement method

Call the SetPointLabelPlacement( ) method to set the default placement of point labels in a chart layer. To turn off point labels, set pointLabelPlacement to ChartPointLabelPlacementNone.

The placement that this method sets might not apply to all the points in a chart layer. To set the default point label placement for the points within an individual series, call the corresponding series style's SetPointLabelPlacement( ) method. To set the point label placement for an individual point, call the corresponding point style's SetPointLabelPlacement( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPointLabelPlacement(pointLabelPlacement As AcChartPointLabelPlacement)

Parameter **pointLabelPlacement**
The default placement of point labels in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the chart type of the chart's base layer, depending on the value of a Boolean parameter. If the base layer is a pie chart, the method calls SetPointLabelPlacement( ) and SetPointLabelSource( ) to display categories as point labels. If the base layer is a bar chart, the method calls SetPointLabelPlacement( ) to turn off point labels.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmPieChart Then
      baseLayer.SetChartType( ChartTypePie )
      baseLayer.SetPointLabelPlacement(
         ChartPointLabelPlacementAuto )
      baseLayer.SetPointLabelSource(
   ChartPointLabelSourceCategory )
   Else
      ' Use default series placement.
      baseLayer.SetChartType( ChartTypeBar )
      ' Disable point labels.
      baseLayer.SetPointLabelPlacement(
         ChartPointLabelPlacementNone )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartPointLabelPlacement
AcChartPointStyle::SetPointLabelPlacement method

AcChartLayer::SetPointLabelSource method

Call the SetPointLabelSource( ) method to set the default source for point label values in a chart layer.

The source that this method specifies might not apply to all the points in a chart layer. To set the point label source for an individual series, call the corresponding series style's SetPointLabelSource( ) method. To set the point label value for an individual point, call the point's SetCustomLabelValue( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetPointLabelSource(pointLabelSource As AcChartPointLabelSource)

Parameter **pointLabelSource**
The default source for point label values in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the chart type of the chart's base layer, depending on the value of a Boolean parameter. If the base layer is a pie chart, the method calls SetPointLabelSource( ) to display categories as point labels. If the base layer is a bar chart, the method calls SetPointLabelSource( ) to display categories as point labels.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmPieChart Then
      baseLayer.SetChartType( ChartTypePie )
      baseLayer.SetPointLabelPlacement(
   +     ChartPointLabelPlacementOutsideEnd )
      baseLayer.SetPointLabelSource(
         ChartPointLabelSourceCategory )
   Else
      ' Use default series placement.
      baseLayer.SetChartType( ChartTypeBar )
      baseLayer.SetPointLabelPlacement(
   +     ChartPointLabelPlacementInsideBase )
      baseLayer.SetPointLabelSource(
         ChartPointLabelSourceSeries )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetPointLabelSource method
AcChartPoint::SetCustomLabelValue method
AcChartPointLabelSource
AcChartSeriesStyle::SetPointLabelSource method

AcChartLayer::SetPointLabelStyle method

Call the SetPointLabelStyle( ) method to set the default style for point labels in a chart layer. The style that this method sets might not apply to all the points in a chart layer. To set the default point label style for the points within an individual series, call the corresponding series style's SetPointLabelStyle( ) method. To set the point label style for an individual point, call the corresponding point style's SetPointLabelStyle( ) method.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub GetPointLabelStyle(pointLabelStyle As AcDrawingTextStyle)

Parameter **pointLabelStyle**
The default style for point labels in the chart layer.

Example The following example overrides a chart’s CustomizeLayers( ) method to make point labels italic in the chart’s base layer, depending on the value of a Boolean parameter. GetPointLabelStyle( ) retrieves the default settings so that only the text style’s Font member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim textStyle As AcDrawingTextStyle
   textStyle = baseLayer.GetPointLabelStyle( )
   textStyle.Font.Italic = parmItalicPointLabels
   baseLayer.SetPointLabelStyle( textStyle )
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetPointLabelStyle method
AcChartPointStyle::GetPointLabelStyle method
AcDrawingTextStyle

AcChartLayer::SetSeriesLabelFormat method

Call the SetSeriesLabelFormat( ) method to set the default format pattern used to format series labels in a chart layer. You cannot call this method on a pie chart layer. Pie chart layers do not have series labels. Legend items for a pie chart layer are category labels.

The format pattern is ignored for string label values.

You can call this method only from:

- A chart’s CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart’s MakeLayers( ) method
- A chart’s Localize( ) method

Syntax Sub Format(seriesLabelFormat As String)

Parameter **seriesLabelFormat**
The format pattern.

Example The following example overrides a chart’s CustomizeLayers( ) method to use a short or long calendar quarter format for series labels from the chart’s base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmUseShortQuarterFormat Then
      baseLayer.SetSeriesLabelFormat( "Short Quarter" )
   Else
      baseLayer.SetSeriesLabelFormat( "Long Quarter" )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChart::Localize method
AcChartLayer::GetSeriesLabelFormat method
AcChartLayer::SetCategoryLabelFormat method
AcChartLayer::SetPointLabelFormat method

AcChartLayer::SetSeriesOverlapRatio method

Call the SetSeriesOverlapRatio( ) method to specify the amount that adjacent series in a bar chart layer overlap, relative to the width of a single bar. The amount of overlap is defined relative to the width of a single bar. If the amount of overlap is 0.5, adjacent bars overlap by half the width of a single bar.

Negative overlaps are permitted. If the amount of overlap is -0.5, there will be a gap half the width of a single bar between adjacent bars.

You can call this method only on a two-dimensional bar chart layer.

The recommended methods from which to call SetSeriesOverlapRatio( ) are:

- A chart’s CustomizeLayers( ) method
- A chart’s AdjustChart( ) method

Syntax Sub SetSeriesOverlapRatio(seriesOverlapRatio As Double)

Parameter **seriesOverlapRatio**
The amount that adjacent series in the bar chart layer overlap, relative to the width of a single bar. Negative values mean there is a gap instead of an overlap. Must be in the range -1 through 1.

Example The following example overrides a chart’s CustomizeLayers( ) method to add a gap between adjacent bars in the chart’s base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmGapsBetweenSeries Then
      baseLayer.SetSeriesOverlapRatio( -0.5 )
   End If
End Sub
```

See also AcChart::AdjustChart method

AcChart::CustomizeLayers method
AcChartLayer::GetSeriesOverlapRatio method
AcChartLayer::SetCategoryGapRatio method

AcChartLayer::SetSeriesPlacement method

Call the SetSeriesPlacement( ) method to set the relative placement of points for multiple series within a category in a chart layer.

SetSeriesPlacement( ) automatically resets the overlap between adjacent bars in bar chart layers as shown in Table 7-11.

Table 7-11 Specifying how to place a series on a chart

seriesPlacement	**Series overlap ratio**
ChartSeriesPlacementAsPercentages	-1
ChartSeriesPlacementOnZAxis	0
ChartSeriesPlacementSideBySide	0
ChartSeriesPlacementStacked	-1

You cannot call this method on chart layers with the following chart types:

- Pie
- Scatter
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetSeriesPlacement(seriesPlacement As AcChartSeriesPlacement)

Parameter **seriesPlacement**
The relative placement of points for multiple series within a category in the chart layer. Must not be set to ChartSeriesPlacementOnZAxis if the chart is not three-dimensional. You must not set Placement to ChartSeriesPlacementOnZAxis if the chart layer's chart type is step.

Example The following example overrides a chart's CustomizeLayers( ) method to make the chart's base layer show series either as percentages or stacked, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmPlotSeriesAsPercentages Then
      baseLayer.SetSeriesPlacement(
         ChartSeriesPlacementAsPercentages )
   Else
      baseLayer.SetSeriesPlacement(
         ChartSeriesPlacementStacked )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetSeriesPlacement method
AcChartLayer::SetChartType method
AcChartLayer::SetSeriesOverlapRatio method
AcChartSeriesPlacement

AcChartLayer::SetStartAngle method

Call the SetStartAngle( ) method to set the angle at which the first slice in a pie chart layer is drawn. The angle is measured in degrees clockwise from vertical.

You can call this method only on a pie chart layer.

The recommended method from which to call SetStartAngle( ) is a chart's CustomizeLayers( ) method.

Syntax Sub SetStartAngle(startAngle As AcAngle)

Parameter **startAngle**
The angle at which the first slice in the pie chart layer is drawn.

Example The following example overrides a chart's CustomizeLayers( ) method to make the chart's base layer show series either as percentages or stacked, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmPlotSeriesAsPercentages Then
      baseLayer.SetSeriesPlacement(
         ChartSeriesPlacementAsPercentages )
   Else
      baseLayer.SetSeriesPlacement(
         ChartSeriesPlacementStacked )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetStartAngle method

AcChartLayer::SetStockHasClose method

Call the SetStockHasClose( ) method to specify whether a stock chart layer has a Close series.

You can call this method only on a stock chart layer.

You cannot call this method on a stock chart layer whose data has been specified using Chart Builder. You must call this method if you are creating data in a stock chart programmatically.

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetStockHasClose(stockHasClose As Boolean)

Parameter **stockHasClose**
True if the stock chart layer has a Close series.
False if the stock chart layer does not have a Close series.

See also AcChart::CustomizeLayers method
AcChartLayer::SetStockHasOpen method
AcChartLayer::StockHasClose method

AcChartLayer::SetStockHasOpen method

Call the SetStockHasOpen( ) method to specify whether a stock chart layer has an Open series. You can call this method only on a stock chart layer. You must call this method if you are creating data in a stock chart programmatically.

You cannot call SetStockHasOpen( ) on a stock chart layer whose data has been specified using Chart Builder.

Call SetStockHasOpen( ) only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetStockHasOpen(stockHasOpen As Boolean)

Parameter **stockHasOpen**
True if the stock chart layer has an Open series.
False if the stock chart layer does not have an Open series.

See also AcChart::CustomizeLayers method
AcChartLayer::SetStockHasClose method
AcChartLayer::StockHasOpen method

AcChartLayer::SetStudyHeightRatio method

Call the SetStudyHeightRatio( ) method to set the ratio of the height of a study layer to the height of its parent chart's base layer. For example, to set the study layer to be half the height of the base layer, call this method with studyHeightRatio set to 0.5.

You can call SetStudyHeightRatio( ) only on a study layer.

The ratio that this method sets is not simply the ratio of the heights of the layer's *y*-axes. The heights of chart layers include their axes, axis labels and some additional space. You can experiment to get the exact appearance you require.

The recommended method from which to call SetStudyHeightRatio( ) is a chart's CustomizeLayers( ) method.

Syntax Sub SetStudyHeightRatio(studyHeightRatio As Double)

Parameter **studyHeightRatio**
The ratio of the height of the study layer to the height of its parent chart's base layer. Must be in the range 0.2 through 5.0.

Example The following example overrides a chart's CustomizeLayers( ) method to increase the height of the chart's first study layer relative to the chart's base layer, depending on the value of a Boolean parameter:

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
    If parmBigStudy Then
        studyLayers(1).SetStudyHeightRatio( 1 )
    End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetStudyHeightRatio method

AcChartLayer::SetThreeDFloorFillStyle method

Call the SetThreeDFloorFillStyle( ) method to set the background fill style for a three-dimensional chart's floor.

The recommended method from which to call SetThreeDFloorFillStyle( ) is a chart's CustomizeLayers( ) method.

You can call this method only on:

- A chart's base layer
- A three-dimensional chart layer

You cannot call this method on a three-dimensional pie chart layer. A three-dimensional pie chart layer does not have walls or a floor.

Syntax Sub SetThreeDFloorFillStyle(threeDFloorFillStyle As AcDrawingFillStyle)

Parameter **threeDFloorFillStyle**
The background fill style for the chart layer's floor.

Example The following example overrides a chart's CustomizeLayers( ) method to create a patterned floor, depending on the value of a Boolean parameter. GetThreeDFloorFillStyle( ) retrieves the default settings so that only the fill style's Pattern member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAddFloorPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetThreeDFloorFillStyle( )
      fillStyle.Pattern = DrawingFillPatternBrickHorizontal
      baseLayer.SetThreeDFloorFillStyle( fillStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetThreeDFloorFillStyle method
AcChartLayer::SetPlotAreaFillStyle method
AcChartLayer::SetThreeDWallFillStyle method
AcDrawingFillStyle

AcChartLayer::SetThreeDWallFillStyle method

Call the SetThreeDWallFillStyle( ) method to set the background fill style for a three-dimensional chart's walls. The recommended method from which to call SetThreeDWallFillStyle( ) is a chart's CustomizeLayers( ) method.

You can call this method only on:

- A chart's base layer
- A three-dimensional chart layer

You cannot call this method on a three-dimensional pie chart layer. A three-dimensional pie chart layer does not have walls or a floor.

You cannot set the fill styles for a three-dimensional chart layer's back wall and side wall independently.

Syntax Sub SetThreeDWallFillStyle(threeDWallFillStyle As AcDrawingFillStyle)

Parameter **threeDWallFillStyle**
The background fill style for the chart layer's walls.

Example The following example overrides a chart's CustomizeLayers( ) method to create patterned walls, depending on the value of a Boolean parameter.

GetThreeDSideWallFillStyle( ) retrieves the default settings so that only the fill style's Pattern member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmAddWallPattern Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetThreeDSideWallFillStyle( )
      fillStyle.Pattern = DrawingFillPattern20Percent
      baseLayer.SetThreeDWallFillStyle( fillStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetThreeDBackWallFillStyle method
AcChartLayer::GetThreeDSideWallFillStyle method
AcChartLayer::SetPlotAreaFillStyle method
AcChartLayer::SetThreeDFloorFillStyle method
AcDrawingFillStyle

AcChartLayer::SetUpBarBorderStyle method

Call the SetUpBarBorderStyle( ) method to set the style of the borders around up bars in a chart layer. To turn off borders around up bars, set the border style's Pen member to DrawingLinePenNone.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetUpBarBorderStyle(UpBarBorderStyle As AcDrawingBorderStyle)

Parameter **UpBarBorderStyle**
The style for borders around up bars in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of the border around up bars in the chart's base layer, depending on the value of a Boolean parameter. GetUpBarBorderStyle( ) retrieves the default settings so that only the border style's Color member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
```

```
   If parmGreenOutlinedUpBars Then
      Dim borderStyle As AcDrawingBorderStyle
      borderStyle = baseLayer.GetUpBarBorderStyle( )
      borderStyle.Color = Green
      baseLayer.SetUpBarBorderStyle( borderStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetUpBarBorderStyle method
AcChartLayer::SetDownBarBorderStyle method
AcChartLayer::SetPlotUpDownBars method
AcChartLayer::SetUpBarFillStyle method
AcDrawingBorderStyle

AcChartLayer::SetUpBarFillStyle method

Call the SetUpBarFillStyle( ) method to set the fill style for up bars in a chart layer.

You can call this method only on chart layers with the following chart types:

- Line
- Stock

You can call this method only from:

- A chart's CustomizeLayers( ) method
- Code that is creating a chart dynamically, after you call the chart's MakeLayers( ) method

Syntax Sub SetUpBarFillStyle(UpBarFillStyle As AcDrawingFillStyle)

Parameter **UpBarFillStyle**
The fill style for up bars in the chart layer.

Example The following example overrides a chart's CustomizeLayers( ) method to change the color of up bars in the chart's base layer, depending on the value of a Boolean parameter. GetUpBarFillStyle( ) retrieves the default settings so that only the fill style's Color1 member needs to change.

```
Sub CustomizeLayers( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmGreenFilledUpBars Then
      Dim fillStyle As AcDrawingFillStyle
      fillStyle = baseLayer.GetUpBarFillStyle( )
      fillStyle.Color = Green
      baseLayer.SetUpBarFillStyle( fillStyle )
   End If
End Sub
```

See also AcChart::CustomizeLayers method
AcChartLayer::GetUpBarFillStyle method
AcChartLayer::SetDownBarFillStyle method
AcChartLayer::SetPlotUpDownBars method
AcChartLayer::SetUpBarBorderStyle method
AcDrawingFillStyle

AcChartLayer::StockHasClose method

Determines whether a stock chart layer has a Close series.

You can call this method only on a stock chart layer.

Syntax Function StockHasClose( ) As Boolean

Description The StockHasClose( ) method determines whether a stock chart layer has a Close series.

Returns True if the stock chart layer has a Close series.
False if the stock chart layer does not have a Close series.

See also AcChart::CustomizeLayers method
AcChartLayer::SetStockHasClose method
AcChartLayer::StockHasOpen method

AcChartLayer::StockHasOpen method

Determines whether a stock chart layer has an Open series.

You can call this method only on a stock chart layer.

Syntax Function StockHasOpen( ) As Boolean

Returns True if the stock chart layer has an Open series.
False if the stock chart layer does not have an Open series.

See also AcChart::CustomizeLayers method
AcChartLayer::SetStockHasOpen method
AcChartLayer::StockHasClose method

Class AcChartPoint

Defines a point within a chart series. Figure 7-10 shows the class hierarchy of AcChartPoint.

Figure 7-10 AcChartPoint

Description Use the AcChartPoint class to represent a single point within a chart series. Do not create AcChartPoint objects explicitly from your own code. Instead, AcChartSeries objects create AcChartPoint objects automatically as necessary to build complete charts.

Use AcChartSeries methods to access a chart series' points. You can manipulate the appearance of a chart by calling methods on the chart's points.

About empty points

If there is no value available for a point in a chart layer that has a category scale, the point still exists but has no value. Such points are called empty or missing points.

For example, a chart shows a count of customers with regions North, South, East, and West as categories and credit ranks A, B, and C as series. There are no customers in the East region with credit rank A. The chart still includes a point in series A for category East but that point is empty.

Customizing points

By default, a chart point is displayed using its parent series' series style settings. If you want an individual point to have a different appearance from the other points in its parent series, add a custom point style, point label value, or point label format to that point.

Example For an example of how to use this class to build a chart dynamically, see the dynamic chart Example for the AcChart class.

See also Class AcChart
Class AcChartAxis
Class AcChartCategory
Class AcChartGridLine
Class AcChartLayer
Class AcChartPointStyle
Class AcChartSeries
Class AcChartSeriesStyle
Class AcChartTrendline

Methods for Class AcChartPoint

Methods defined in Class AcChartPoint

AddCustomStyle, ClearCustomLabelFormat, ClearCustomLabelValue, ClearValues, ExplodeSlice, GetCategory, GetCustomLabelFormat, GetCustomLabelValue, GetCustomStyle, GetIndex, GetLabelText, GetSeries, GetXValue, GetYValue, GetZValue, HasCustomLabelFormat, HasCustomLabelValue, HasCustomStyle, IsMissing, SetCustomLabelFormat, SetCustomLabelValue, SetExplodeSlice, SetValues, SetXValue, SetYValue, SetZValue

AcChartPoint::AddCustomStyle method

Call the AddCustomStyle( ) method to add a custom style to a chart point.

You can call this method only from:

- A chart's AdjustChart( ) method
- Code that is creating a chart dynamically

You cannot add a new custom style to a point that already has a custom style.

When you add a custom style to a point, the new custom style is initialized with the series style values for the point's parent series.

Syntax Function AddCustomStyle( ) As AcChartPointStyle

Returns A reference to the new custom point style.

Example The following example overrides a chart's AdjustChart( ) method to highlight all points in the first series having values greater than 15, using custom point styles:

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim series As AcChartSeries

   ' Get the first series.
   Set series = baseLayer.GetSeries( 1 )

   ' Determine how many points are there in the series
   Dim numberOfPoints As Integer
   numberOfPoints = series.GetNumberOfPoints( )

   ' Loop through all the points in the series.
   Dim pointIndex As Integer
   For pointIndex = 1 To numberOfPoints
      ' Get the point.
      Dim point As AcChartPoint
      Set point = series.GetPoint( pointIndex )
      ' Get the y value of the point.
```

```
        Dim pointValue As Variant
        pointValue = point.GetYValue( )
        If (pointValue > 15) Then
           ' Give the point a custom style.
           Dim pointStyle As AcChartPointStyle
           Set pointStyle = point.AddCustomStyle( )
           ' Color the point green.
           pointStyle.SetBackgroundColor( Green )
           ' Show the point's value as a point label.
           pointStyle.SetPointLabelPlacement( +
              ChartPointLabelPlacementCenter )
           point.SetCustomLabelValue( pointValue )
        End If
     Next pointIndex
  End Sub
```

See also AcChart::AdjustChart method
AcChartPoint::HasCustomStyle method
Class AcChartPointStyle

AcChartPoint::ClearCustomLabelFormat method

Call the ClearCustomLabelFormat( ) method to remove a custom label format pattern from a chart point. You can call this method only from a chart's AdjustChart( ) method.

Syntax Sub ClearCustomLabelFormat( )

See also AcChart::AdjustChart method
AcChartPoint::ClearCustomLabelValue method
AcChartPoint::GetCustomLabelFormat method
AcChartPoint::HasCustomLabelFormat method
AcChartPoint::SetCustomLabelFormat method

AcChartPoint::ClearCustomLabelValue method

Call the ClearCustomLabelValue( ) method to remove a custom label value from a chart point.

You can call this method only from a chart's AdjustChart( ) method.

Syntax Sub ClearCustomLabelValue( )

See also AcChart::AdjustChart method
AcChartPoint::ClearCustomLabelFormat method
AcChartPoint::GetCustomLabelValue method
AcChartPoint::HasCustomLabelValue method
AcChartPoint::SetCustomLabelValue method

AcChartPoint::ClearValues method

Call the ClearValues( ) method to make a chart point into an empty point.

You can call this method only from a chart's AdjustChart( ) method.

Syntax Sub ClearValues( )

See also AcChart::AdjustChart method
AcChartPoint::SetValues method
AcChartPoint::SetXValue method
AcChartPoint::SetYValue method
AcChartPoint::SetZValue method

AcChartPoint::ExplodeSlice method

Determines whether a chart point is a pie chart slice that is exploded.

You can call this method only on a chart point that is a pie chart slice.

You can call this method only after a chart's ComputeScales( ) method has been called.

Syntax Function ExplodeSlice( ) As Boolean

Returns True if the chart point is a pie chart slice that is exploded.
False if the chart point is a pie chart slice that is not exploded.

Example The following example overrides a chart's AdjustChart( ) method to add a point label to each exploded slice in the chart's pie chart base layer:

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' A pie chart layer has only one series - get that series.
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   ' Determine how many slices there are in the pie
   Dim numberOfSlices As Integer
   numberOfSlices = series.GetNumberOfPoints( )
   ' Loop through all the slices.
   Dim sliceIndex As Integer
   For sliceIndex = 1 To numberOfSlices
      ' Get the slice.
      Dim slice As AcChartPoint
      Set slice = series.GetPoint( sliceIndex )
      If slice.ExplodeSlice( ) Then
         ' Give the slice a custom style.
         Dim pointStyle As AcChartPointStyle
         Set pointStyle = slice.AddCustomStyle( )
         ' Show the slice's value as a point label.
```

```
            pointStyle.SetPointLabelPlacement(
               + ChartPointLabelPlacementCenter )
            slice.SetCustomLabelValue( slice.GetYValue( ) )
         End If
      Next sliceIndex
   End Sub
```

See also AcChart::AdjustChart method
AcChartPoint::SetExplodeSlice method

AcChartPoint::GetCategory method

Returns a reference to the chart category corresponding to a chart point.

You can call this method only on a chart point in a chart layer that has a category scale *x*-axis.

Syntax Function GetCategory( ) As AcChartCategory

Returns A reference to the chart category corresponding to the chart point.

See also Class AcChartCategory
AcChartPoint::GetSeries method

AcChartPoint::GetCustomLabelFormat method

Returns the custom format pattern used to format a chart point's label. You can call this method only on a chart point that has a custom label format. To check whether a point has a custom label value, use HasCustomLabelFormat( ).

If a point does not have a custom label format, its point label is formatted using the point label format pattern in the series style corresponding to the point's parent series.

Syntax Function GetCustomLabelFormat( ) As String

Returns The custom format pattern used to format the chart point's label.

See also AcChartPoint::ClearCustomLabelFormat method
AcChartPoint::GetCustomLabelValue method
AcChartPoint::GetLabelText method
AcChartPoint::HasCustomLabelFormat method
AcChartPoint::SetCustomLabelFormat method
AcChartSeriesStyle::GetPointLabelFormat method

AcChartPoint::GetCustomLabelValue method

Returns the custom value of a chart point's label. You can call this method only on a chart point that has a custom label value. Use HasCustomLabelValue( ) to check whether a point has a custom label value. If a point does not have a custom label value, its point label value is calculated from the point label source specified in the series style corresponding to the point's parent series.

Syntax Function GetCustomLabelValue( ) As Variant

Returns The custom value of the chart point's label.

See also AcChartPoint::ClearCustomLabelValue method
AcChartPoint::GetCustomLabelFormat method
AcChartPoint::GetLabelText method
AcChartPoint::HasCustomLabelValue method
AcChartPoint::SetCustomLabelValue method
AcChartSeriesStyle::GetPointLabelSource method

AcChartPoint::GetCustomStyle method

Returns a reference to the custom style for a chart point. You can call this method only on a chart point that has a custom style. To check whether a point has a custom style, use HasCustomStyle( ). To add a custom style to a point, use AddCustomStyle( ).

If a point does not have a custom style, it is displayed using the series style corresponding to the point's parent series.

Syntax Function GetCustomStyle( ) As AcChartPointStyle

Returns A reference to the custom style for the chart point.

See also AcChartPoint::AddCustomStyle method
AcChartPoint::HasCustomStyle method
AcChartSeries::GetStyle method
Class AcChartPointStyle

AcChartPoint::GetIndex method

Returns the index of a chart point within its parent chart series' list of points. The first point in a series is index 1.

Syntax Function GetIndex( ) As Integer

Returns The index of the chart point within its parent chart series' list of points.

AcChartPoint::GetLabelText method

Returns the formatted text of a chart point's label. String label values return unformatted.

Syntax Function GetLabelText( ) As String

Returns The formatted text of the chart point's label.

See also AcChartPoint::GetCustomLabelFormat method
AcChartPoint::GetCustomLabelValue method

AcChartPoint::GetSeries method

Returns a reference to the parent chart series of a chart point.

Syntax Function GetSeries( ) As AcChartSeries

Returns A reference to the parent chart series of the chart point.

See also AcChartPoint::GetCategory method
Class AcChartSeries

AcChartPoint::GetXValue method

Returns the x value of a chart point. You can call this method only on a point in a bubble or scatter chart layer.

Syntax Function GetXValue( ) As Variant

Returns The x value of the chart point.
Null if the chart point is empty.

See also AcChartPoint::GetYValue method
AcChartPoint::GetZValue method
AcChartPoint::IsMissing method
AcChartPoint::SetValues method
AcChartPoint::SetXValue method

AcChartPoint::GetYValue method

Returns the y value of a chart point.

Syntax Function GetYValue( ) As Variant

Returns The y value of the chart point.
Null if the chart point is empty.

See also AcChartCategory::GetSumOfPointValues method
AcChartPoint::GetXValue method
AcChartPoint::GetZValue method
AcChartPoint::IsMissing method
AcChartPoint::SetValues method
AcChartPoint::SetYValue method
AcChartSeries::GetSumOfPointValues method
AcChartSeries::GetSumOfSliceValues method

AcChartPoint::GetZValue method

Returns the z value of a chart point. You can call this method only on a point in a bubble chart layer.

Syntax Function GetZValue( ) As Variant

Returns The z value of a chart point.
Null if the chart point is empty.

See also AcChartPoint::GetXValue method
AcChartPoint::GetYValue method
AcChartPoint::IsMissing method
AcChartPoint::SetValues method
AcChartPoint::SetZValue method

AcChartPoint::HasCustomLabelFormat method

Determines whether a chart point has a custom label format pattern.

Syntax Function HasCustomLabelFormat( ) As Boolean

Returns True if the chart point has a custom label format pattern.
False if the chart point uses the point label format pattern from the point's parent series' series style.

See also AcChartPoint::ClearCustomLabelFormat method
AcChartPoint::GetCustomLabelFormat method
AcChartPoint::HasCustomLabelValue method
AcChartPoint::SetCustomLabelFormat method

AcChartPoint::HasCustomLabelValue method

Determines whether a chart point has a custom label value.

Syntax Function HasCustomLabelValue( ) As Boolean

Returns True if the chart point has a custom label value.
False if the chart point's label value is calculated from the point label source specified in the series style corresponding to the point's parent series.

See also AcChartPoint::ClearCustomLabelValue method
AcChartPoint::GetCustomLabelValue method
AcChartPoint::HasCustomLabelFormat method
AcChartPoint::SetCustomLabelValue method

AcChartPoint::HasCustomStyle method

Determines whether a chart point has a custom style.

Syntax Function HasCustomStyle( ) As Boolean

Returns True if the chart point has a custom style.
False if the chart point is displayed using the series style corresponding to the point's parent series.

See also AcChartPoint::ClearCustomLabelValue method
AcChartPoint::GetCustomLabelValue method
AcChartPoint::HasCustomLabelFormat method
AcChartPoint::SetCustomLabelValue method
AcChartSeries::GetStyle method

AcChartPoint::IsMissing method

Determines whether a chart point is empty.

Syntax Function IsMissing( ) As Boolean

Returns True if the chart point is empty.
False if the chart point has a value.

Example The following example overrides a chart's CustomizeCategoriesAndSeries( ) method to remove any category where the sum of the values of the points in that category is less than 10. The example uses IsMissing( ) to skip points that have no value.

```
Sub CustomizeCategoriesAndSeries(baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )

   ' Loop through all the categories.
   Dim numberOfCategories As Integer
   numberOfCategories = baseLayer.GetNumberOfCategories( )
   Dim categoryIndex As Integer

   ' Use reverse order so that deleting categories
   ' does not invalidate the current index.
   For categoryIndex = numberOfCategories To 1 Step -1

      ' Add all the values in the category.
      Dim total As Double
      total = 0
      Dim numberOfSeries As Integer
      numberOfSeries = baseLayer.GetNumberOfSeries( )
      Dim seriesIndex As Integer
      For seriesIndex = 1 To numberOfSeries
         Dim series As AcChartSeries
         Set series = baseLayer.GetSeries( seriesIndex )
         Dim point As AcChartPoint
         Set point = series.GetPoint( categoryIndex )
         ' Ignore missing values.
         If Not point.IsMissing( ) Then
            total = total + point.GetYValue( )
         End If
      Next seriesIndex
      ' Remove categories whose values total less than 10.
      If (total < 10) Then
         baseLayer.RemoveCategory( categoryIndex )
      End If
   Next categoryIndex
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartPoint::ClearValues method
AcChartPoint::SetValues method
AcChartPoint::SetXValue method
AcChartPoint::SetYValue method
AcChartPoint::SetZValue method

AcChartPoint::SetCustomLabelFormat method

Call the SetCustomLabelFormat( ) method to add a custom label format pattern to a chart point. The format pattern is ignored for string label values.

You can call SetCustomLabelFormat( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

If a point does not have a custom label format, its point label is formatted using the point label format pattern in the series style corresponding to the point's parent series.

Syntax Sub SetCustomLabelFormat(customLabelFormat As String)

Parameter **customLabelFormat**
The custom label format pattern for the chart point.

Example For an example of how to use this method, see the example for the AcChartPoint::SetCustomLabelValue( ) method.

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartPoint::ClearCustomLabelFormat method
AcChartPoint::GetCustomLabelFormat method
AcChartPoint::HasCustomLabelFormat method
AcChartPoint::SetCustomLabelValue method
AcChartSeriesStyle::SetPointLabelFormat method

AcChartPoint::SetCustomLabelValue method

Call the SetCustomLabelValue( ) method to add a custom label value to a chart point.

You can call SetCustomLabelValue( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

If a point does not have a custom label value, its point label value is calculated from the point label source specified in the series style corresponding to the point's parent series.

A point label is displayed only for a point if that point's point label placement setting is not ChartPointLabelPlacementNone. You can change the point label placement settings for points by:

- Specifying the point label placement in Advanced Chart Options
- Setting a default point label placement for all points in a chart layer using the layer's SetPointLabelPlacement( ) method
- Setting a default point label placement for all points in a chart series using the SetPointLabelPlacement( ) method of the series style corresponding to the series
- Adding custom chart point styles to individual chart points and using the SetPointLabelPlacement( ) method of those styles

Syntax Sub SetCustomLabelValue(customLabelValue As Variant)

Parameter **customLabelValue**
The custom label value for the chart point.

Example The following example overrides a chart's AdjustChart( ) method to add a special message to the point label for the point with the highest value in the first series in the chart's base layer:

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   Dim numberOfPoints As Integer
   numberOfPoints = series.GetNumberOfPoints( )
   Dim pointIndex As Integer
   For pointIndex = 1 To numberOfPoints
      Dim point As AcChartPoint
      Dim maxPoint As AcChartPoint
      Set point = series.GetPoint( pointIndex )
```

```
            ' Ignore missing values.
            If Not point.IsMissing( ) Then
               If maxPoint Is Nothing Then
                  Set maxPoint = point
               ElseIf (maxPoint.GetYValue( ) < point.GetYValue( )) Then
                  Set maxPoint = point
               End If
            End If
         Next pointIndex
         If Not maxPoint Is Nothing Then
            ' Get the standard point label text.
            Dim pointLabelText As String
            pointLabelText = maxPoint.GetLabelText( )
            ' Suppress the numeric point label format.
            maxPoint.SetCustomLabelFormat( "" )
            ' Add a special message to the point label.
            maxPoint.SetCustomLabelValue( pointLabelText & " - Best
               Region!" )
         End If
      End Sub
```

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartLayer::SetPointLabelPlacement method
AcChartPoint::ClearCustomLabelValue method
AcChartPoint::GetCustomLabelValue method
AcChartPoint::HasCustomLabelValue method
AcChartPoint::SetCustomLabelFormat method
AcChartPointStyle::SetPointLabelPlacement method
AcChartSeriesStyle::SetPointLabelSource method

AcChartPoint::SetExplodeSlice method

Call the SetExplodeSlice( ) method to specify whether a chart point that is a pie slice is exploded.

You can call SetExplodeSlice( ) from:

- A chart's AdjustChart( ) method
- Code that is creating a chart dynamically, after you have called the chart's ComputeScales( ) method
- A chart's Localize( ) method

To enable pie slice explosion, use a chart layer's SetPieExplosion( ) method.

Syntax Sub SetExplodeSlice(explode As Boolean)

Parameter **explode**
True causes the pie slice to be exploded. False causes the pie slice not to be exploded.

Example The following example overrides a chart's Localize( ) method to explode the pie slice whose category key corresponds to the user's viewing locale, so that users in different locales see different slices exploded without re-running the report:

```
Sub Localize( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' Convert the user's locale to a country name.
   Dim localeName As String
   localeName = GetLocaleName( )
   Dim userCountry As String
   Select Case localeName
   Case "fr_FR"
      userCountry = "France"
   Case "es_ES"
      userCountry = "Spain"
   Case "en_US"
      userCountry = "USA"
   End Select
   ' Enable conditional pie slice explosion and get the series.
   baseLayer.SetPieExplosion( ChartPieExplodeSpecificSlices )
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   ' Determine how many slices are in the pie
   Dim numberOfSlices As Integer
   numberOfSlices = series.GetNumberOfPoints( )
   ' Loop through all the slices.
   Dim sliceIndex As Integer
   For sliceIndex = 1 To numberOfSlices
      ' Get the slice.
      Dim slice As AcChartPoint
      Set slice = series.GetPoint( sliceIndex )
      Dim category As AcChartCategory
      Set category = slice.GetCategory( )
      ' Explode slices whose country name is the user's
      ' country.
      slice.SetExplodeSlice( category.GetKeyValue( ) =
         userCountry )
   Next sliceIndex
End Sub
```

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartLayer::SetPieExplosion method
AcChartPoint::ExplodeSlice method

AcChartPoint::SetValues method

Call the SetValues( ) method to set the values of a chart point.

You can call this method only on a point in a scatter chart layer.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax

```
Sub SetValues( xValue As Variant, yValue As Variant )
```

```
Sub SetValues( xValue As Variant, yValue As Variant, zValue As Variant )
```

Parameters

xValue
The x value for the point. If Null, the point is made into an empty point.

yValue
The y value for the point. If Null, the point is made into an empty point.

zValue
The z value for the point. If Null, the point is made into an empty point.

See also

AcChart::CustomizeCategoriesAndSeries method
AcChartPoint::ClearValues method
AcChartPoint::SetXValue method
AcChartPoint::SetYValue method
AcChartPoint::SetZValue method

AcChartPoint::SetXValue method

Call the SetXValue( ) method to set the x value of a chart point.

You can call this method only on a point in a bubble or scatter chart layer.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax

```
Sub SetXValue( xValue As Variant )
```

Parameter

xValue
The x value for the point. If Null, the point turns into an empty point.

See also

AcChart::CustomizeCategoriesAndSeries method
AcChartPoint::ClearValues method
AcChartPoint::GetXValue method
AcChartPoint::SetValues method

AcChartPoint::SetYValue method
AcChartPoint::SetZValue method

AcChartPoint::SetYValue method

Call the SetYValue( ) method to set the y value of a chart point.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetYValue(yValue As Variant)

Parameter **yValue**
The y value for the point. If Null, the point is made into an empty point.

See also AcChart::CustomizeCategoriesAndSeries method
AcChartPoint::ClearValues method
AcChartPoint::GetYValue method
AcChartPoint::SetValues method
AcChartPoint::SetXValue method
AcChartPoint::SetZValue method

AcChartPoint::SetZValue method

Call SetZValue( ) to set the z value of a chart point.

You can call this method only on a point in a bubble chart layer.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetZValue(zValue As Variant)

Parameter **zValue**
The z value for the point. If Null, the point is made into an empty point.

See also AcChart::CustomizeCategoriesAndSeries method
AcChartPoint::ClearValues method
AcChartPoint::GetZValue method
AcChartPoint::SetValues method
AcChartPoint::SetXValue method
AcChartPoint::SetYValue method

Class AcChartPointStyle

A custom style for a chart point. Figure 7-11 shows the class hierarchy of AcChartPointStyle.

Figure 7-11 AcChartPointStyle

Description Use the AcChartPointStyle class to represent a custom style for a single point within a chart series. Do not create AcChartPointStyle objects explicitly from your own code. Instead, AcChartPoint objects create AcChartPointStyle objects automatically as necessary to build complete charts.

Use AcChartPoint’s methods to create and access a chart point’s custom style. You can manipulate the appearance of a chart by calling methods on the chart’s custom point styles.

A chart point’s default behavior is to use its parent series’ series style settings. If you want an individual point to have a different appearance from the other points in its parent series, you can add a custom point style. When you add a custom style to a point, the new custom style is initialized with the series style values for the point’s parent series.

Example The following example overrides a chart’s AdjustChart( ) method to change the appearance of all points in the first series in the chart’s base layer whose values are greater than 15:

- Diagonal yellow stripes are added to the points. GetFillStyle( ) retrieves the default settings so that the points’ background color is preserved.
- The color of the borders around the points is changed to red. GetBorderStyle( ) retrieves the default settings so that only the border style’s Color member needs to change.
- Centered point labels are added to the points.
- The point’s point labels are given borders and white backgrounds. GetPointLabelStyle( ) retrieves the default settings so that only the border pen and background color need to change.

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' Get the first series.
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   ' How many points are there in the series?
   Dim numberOfPoints As Integer
   numberOfPoints = series.GetNumberOfPoints( )
   ' Loop through all the points in the series.
   Dim pointIndex As Integer
   For pointIndex = 1 To numberOfPoints
      ' Get the point.
      Dim point As AcChartPoint
      Set point = series.GetPoint( pointIndex )
      ' Get the y value of the point.
      Dim pointValue As Variant
      pointValue = point.GetYValue( )
      If (pointValue > 15) Then
         ' Give the point a custom style.
         Dim pointStyle As AcChartPointStyle
         Set pointStyle = point.AddCustomStyle( )
         ' Add diagonal yellow stripes to the point.
         Dim fillStyle As AcDrawingFillStyle
         fillStyle = pointStyle.GetFillStyle( )
         fillStyle.Color2 = Yellow
         fillStyle.Pattern = DrawingFillPatternDiagonalUpWide
         pointStyle.SetFillStyle( fillStyle )
         ' Give the point a red border.
         Dim borderStyle As AcDrawingBorderStyle
         borderStyle = pointStyle.GetBorderStyle( )
         borderStyle.Color = Red
         pointStyle.SetBorderStyle( borderStyle )
         ' Show the point's value as a point label.
         pointStyle.SetPointLabelPlacement(
            + ChartPointLabelPlacementCenter )
         point.SetCustomLabelValue( pointValue )
         ' Give the point label a white background and a border.
         Dim labelStyle As AcDrawingTextStyle
         labelStyle = pointStyle.GetPointLabelStyle( )
         labelStyle.Border.Pen = DrawingLinePenSolid
         labelStyle.BackgroundColor = White
         pointStyle.SetPointLabelStyle( labelStyle )
      End If
   Next pointIndex
End Sub
```

See also

Class AcChart
Class AcChartAxis
Class AcChartCategory
Class AcChartGridLine
Class AcChartLayer
Class AcChartPoint
Class AcChartSeries
Class AcChartSeriesStyle
Class AcChartTrendline

Methods for Class AcChartPointStyle

Methods defined in Class AcChartPointStyle

GetBorderStyle, GetFillStyle, GetMarkerFillColor, GetMarkerLineColor, GetMarkerShape, GetMarkerSize, GetPieExplosionAmount, GetPointLabelPlacement, GetPointLabelStyle, SetBackgroundColor, SetBorderStyle, SetFillStyle, SetMarkerFillColor, SetMarkerLineColor, SetMarkerShape, SetMarkerSize, SetPieExplosionAmount, SetPointLabelPlacement, SetPointLabelStyle

AcChartPointStyle::GetBorderStyle method

Returns the style of the border around a chart point. To change the border around a chart point, call this method on the point's point style to get the default settings. You can call this method only on point styles for points in chart layers with the following chart types:

- Area
- Bar
- Pie
- Step

Syntax Function GetBorderStyle( ) As AcDrawingBorderStyle

Returns The style of the border around the chart point.

Example For an example of how to use this method, see the example for the AcChartPointStyle class.

See also AcChartLayer::GetPointBorderStyle method
AcChartPointStyle::SetBorderStyle method
Class AcChartPointStyle
AcDrawingBorderStyle

AcChartPointStyle::GetFillStyle method

Returns the background fill style for a chart point. To change the fill style for a chart point, call this method on the point's point style to get the default settings. You can call this method only on point styles for points in chart layers with the following chart types:

- Area
- Bar
- Pie
- Step

Syntax Function GetFillStyle( ) As AcDrawingFillStyle

Returns The background fill style for the chart point.

Example For an example of how to use this method, see the example for the AcChartPointStyle class.

See also AcChartPointStyle::SetBackgroundColor method
AcChartPointStyle::SetFillStyle method
Class AcChartPointStyle
AcDrawingFillStyle

AcChartPointStyle::GetMarkerFillColor method

Returns the fill color of the marker for a chart point.

You can call this method only on point styles for points in chart layers with the following chart types:

- Line
- Scatter
- Stock

Syntax Function GetMarkerFillColor( ) As AcColor

Returns The fill color of the marker for the chart point.

See also AcChartPointStyle::GetMarkerLineColor method
AcChartPointStyle::SetMarkerFillColor method

AcChartPointStyle::GetMarkerLineColor method

Returns the line color of the marker for a chart point.

You can call this method only on point styles for points in chart layers with the following chart types:

- Line
- Scatter
- Stock

Syntax Function GetMarkerLineColor( ) As AcColor

Returns The line color of the marker for the chart point.

See also AcChartPointStyle::GetMarkerFillColor method
AcChartPointStyle::SetMarkerLineColor method

AcChartPointStyle::GetMarkerShape method

Returns the shape of the marker for a chart point.

You can call this method only on point styles for points in chart layers with the following chart types:

- Line
- Scatter
- Stock

Syntax Function GetMarkerShape( ) As AcChartMarkerShape

Returns The shape of the marker for the chart point.
ChartMarkerShapeNone if no marker is displayed at the point.

See also AcChartMarkerShape
AcChartPointStyle::SetMarkerShape method

AcChartPointStyle::GetMarkerSize method

Returns the size of the marker for a chart point. You can call this method only on point styles for points in chart layers with the following chart types:

- Line
- Scatter
- Stock

Syntax Function GetMarkerSize( ) As AcTwips

Returns The size of the marker for the chart point.

See also AcChartLayer::GetMarkerSize method
AcChartPointStyle::SetMarkerSize method
AcTwips

AcChartPointStyle::GetPieExplosionAmount method

Returns the amount that a pie slice chart point is exploded in a pie chart layer. The amount is relative to the radius of the pie. If the amount is 0.25, the slice is moved outward from the center of the pie by one quarter of the radius of the pie.

You can call this method only on point styles for points in pie chart layers.

Syntax Function GetPieExplosionAmount( ) As Double

Returns The amount that a pie slice chart point is exploded in a pie chart layer.

See also AcChartLayer::GetPieExplosionAmount method
AcChartPointStyle::SetPieExplosionAmount method

AcChartPointStyle::GetPointLabelPlacement method

Returns the placement of the point label for a chart point.

Syntax Function GetPointLabelPlacement( ) As AcChartPointLabelPlacement

Returns The placement of the point label for the chart point.

See also AcChartLayer::GetPointLabelPlacement method
AcChartPointLabelPlacement
AcChartPointStyle::SetPointLabelPlacement method

AcChartPointStyle::GetPointLabelStyle method

Returns the style of the point label for a chart point. To change the style of a point's point label, call this method to retrieve the default settings.

Syntax Function GetPointLabelPlacement( ) As AcChartPointLabelPlacement

Returns The style of the point label for the chart point.

Example For an example of how to use this method, see the example for the AcChartPointStyle class.

See also AcChartLayer::GetPointLabelStyle method
AcChartPointStyle::GetPointLabelStyle method
AcDrawingTextStyle

AcChartPointStyle::SetBackgroundColor method

Call the SetBackgroundColor( ) method to set the background color for a chart point. This method sets a chart point's fill style to a single solid color. This method sets a chart point's fill style members as follows:

- Sets the Color1 member to the specified background color
- Does not affect the Color2 member
- Sets the Pattern member to DrawingFillPatternSolid

You can call this method only on point styles for points in chart layers with the following chart types:

- Area
- Bar
- Pie
- Step

You can call SetBackgroundColor( ) from:

- A chart's AdjustChart( ) method

- A chart's Localize( ) method
- Code that is creating a chart dynamically

Syntax Sub SetBackgroundColor(customLabelFormat As String)

Parameter **customLabelFormat**
The custom label format pattern for the chart point.

Example The following example overrides a chart's AdjustChart( ) method to set the color of all unexploded slices in the chart's pie chart base layer to light gray:

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' A pie chart layer has only one series. Get that series.
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )

   ' Determine how many slices are in the pie
   Dim numberOfSlices As Integer
   numberOfSlices = series.GetNumberOfPoints( )

   ' Loop through all the slices.
   Dim sliceIndex As Integer
   For sliceIndex = 1 To numberOfSlices
      ' Get the slice.
      Dim slice As AcChartPoint
      Set slice = series.GetPoint( sliceIndex )
      If Not slice.ExplodeSlice( ) Then
         ' Give the slice a custom style.
         Dim pointStyle As AcChartPointStyle
         Set pointStyle = slice.AddCustomStyle( )
         pointStyle.SetBackgroundColor( LightGray )
      End If
   Next sliceIndex
End Sub
```

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartPointStyle::GetFillStyle method
AcChartPointStyle::SetFillStyle method

AcChartPointStyle::SetBorderStyle method

Call the SetBorderStyle( ) method to set the style of the border around a chart point. You can call SetBorderStyle( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

You can call this method only on point styles for points in chart layers with the following chart types:

- Area
- Bar
- Pie
- Step

Syntax Sub SetBorderStyle(borderStyle As AcDrawingBorderStyle)

Parameter **borderStyle**
The style for the border around the chart point.

Example For an example of how to use this method, see the example for the AcChartPointStyle class.

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartLayer::SetPointBorderStyle method
AcChartPointStyle::GetBorderStyle method
AcChartPointStyle
AcDrawingBorderStyle

AcChartPointStyle::SetFillStyle method

Call the SetFillStyle( ) method to set the background fill style for a chart point.

You can call SetFillStyle( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

You can call this method only on point styles for points in chart layers with the following chart types:

- Area
- Bar
- Pie
- Step

Syntax Sub SetFillStyle(fillStyle As AcDrawingFillStyle)

Parameter **fillStyle**
The background fill style for the chart point.

Example For an example of how to use this method, see the example for the AcChartPointStyle class.

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartPointStyle::GetFillStyle method
AcChartPointStyle::SetBackgroundColor method
AcChartPointStyle
AcDrawingFillStyle

AcChartPointStyle::SetMarkerFillColor method

Call the SetMarkerFillColor( ) method to set the fill color of the marker for a chart point. You can call this method only on point styles for points in chart layers with the following chart types:

- Line
- Scatter
- Stock

You can call SetMarkerFillColor( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

Syntax

```
Sub SetMarkerFillColor( markerFillColor As AcColor )
```

Parameter **markerFillColor**
The fill color of the marker for the chart point.

Example For an example of how to use this method, see the example for the AcChartSeriesStyle class.

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartPointStyle::GetMarkerFillColor method
AcChartPointStyle::SetMarkerLineColor method
Class AcChartSeriesStyle

AcChartPointStyle::SetMarkerLineColor method

Call the SetMarkerLineColor( ) method to set the line color of the marker for a chart point.

You can call this method only on point styles for points in chart layers with the following chart types:

- Line
- Scatter

- Stock

You can call SetMarkerLineColor( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

Syntax Sub SetMarkerLineColor(markerLineColor As AcColor)

Parameter **markerLineColor**
The line color of the marker for the chart point.

Example For an example of how to use this method, see the example for the AcChartSeriesStyle class.

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartPointStyle::GetMarkerLineColor method
AcChartPointStyle::SetMarkerFillColor method
Class AcChartSeriesStyle

AcChartPointStyle::SetMarkerShape method

Call the SetMarkerShape( ) method to set the shape of the marker for a chart point.

You can call this method only on point styles for points in chart layers with the following chart types:

- Line
- Scatter
- Stock

You can call SetMarkerShape( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

Syntax Sub SetMarkerShape(markerShape As AcChartMarkerShape)

Parameter **markerShape**
The shape of the marker for the chart point. To turn off the marker at the point, set markerShape to ChartMarkerShapeNone.

Example For an example of how to use this method, see the example for the AcChartSeriesStyle class.

See also AcChart::AdjustChart method

AcChart::Localize method
AcChartMarkerShape
AcChartPointStyle::GetMarkerShape method
Class AcChartSeriesStyle

AcChartPointStyle::SetMarkerSize method

Call the SetMarkerSize( ) method to set the size of the marker for a chart point. You can call this method only on point styles for points in chart layers with the following chart types:

- Line
- Scatter
- Stock

You can call SetMarkerSize( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

Syntax Sub SetMarkerSize(markerSize As AcTwips)

Parameter **markerSize**
The size of the marker for the chart point.

Example For an example of how to use this method, see the example for the AcChartSeriesStyle class.

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartLayer::SetMarkerSize method
AcChartPointStyle::GetMarkerSize method
Class AcChartSeriesStyle
AcTwips

AcChartPointStyle::SetPieExplosionAmount method

Call the SetPieExplosionAmount( ) method to set the amount that a pie slice chart point is exploded in a pie chart layer. The amount is relative to the radius of the pie. If the amount is 0.25, the slice is moved outward from the center of the pie by one quarter of the radius of the pie.

You can call this method only on point styles for points in pie chart layers.

You can call SetPieExplosionAmount( ) from:

- A chart's AdjustChart( ) method

- A chart's Localize( ) method
- Code that is creating a chart dynamically

Syntax Sub SetPieExplosionAmount(pieExplosionAmount As Double)

Parameter **pieExplosionAmount**
The amount that the pie slice chart point is exploded. Must be in the range 0 through 0.4.

Example The following example overrides a chart's AdjustChart( ) method to explode pie chart slices in the chart's base layer in proportion to the percentages of the total pie represented by those slices:

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' Explode all pie slices.
   baseLayer.SetPieExplosion( ChartPieExplodeAllSlices )
   ' A pie chart layer has only one series - get that series.
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   ' Determine the total value of the pie
   Dim sumOfSliceValues As Variant
   sumOfSliceValues = series.GetSumOfSliceValues( )
   ' Determine how many slices are in the pie
   Dim numberOfSlices As Integer
   numberOfSlices = series.GetNumberOfPoints( )
   ' Loop through all the slices.
   Dim sliceIndex As Integer
   For sliceIndex = 1 To numberOfSlices
      ' Get the slice.
      Dim slice As AcChartPoint
      Set slice = series.GetPoint( sliceIndex )
      ' Compute the explosion amount.
      Dim explosionAmount As Double
      explosionAmount = slice.GetYValue( ) / sumOfSliceValues
      If (explosionAmount > 0.4) Then
         explosionAmount = 0.4
      End If
      ' Give the slice a custom style.
      Dim pointStyle As AcChartPointStyle
      Set pointStyle = slice.AddCustomStyle( )
      pointStyle.SetPieExplosionAmount( explosionAmount )
   Next sliceIndex
End Sub
```

See also AcChart::AdjustChart method
AcChart::Localize method
AcChartLayer::SetPieExplosionAmount method
AcChartPointStyle::GetPieExplosionAmount method

AcChartPointStyle::SetPointLabelPlacement method

Call the SetPointLabelPlacement( ) method to set the placement of the point label for a chart point. You can call SetPointLabelPlacement( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

Syntax

```
Sub SetPointLabelPlacement( pointLabelPlacement As
    AcChartPointLabelPlacement )
```

Parameter

pointLabelPlacement
The placement of the point label for the chart point. To turn off the point label, set pointLabelPlacement to ChartPointLabelPlacementNone.

Example

For an example of how to use this method, see the AcChartPointStyle class.

See also

AcChart::AdjustChart method
AcChart::Localize method
AcChartLayer::SetPointLabelPlacement method
AcChartPointLabelPlacement
Class AcChartPointStyle
AcChartPointStyle::GetPointLabelPlacement method
AcDrawingTextStyle

AcChartPointStyle::SetPointLabelStyle method

Call the SetPointLabelStyle( ) method to set the style of the point label for a chart point. You can call SetPointLabelStyle( ) from:

- A chart's AdjustChart( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically

Syntax

```
Sub SetPointLabelStyle( pointLabelStyle As AcDrawingTextStyle )
```

Parameter

pointLabelStyle
The style of the point label for the chart point.

Example

For an example of how to use this method, see the AcChartPointStyle class.

See also

AcChart::AdjustChart method
AcChart::Localize method
AcChartLayer::SetPointLabelStyle method
AcChartPointStyle::GetPointLabelStyle method
Class AcChartPointStyle
AcDrawingTextStyle

Class AcChartSeries

A series within a chart layer. Figure 7-12 shows the class hierarchy of AcChartSeries.

Figure 7-12 AcChartSeries

Description Use the AcChartSeries class to represent a single series within a chart layer. Do not create AcChartSeries objects explicitly from your own code. AcChartLayer objects create AcChartSeries objects automatically as necessary to build complete charts. Use AcChartLayer′s methods to access a chart layer′s series. You can manipulate the appearance of a chart by calling methods on the chart's series.

All types of chart layer except pie chart layers have at least one series. Pie chart layers have only one series.

Example For an example of how to use this class to build a chart dynamically, see the dynamic chart example for the AcChart class.

See also Class AcChart
Class AcChartAxis
Class AcChartCategory
Class AcChartGridLine
Class AcChartLayer
Class AcChartPoint
Class AcChartPointStyle
Class AcChartSeriesStyle
Class AcChartTrendline

Methods for Class AcChartSeries

Methods defined in Class AcChartSeries

AddEmptyPoint, AddPoint, AddTrendline, GetIndex, GetKeyValue, GetLabelText, GetLabelValue, GetLayer, GetNumberOfPoints, GetNumberOfTrendlines, GetPoint, GetStyle, GetSumOfPointValues, GetSumOfSliceValues, GetTrendline, InsertEmptyPoint, InsertPoint, InsertTrendline, RemovePoint, RemoveTrendline, SetKeyValue, SetLabelValue

AcChartSeries::AddEmptyPoint method

Call the AddEmptyPoint( ) method to append a new empty point to the end of a chart series' list of points. You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method

- Code that is creating a chart dynamically, after you have set the chart's status to ChartStatusBuilding

Syntax Function AddEmptyPoint( ) As AcChartPoint

Returns A reference to the new point.

Example For an example of using this method, see the dynamic chart example for AcChart.

See also AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::AddPoint method
AcChartSeries::InsertEmptyPoint method
AcChartSeries::RemovePoint method
Class AcChart
Class AcChartPoint

AcChartSeries::AddPoint method

Call the AddPoint( ) method to append a new point to the end of a chart series' list of points. You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, after you have set the chart's status to ChartStatusBuilding

If xValue, yValue, or zValue is Null, the new point will be an empty point.

Syntaxes

```
Function AddPoint( yValue As Variant ) As AcChartPoint

Function AddPoint( xValue As Variant, yValue As Variant ) As AcChartPoint

Function AddPoint( xValue As Variant, yValue As Variant,
    zValue As Variant ) As AcChartPoint
```

Parameters

xValue
The x value for the new point. You can specify xValue only if you are adding a point to a series in a bubble or scatter chart layer.

yValue
The y value for the new point.

zValue
The z value for the new point. You can specify zValue only if you are adding a point to a series in a bubble chart layer.

Returns A reference to the new point.

Example For an example of using this method, see the dynamic chart example for AcChart.

See also AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::AddEmptyPoint method
AcChartSeries::InsertPoint method

AcChartSeries::RemovePoint method
Class AcChart
Class AcChartPoint

AcChartSeries::AddTrendline method

Call this method to add a trendline to the end of a chart series' list of trendlines. You can call this method only from:

- A chart's CustomizeCategoriesAndSeries method
- Code that creates a chart dynamically

Syntax

```
Function AddTrendline( labelText As String,
    trendlineType As AcChartTrendlineType ) As AcChartTrendline
```

Parameters

labeText
The text shown in the chart legend for the trendline is drawn. Null or "" if you do not want the trendline to be listed in the chart's legend.

trendlineType
Defines how the trendline will be fitted to the points in its parent series.

Returns A handle to the new trendline object.

Example For an example of how to use this method, see the example for the AcChartTrendline class.

See also
AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::InsertTrendline method
AcChartSeries::RemoveTrendline method
AcChartTrendline
AcChartType

AcChartSeries::GetIndex method

Returns the index of a chart series within its parent chart layer's list of series.

Syntax

```
Function GetIndex( ) As Integer
```

Returns The 1-based index of the chart series within its parent chart layer's list of series.

AcChartSeries::GetKeyValue method

Returns the unique key value for a chart series.

Syntax

```
Function GetKeyValue( ) As Variant
```

Returns The unique key value for the chart series.

See also
AcChartSeries::GetLabelValue method
AcChartSeries::SetKeyValue method

AcChartSeries::GetLabelText method

Returns the formatted label text for a chart series. String label values are returned unformatted.

Syntax Function GetLabelText( ) As String

Returns The formatted label text for a chart series.

See also AcChartSeries::GetKeyValue method
AcChartSeries::GetLabelValue method
AcChartSeries::SetLabelValue method

AcChartSeries::GetLabelValue method

Returns the label value for a chart series.

Syntax Function GetLabelValue( ) As Variant

Returns The label value for a chart series.

See also AcChartSeries::GetKeyValue method
AcChartSeries::GetLabelText method
AcChartSeries::SetLabelValue method

AcChartSeries::GetLayer method

Returns a reference to the parent chart layer of a chart series.

Syntax Function GetLayer( ) As AcChartLayer

Returns A reference to the parent chart layer of the chart series.

See also AcChartLayer

AcChartSeries::GetNumberOfPoints method

Returns the number of points in a chart series.

Syntax Function GetNumberOfPoints( ) As Integer

Returns The number of points in the chart series.

AcChartSeries::GetNumberOfTrendlines method

Determines the number of trendlines in a chart series.

Syntax Function GetNumberOfTrendlnes( ) As Integer

Returns The number of trendlines in the chart series.

See also AcChartSeries::GetTrendline method

AcChartTrendline

AcChartSeries::GetPoint method

Returns a reference to a point in a chart series. To retrieve the number of points in a chart series, call the series' GetNumberOfPoints( ) method.

Syntax Function GetPoint(index As Integer) As AcChartPoint

Parameter **index**
An index into the series' list of points. The first point is index 1.

Returns A reference to the specified point in the chart series.

See also AcChartSeries::GetNumberOfPoints method
Class AcChartPoint

AcChartSeries::GetStyle method

Returns a reference to the series style corresponding to a chart series. You cannot call GetStyle( ) on a series in a pie chart layer. A pie chart layer has only one series. Each slice in a pie corresponds to a category, not a series. Series styles in pie chart layers correspond to categories, not series. To retrieve the series styles for pie slices, use a pie chart layer's GetSeriesStyle( ) method.

Syntax Function GetStyle( ) As AcChartSeriesStyle

Returns A reference to the series style corresponding to the chart series.

See also AcChartLayer::GetSeriesStyle method
AcChartSeriesStyle

AcChartSeries::GetSumOfPointValues method

Returns the sum of the y values of all the points in a chart series.

Syntax Function GetSumOfPointValues( ) As Variant

Returns The sum of the y values of all the points in the series.

See also AcChartCategory::GetSumOfPointValues method
AcChartPoint::GetYValue method
AcChartSeries::GetSumOfSliceValues method

AcChartSeries::GetSumOfSliceValues method

Returns the sum of the values of all the slices in a pie chart series.

You can call this method only on a series in a pie chart layer.

Syntax Function GetSumOfSliceValues( ) As Variant

Returns The sum of the values of all the slices in the pie chart series.

See also AcChartPoint::GetYValue method
AcChartSeries::GetSumOfPointValues method

AcChartSeries::GetTrendline method

Returns a reference to the specified trendline for a chart series. To determine the number of trendlines in a chart series, call the chart series' GetNumberOfTrendlines( ) method.

Syntax Function GetTrendline(index As Integer) As AcChartTrendline

Parameter **index**
An index into the chart series's list of trendlines. The first trendline is index 1.

Returns A reference to the specified trendline for the chart series.

See also AcChartSeries::GetNumberOfTrendlines method
AcChartTrendline

AcChartSeries::InsertEmptyPoint method

Call the InsertEmptyPoint( ) method to insert a new empty point at a specific position in a chart series' list of points. You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, after you have set the chart's status to ChartStatusBuilding

Syntax Function InsertEmptyPoint(index As Integer) As AcChartPoint

Parameter **index**
The position in the chart series' list of points at which the new empty point will be inserted. The first point is index 1. Must be greater than or equal to one. Must be less than or equal to the current number of points in the chart series plus one.

Returns A reference to the new empty point.

See also AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::AddEmptyPoint method
AcChartSeries::InsertPoint method
AcChartSeries::RemovePoint method
AcChartPoint

AcChartSeries::InsertPoint method

Call the InsertPoint( ) method to insert a new point at a specific position in a chart series' list of points. You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, after you have set the chart's status to ChartStatusBuilding

If xValue, yValue, or zValue is Null, the new point will be an empty point.

Syntaxes Function InsertPoint(index As Integer, yValue AsVariant) As AcChartPoint

Function InsertPoint(index As Integer, xValue As Variant,
yValue As Variant) As AcChartPoint

Function InsertPoint(index As Integer, xValue As Variant,
yValue As Variant, zValue As Variant) As AcChartPoint

Parameter **index**
The position in the chart series' list of points at which the new empty point will be inserted. The first point is index 1. Must be greater than or equal to one. Must be less than or equal to the current number of points in the chart series plus one.

xValue
The x value for the new point. You can specify xValue only if you are adding a point to a series in a bubble or scatter chart layer.

yValue
The y value for the new point.

zValue
The z value for the new point. You can specify zValue only if you are adding a point to a series in a bubble chart layer.

Returns A reference to the new point.

Example The following example overrides a chart's CustomizeCategoriesAndSeries( ) method to insert a new series into the chart's base layer. The new series appear as the first series on the chart's x-axis. Each point in the new series is populated with the mean value of the points in the same category for all the other series.

```
Sub CustomizeCategoriesAndSeries(baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   ' Insert a new series.
   Dim newSeries As AcChartSeries
   Set newSeries = baseLayer.InsertSeries( 1, "Mean" )

   ' Loop through all the categories.
   Dim numberOfCategories As Integer
   numberOfCategories = baseLayer.GetNumberOfCategories( )
   Dim categoryIndex As Integer
   For categoryIndex = 1 To numberOfCategories
      ' Get the mean value of all the points in this category.
      Dim point As AcChartPoint
      Dim total As Double
      total = 0
      Dim count As Integer
      count = 0
      Dim numberOfSeries As Integer
      numberOfSeries = baseLayer.GetNumberOfSeries( )
      Dim seriesIndex As Integer

      ' Ignore the first series, because that is the new series.
      For seriesIndex = 2 To numberOfSeries
         Dim series As AcChartSeries
         Set series = baseLayer.GetSeries( seriesIndex )
         Set point = series.GetPoint( categoryIndex )

         ' Ignore missing values.
         If Not point.IsMissing( ) Then
            total = total + point.GetYValue( )
            count = count + 1
         End If
      Next seriesIndex

      ' Put the mean value into a new point in the new series.
      Set point = newSeries.InsertPoint( categoryIndex, total /
         count )
   Next categoryIndex
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::AddPoint method
AcChartSeries::InsertEmptyPoint method
AcChartSeries::RemovePoint method
AcChartPoint

AcChartSeries::InsertTrendline method

Call this method to insert a trendline at a specific position within a chart series' list of trendlines. When you insert a new trendline, the original trendline at the insertion point and all trendlines above the insertion point move up one place.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries method
- Code that creates a chart dynamically

Syntax

```
Function InsertTrendline( index As Integer, labelText As String,
    trendlineType As AcChartTrendlineType ) As AcChartTrendline
```

Parameters

index
The position in the chart series' list of trendlines at which the new trendline will be inserted. The first trendline is index 1.
Must be greater than or equal to one. Must be less than or equal to the current number of trendlines for the chart series plus one.

labelText
The text shown in the chart legend for the trendline is drawn.
Null or "" if you do not want the trendline to be listed in the chart's legend.

trendlineType
Defines how the trendline will be fitted to the points in its parent series.

Returns

A handle to the new trendline object.

See also

AcChartSeries::AddTrendline method
AcChartSeries::RemoveTrendline method
AcChartTrendline
AcChartType

AcChartSeries::RemovePoint method

Call the RemovePoint( ) method to remove a point from a chart series. You can call this method only from a chart's CustomizeCategoriesAndSeries( ) method.

To determine the number of points in a chart series, call the series' GetNumberOfPoints( ) method.

Each chart series in a chart layer that has a category scale must have one point for each category in the chart layer. If you remove a point from a chart layer with a category scale, you must replace it with a new point.

If you do not want a point in a chart layer with a category scale to be visible, do not use RemovePoint( ) to remove it. Instead, call the point's ClearValues( ) method to convert it into an empty point.

Syntax

```
Sub RemovePoint( index As Integer )
```

Parameter

index
An index into the series' list of points. The first point is index 1.

See also

AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::AddPoint method
AcChartSeries::GetNumberOfPoints method
AcChartSeries::InsertPoint method
AcChartPoint::ClearValues method

AcChartSeries::RemoveTrendline method

Call this method to remove a trendline at a specific position within a chart series' list of trendlines. When you remove a trendline from a chart series, all the trendlines above that move down one place.

You can call this method only from a chart's CustomizeCategoriesAndSeries method.

Syntax Sub RemoveTrendline(index As Integer)

Parameter **index**
The position in the chart series' list of trendlines from which the trendline will be removed. The first trendline is index 1.

Must be greater than or equal to one. Must be less than or equal to the current number of trendlines for the chart series.

Example In the following example, a chart's CustomizeCategoriesAndSeries( ) method has been overridden to remove the first trendline from the chart's first series when a Boolean parameter's value is True:

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmRemoveTrendline Then
      baseLayer.GetSeries( 1 ).RemoveTrendline( 1 )
   End If
End Sub
```

See also AcChartSeries::AddTrendline method
AcChartSeries::InsertTrendline method
AcChartTrendline

AcChartSeries::SetKeyValue method

Call SetKeyValue( ) to set the unique key value for a chart series. A chart series' initial key value is set when the series is created. This method changes the value.

Changing a series' key value has no effect on the order in which series appear.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetKeyValue(keyValue As Variant)

Parameter **keyValue**
The unique key value for the chart series.

See also AcChart::CustomizeCategoriesAndSeries method

AcChartSeries::GetKeyValue method
AcChartSeries::SetLabelValue method

AcChartSeries::SetLabelValue method

Call the SetLabelValue( ) method to set the label value for a chart series. If the label value is "" or Null, the series will not be listed in the chart's legend.

A chart series' initial label value is set when the series is created. This method changes the value.

Changing a series' label does not affect the order in which series appear.

The label value does not have to be a string. Label values are formatted into text when the chart is viewed, to support locale-specific formatting. For example, if you set labelValue to 1.5, when the chart is viewed in the US English locale the label text is 1.5 but the text is 1,5 when the chart is viewed in the French locale.

You can call this method only from:

- A chart's CustomizeCategoriesAndSeries( ) method
- A chart's Localize( ) method
- Code that is creating a chart dynamically, before you call the chart's ComputeScales( ) method

Syntax Sub SetLabelValue(labelValue As Variant)

Parameter **labelValue**
The label value for the chart series.
Null or "" if you do not want the series to be listed in the chart's legend.

Example The following example overrides a chart's Localize( ) method to translate series labels in the chart's base layer into French at view time if the viewing locale is French:

```
Sub Localize( baseLayer As AcChartLayer, overlayLayer As
   AcChartLayer, studyLayers( ) As AcChartLayer )
   If (GetLocaleName( ) = "fr_FR") Then
      Dim numberOfSeries As Integer
      numberOfSeries = baseLayer.GetNumberOfSeries( )
      Dim seriesIndex As Integer
      For seriesIndex = 1 To numberOfSeries
         Dim series As AcChartSeries
         Set series = baseLayer.GetSeries( seriesIndex )
         Select Case series.GetLabelValue( labelIndex )
         Case "North"
            series.SetLabelValue( "Nord" )
         Case "South"
            series.SetLabelValue( "Sud" )
         Case "East"
            series.SetLabelValue( "Est" )
         Case "West"
            series.SetLabelValue( "Ouest" )
         End Select
      Next seriesIndex
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChart::Localize method
AcChartSeries::SetKeyValue method
AcChartSeries::GetLabelValue method

Class AcChartSeriesStyle

A custom style for a chart series. Figure 7-13 shows the class hierarchy of AcChartSeriesStyle.

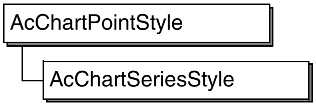


Figure 7-13 AcChartSeriesStyle

Description Use the AcChartSeriesStyle class to represent a custom style for a chart series. Do not create AcChartSeriesStyle objects explicitly from your code. Instead, AcChartLayer objects create AcChartSeriesStyle objects automatically as necessary to build complete charts.

Use AcChartLayer's methods to create and access a chart series' style. You can change the appearance of a chart by calling methods on the chart's series styles.

A series style is used for both the default style for points within the corresponding chart series and the series as a whole. An example of a style setting for individual points is the marker size. An example of a style setting for a series as a whole is the line width.

About default series style

When a new series style is created, the following default settings are copied from its parent chart layer:

- Plot bars as lines
- Border style
- Plot lines between points
- Line width
- Plot markers at points
- Marker size
- Point label format, placement, source, and style

In addition, the series' fill color and line color are set from the standard sequences of fill and line colors.

About custom point styles

By default, a chart point is displayed using its parent series' series style settings. If you want an individual point to have a different appearance from the other points in its parent series, you can add a custom point style.

Example The following example overrides a chart's CustomizeSeriesStyles( ) method to change the appearance of the first series in the chart's scatter chart base layer to:

- Display dotted lines between points. GetLineStyle( ) retrieves the default line style so that only the Pen member of the line style needs to change.
- Display markers at points. The markers are 8 pt red squares filled with yellow.

```
Sub CustomizeSeriesStyles( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )

   ' Get the first series style.
   Dim seriesStyle As AcChartSeriesStyle
   Set seriesStyle = baseLayer.GetSeriesStyle( 1 )

   ' Plot dotted lines between points.
   seriesStyle.SetPlotLinesBetweenPoints( True )
   Dim lineStyle As AcDrawingLineStyle
   lineStyle = seriesStyle.GetLineStyle( )
   lineStyle.Pen = DrawingLinePenDot
   seriesStyle.SetLineStyle( lineStyle )

   ' Draw markers at points.
   seriesStyle.SetPlotMarkersAtPoints( True )
   seriesStyle.SetMarkerFillColor( Yellow )
   seriesStyle.SetMarkerLineColor( Red )
   seriesStyle.SetMarkerShape( ChartMarkerShapeSquare )
   seriesStyle.SetMarkerSize( 8 * OnePoint )
End Sub
```

See also Class AcChart
Class AcChartAxis
Class AcChartCategory
Class AcChartGridLine
Class AcChartLayer
Class AcChartPoint
Class AcChartPointStyle
Class AcChartSeries
Class AcChartTrendline

Methods for Class AcChartSeriesStyle

Methods defined in Class AcChartSeriesStyle

GetLineStyle, GetPointLabelFormat, GetPointLabelSource, PlotBarsAsLines, PlotLinesBetweenPoints, PlotMarkersAtPoints, SetLineStyle, SetPlotBarsAsLines, SetPlotLinesBetweenPoints, SetPlotMarkersAtPoints, SetPointLabelFormat, SetPointLabelSource

Methods inherited from Class AcChartPointStyle

GetBorderStyle, GetFillStyle, GetMarkerFillColor, GetMarkerLineColor, GetMarkerShape, GetMarkerSize, GetPieExplosionAmount, GetPointLabelPlacement, GetPointLabelStyle, SetBackgroundColor, SetBorderStyle, SetFillStyle, SetMarkerFillColor, SetMarkerLineColor, SetMarkerShape, SetMarkerSize, SetPieExplosionAmount, SetPointLabelPlacement, SetPointLabelStyle

AcChartSeriesStyle::GetLineStyle method

Returns the style of lines between points in a chart series. To change the style of lines between points, call this method to retrieve the default settings.

You can call this method only on series styles in chart layers with the following chart types:

- Stacked bar
- Line
- Scatter

Syntax Function GetLineStyle( ) As AcDrawingLineStyle

Returns The style of lines between points in the chart series.

Example For an example of how to use this method, see the example for the AcChartSeriesStyle class.

See also AcChartLayer::GetLineWidth method
AcChartSeriesStyle::PlotLinesBetweenPoints method
AcChartSeriesStyle::SetLineStyle method
Class AcChartSeriesStyle
AcDrawingLineStyle

AcChartSeriesStyle::GetPointLabelFormat method

Returns the format pattern used to format point labels in a chart series.

The format pattern that this method returns might not apply to all the points in a chart series. To retrieve the point label format pattern for an individual point, call the point's GetCustomLabelFormat( ) method.

Syntax Function GetPointLabelFormat( ) As String

Returns The format pattern used to format point labels in the chart series.

See also AcChartLayer::GetPointLabelFormat method
AcChartPoint::GetCustomLabelFormat method
AcChartPoint::HasCustomLabelFormat method
AcChartSeriesStyle::SetPointLabelFormat method

AcChartSeriesStyle::GetPointLabelSource method

Returns the source for point label values in a chart series.

The source that this method returns might not apply to all the points in a chart series. To retrieve the point label value for an individual point, call the point's GetCustomLabelValue( ) method.

Syntax Function GetPointLabelSource( ) As AcChartPointLabelSource

Returns The source for point label values in the chart series.

See also AcChartLayer::GetPointLabelSource method
AcChartPoint::GetCustomLabelValue method
AcChartPoint::HasCustomLabelValue method
AcChartSeriesStyle::SetPointLabelSource method
AcChartPointLabelSource

AcChartSeriesStyle::PlotBarsAsLines method

Determines whether points are plotted as lines in a bar chart series.

You can call this method only on series styles in two-dimensional bar chart layers.

Syntax Function PlotBarsAsLines( ) As Boolean

Returns True if points in the bar chart series are plotted as lines instead of bars.
False if points in the bar chart series are plotted as bars.

See also AcChartLayer::PlotBarsAsLines method
AcChartSeriesStyle::SetPlotBarsAsLines method

AcChartSeriesStyle::PlotLinesBetweenPoints method

Determines whether lines are plotted between points in a chart series.

You can call this method only on series styles in chart layers with the following chart types:

- Stacked bar
- Line
- Scatter

Syntax Function PlotLinesBetweenPoints( ) As Boolean

Returns True if lines are plotted between points in a chart series.
False if lines are not plotted between points in a chart series.

See also AcChartLayer::PlotLinesBetweenPoints method
AcChartSeriesStyle::SetPlotLinesBetweenPoints method

AcChartSeriesStyle::PlotMarkersAtPoints method

Determines whether markers are drawn by default at points in a chart series.

You can call this method only on series styles in chart layers with the following chart types:

- Line
- Scatter
- Stock

The value that this method returns might not apply to all the points in a chart series. To retrieve the marker shape for an individual point, call the corresponding point style's GetMarkerShape( ) method.

Syntax Function PlotMarkersAtPoints( ) As Boolean

Returns True if the default setting for a chart series is that markers will be drawn at points. False if the default setting for a chart series is that markers will not be drawn at points.

See also AcChartLayer::PlotMarkersAtPoints method
AcChartPointStyle::GetMarkerShape method
AcChartSeriesStyle::SetPlotMarkersAtPoints method

AcChartSeriesStyle::SetLineStyle method

Call the SetLineStyle( ) method to set the style of lines between points in a chart series.

You can call this method only on series styles in chart layers with the following chart types:

- Stacked bar
- Line
- Scatter

The recommended place from which to call SetLineStyle( ) is a chart's CustomizeSeriesStyles( ) method.

Syntax Sub SetLineStyle(lineStyle As AcDrawingLineStyle)

Parameter **lineStyle**
The line style used to draw lines between points in the chart series.

Example The following example overrides a chart's CustomizeSeriesStyles( ) method to change the colors of the lines between points in the first five series in the chart's base layer. GetLineStyle( ) retrieves the default line styles so that only the line styles' Color members need to change.

```
Sub CustomizeSeriesStyles( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim seriesStyle As AcChartSeriesStyle
   Dim seriesStyleIndex As Integer
   For seriesStyleIndex = 1 To 5
      Set seriesStyle = baseLayer.GetSeriesStyle(
         seriesStyleIndex )
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = seriesStyle.GetLineStyle( )
      Select Case seriesStyleIndex
      Case 1
         lineStyle.Color = Red
      Case 2
         lineStyle.Color = Green
      Case 3
         lineStyle.Color = Blue
      Case 4
         lineStyle.Color = Magenta
      Case 5
         lineStyle.Color = Cyan
      End Select
      seriesStyle.SetLineStyle( lineStyle )
   Next seriesStyleIndex
End Sub
```

See also AcChart::CustomizeSeriesStyles method
AcChartLayer::SetLineWidth method
AcChartSeriesStyle::GetLineStyle method
AcChartSeriesStyle::SetPlotLinesBetweenPoints method
AcDrawingLineStyle

AcChartSeriesStyle::SetPlotBarsAsLines method

Call the SetPlotBarsAsLines( ) method to specify whether points are plotted as lines in a bar chart series. You can call this method only on series styles in two-dimensional bar chart layers.

The recommended place from which to call SetPlotBarsAsLines( ) is a chart's CustomizeSeriesStyles( ) method.

Syntax Sub SetPlotBarsAsLines(plotBarsAsLines As Boolean)

Parameter **plotBarsAsLines**
True causes points in the bar chart series to be plotted as lines instead of bars. False causes points in the bar chart series to be plotted as bars.

See also AcChart::CustomizeSeriesStyles method
AcChartLayer::SetPlotBarsAsLines method
AcChartSeriesStyle::PlotBarsAsLines method

AcChartSeriesStyle::SetPlotLinesBetweenPoints method

Call the SetPlotLinesBetweenPoints( ) method to specify whether lines are plotted between points in a chart series.

You can call this method only on series styles in chart layers with the following chart types:

- Stacked bar
- Line
- Scatter

The recommended place from which to call SetPlotLinesBetweenPoints( ) is a chart's CustomizeSeriesStyles( ) method.

Syntax Sub SetPlotLinesBetweenPoints(plotLinesBetweenPoints As Boolean)

Parameter **plotLinesBetweenPoints**
True causes lines to be drawn between the points in the chart series.
False causes lines not to be drawn between the points in the chart series.

Example For an example of how to use this method, see the example for the AcChartSeriesStyle class.

See also AcChart::CustomizeSeriesStyles method
AcChartLayer::SetPlotLinesBetweenPoints method
AcChartSeriesStyle::PlotLinesBetweenPoints method

AcChartSeriesStyle::SetPlotMarkersAtPoints method

Call the SetPlotMarkersAtPoints( ) method to specify whether markers are drawn by default at points in a chart series.

You can call this method only on layers with the following chart types:

- Line
- Scatter
- Stock

The value that this method sets might not apply to all the points in a chart series. To set the marker shape for an individual point, call the corresponding point style's SetMarkerShape( ) method.

The recommended place from which to call SetPlotMarkersAtPoints( ) is a chart's CustomizeSeriesStyles( ) method.

Syntax Sub SetPlotMarkersAtPoints(plotMarkersAtPoints As Boolean)

Parameter **plotMarkersAtPoints**
True causes markers to be drawn at points in the chart layer. False causes markers not to be drawn at points in the chart layer.

Example For an example of how to use this method, see the example for the AcChartSeriesStyle class.

See also AcChart::CustomizeSeriesStyles method
AcChartLayer::SetPlotMarkersAtPoints method
AcChartPointStyle::SetMarkerShape method
AcChartSeriesStyle::PlotMarkersAtPoints method

AcChartSeriesStyle::SetPointLabelFormat method

Call the SetPointLabelFormat( ) method to set the format pattern used to format point labels in a chart series.

The format pattern is ignored for string label values.

The format pattern that this method sets might not apply to all the points in a chart series. To set the point label format pattern for an individual point, call the point's SetCustomLabelFormat( ) method.

The recommended place from which to call SetPointLabelFormat( ) is a chart's CustomizeSeriesStyles( ) method.

Syntax Sub SetPointLabelFormat(pointLabelFormat As String)

Parameter **pointLabelFormat**
The format pattern used to format point labels in the chart series.

Example For an example of how to use this method, see the example for the AcChartSeriesStyle::SetPointLabelSource method.

See also AcChart::CustomizeSeriesStyles method
AcChartSeriesStyle::SetPointLabelSource method
AcChartLayer::SetPointLabelFormat method
AcChartPoint::SetCustomLabelFormat method
AcChartSeriesStyle::GetPointLabelFormat method

AcChartSeriesStyle::SetPointLabelSource method

Call the SetPointLabelSource( ) method to set the source for point label values in a chart series. The source that this method sets does not necessarily apply to all the points in a chart series. To set the point label value for an individual point, call the point's SetCustomLabelValue( ) method.

The recommended place from which to call SetPointLabelSource( ) is a chart's CustomizeSeriesStyles( ) method.

Syntax Sub SetPointLabelSource(pointLabelSource As AcChartPointLabelSource)

Parameter **pointLabelSource**
The default source for point label values in the chart layer.

Example The following example overrides a chart's CustomizeSeriesStyles( ) method to add point labels to the first series in the chart's base layer:

```
Sub CustomizeSeriesStyles( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim seriesStyle As AcChartSeriesStyle
   Set seriesStyle = baseLayer.GetSeriesStyle( 1 )
   seriesStyle.SetPointLabelSource(
      ChartPointLabelSourceYValue )
   seriesStyle.SetPointLabelFormat( "#,##0.00" )
End Sub
```

See also AcChartPointLabelSource
AcChartLayer::SetPointLabelSource method
AcChartPoint::SetCustomLabelValue method
AcChartSeriesStyle::GetPointLabelSource method

Class AcChartTrendline

A trendline in a chart. Figure 7-14 shows the class hierarchy of AcChartTrendline.

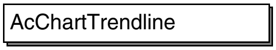


Figure 7-14 AcChartTrendline

Description AcChartTrendline represents a trendline in a chart. A trendline shows the trend for a single series in a chart.

In most cases, you can define trendlines using Advanced Chart Options. If you define trendlines in this way, AcChartTrendline objects are created automatically. To access an AcChartTrendline object that has been created automatically, use the AcChartSeries::GetTrendline( ) method.

If you need to create an AcChartTrendline object in code, you cannot use the New keyword, or the NewInstance( ) or NewPersistentInstance( ) functions. Instead, use the AcChartSeries::AddTrendline( ) or AcChartSeries::InsertTrendline( ) methods.

To change the appearance of a trendline, call methods on the corresponding AcChartTrendline object.

Example In the following example, a chart's CustomizeCategoriesAndSeries( ) method has been overridden to add a trendline to the chart's first series when a Boolean parameter's value is True:

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmShowTrendline Then
      Dim series As AcChartSeries
      Set series = baseLayer.GetSeries( 1 )
      Dim trendline As AcChartTrendline
      Set trendline = series.AddTrendline( "Trend",
+        ChartTrendlineTypePolynomial )
      trendline.SetOrder( 3 )
      Dim lineStyle As AcDrawingLineStyle
      lineStyle = trendline.GetLineStyle( )
      lineStyle.Color = Red
      lineStyle.Width = 2 * OnePoint
      trendline.SetLineStyle( lineStyle )
   End If
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::AddTrendline method
AcChartSeries::InsertTrendline method
Class AcChart

Class AcChartAxis
Class AcChartCategory
Class AcChartGridLine
Class AcChartLayer
Class AcChartPoint
Class AcChartPointStyle
Class AcChartSeries
Class AcChartSeriesStyle

Methods for Class AcChartTrendline

Methods defined in Class AcChartTrendline

ClearIntercept, GetEndYValue, GetIndex, GetIntercept, GetLabelText, GetLineStyle, GetMaximumYValue, GetMinimumYValue, GetOrder, GetPeriod, GetStartYValue, GetTrendlineType, HasIntercept, SetIntercept, SetLabelText, SetLineStyle, SetOrder, SetPeriod, SetTrendlineType

AcChartTrendline::ClearIntercept method

Clears the intercept value for a trendline. You can call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically

Syntax Sub ClearIntercept( )

See also AcChart::CustomizeCategoriesAndSeries method
AcChartTrendline::GetIntercept method
AcChartTrendline::HasIntercept method
AcChartTrendline::SetIntercept method

AcChartTrendline::GetEndYValue method

Returns the y value of the end of a trendline.

You can call this method only from a chart's AdjustChart( ) method.

Syntax Function GetEndYValue( ) As Variant

Returns The y value of the end of the trendline.

Example For an example of how to use this method, see the example for the AcChartTrendline::SetLineStyle method.

See also AcChart::AdjustChart method
AcChartTrendline::GetMaximumYValue method
AcChartTrendline::GetMinimumYValue method

AcChartTrendline::GetStartYValue method
AcChartTrendline::SetLineStyle method

AcChartTrendline::GetIndex method

Returns the index of a trendline within its parent chart series' list of trendlines.

Syntax Function GetIndex( ) As Integer

Returns The index of the trendline within its parent chart series' list of trendlines. The first trendline for a series is index 1.

AcChartTrendline::GetIntercept method

Returns the intercept value for a trendline.

Syntax Function GetIntercept( ) As Variant

Returns The intercept value for the trendline.
Null if the trendline has no intercept value.

See also AcChartTrendline::ClearIntercept method
AcChartTrendline::HasIntercept method
AcChartTrendline::SetIntercept method

AcChartTrendline::GetLabelText method

Returns the label text for a trendline. The label text appears in the chart legend.

Syntax Function GetLabelText( ) As String

Returns The label text for the trendline.

See also AcChartTrendline::SetLabelText method

AcChartTrendline::GetLineStyle method

Returns the line style used to draw a trendline. Call this method to retrieve the default settings before changing a trendline's line style.

Syntax Function GetLineStyle( ) As AcDrawingLineStyle

Returns The line style used to draw the trendline.

Example For an example of how to use this method, see the example for the AcChartTrendline::SetLineStyle method.

See also AcChartTrendline::SetLineStyle method
AcDrawingLineStyle

AcChartTrendline::GetMaximumYValue method

Returns the maximum y value of a trendline. You can call this method only from a chart's AdjustChart( ) method.

Syntax Function GetMaximumYValue( ) As Variant

Returns The maximum y value of the trendline.

See also AcChart::AdjustChart method
AcChartTrendline::GetEndYValue method
AcChartTrendline::GetMinimumYValue method
AcChartTrendline::GetStartYValue method

AcChartTrendline::GetMinimumYValue method

Returns the minimum y value of a trendline.

You can call this method only from a chart's AdjustChart( ) method.

Syntax Function GetMinimumYValue( ) As Variant

Returns The minimum y value of the trendline.

See also AcChart::AdjustChart method
AcChartTrendline::GetEndYValue method
AcChartTrendline::GetMaximumYValue method
AcChartTrendline::GetStartYValue method

AcChartTrendline::GetOrder method

Returns the order of a polynomial trendline.

Syntax Function GetOrder( ) As Integer

Returns The order of the trendline.
Null if the trendline is not a polynomial trendline.

See also AcChartTrendline::SetOrder method

AcChartTrendline::GetPeriod method

Returns the period of a moving average trendline.

Syntax Function GetPeriod( ) As Integer

Returns The period of the trendline.
Null if the trendline is not a moving average trendline.

See also AcChartTrendline::SetPeriod method

AcChartTrendline::GetStartYValue method

Returns the y value of the start of a trendline.

You can call this method only from a chart's AdjustChart( ) method.

Syntax Function GetStartYValue( ) As Variant

Returns The y value of the start of the trendline.

Example For an example of how to use this method, see the example for the AcChartTrendline::SetLineStyle method.

See also AcChart::AdjustChart method
AcChartTrendline::GetEndYValue method
AcChartTrendline::GetMaximumYValue method
AcChartTrendline::GetMinimumYValue method
AcChartTrendline::SetLineStyle method

AcChartTrendline::GetTrendlineType method

Returns a value that indicates how a trendline is fitted to the points in its parent series.

Syntax Function GetTrendlineType( ) As AcChartTrendlineType

Returns A value that indicates how the trendline is fitted to the points in its parent series.

See also AcChartTrendline::SetTrendlineType method
AcChartType

AcChartTrendline::HasIntercept method

Determines whether a trendline has an intercept value.

Syntax Function HasIntercept( ) As Boolean

Returns True if the trendline has an intercept value.
False if the trendline does not have an intercept value.

See also AcChartTrendline::ClearIntercept method
AcChartTrendline::GetIntercept method
AcChartTrendline::SetIntercept method

AcChartTrendline::SetIntercept method

Sets the intercept value for a trendline. You can call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically

You can call this method only when the trendline is one of the following types:

- Linear
- Polynomial
- Exponential

Syntax Function SetIntercept(intercept As Variant)

Parameter **intercept**
The intercept value for the trendline.

Example In the following example, a chart's CustomizeCategoriesAndSeries( ) method has been overridden to remove the first category from a chart and use the y value for that category as the intercept for a trendline:

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   Dim trendline As AcChartTrendline
   Set trendline = series.GetTrendline( 1 )
   Dim point As AcChartPoint
   Set point = series.GetPoint( 1 )
   trendline.SetIntercept( point.GetYValue )
   baseLayer.RemoveCategory( 1 )
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartTrendline::ClearIntercept method
AcChartTrendline::GetIntercept method
AcChartTrendline::HasIntercept method

AcChartTrendline::SetLabelText method

Sets the label text for a trendline. The label text appears in the chart legend. If the label text is "" or Null, the trendline will not be listed in the legend.

You can call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically

Syntax Function SetlLabelText(labelText As String)

Parameter **labelText**
Text that will be shown in the chart legend. Null or "" if you do not want the trendline to be listed in the chart's legend.

Example In the following example, a chart's CustomizeCategoriesAndSeries( ) method has been overridden to suppress the legend entry for a trendline when a Boolean parameter's value is True:

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   If parmRemoveTrendlineFromLegend Then
      Dim series As AcChartSeries
      Set series = baseLayer.GetSeries( 1 )
      Dim trendline As AcChartTrendline
      Set trendline = series.GetTrendline( 1 )
      trendline.SetLabelText( "" )
   End If
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::AddTrendline method
AcChartSeries::InsertTrendline method
AcChartTrendline::GetLabelText method

AcChartTrendline::SetLineStyle method

Sets the line style used to draw a trendline. You can call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method
- A chart's AdjustChart( ) method
- Code that is creating a chart dynamically

Syntax Function SetlLineStyle(lineStyle As AcDrawingLineStyle)

Parameter **lineStyle**
The line style used to draw the trendline.

Example In the following example, a chart's AdjustChart( ) method has been overridden to change the color of a trendline depending on whether the trend is upwards or downwards. GetLineStyle( ) is used to retrieve the trendline's original line style settings so that only the line style's Color member needs to be set.

```
Sub AdjustChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   Dim trendline As AcChartTrendline
   Set trendline = series.GetTrendline( 1 )
   Dim startYValue As Variant
   startYValue = trendline.GetStartYValue( )
   Dim endYValue As Variant
   endYValue = trendline.GetEndYValue( )
   Dim lineStyle As AcDrawingLineStyle
   lineStyle = trendline.GetLineStyle( )
   if (endYValue > startYValue) Then
      lineStyle.Color = Green
   ElseIf (endYValue < startYValue) Then
      lineStyle.Color = Red
   End If
   trendline.SetLineStyle( lineStyle )
End Sub
```

See also AcChart::AdjustChart method
AcChart::CustomizeCategoriesAndSeries method
AcChartTrendline::GetLineStyle method
AcDrawingLineStyle

AcChartTrendline::SetOrder method

Sets the order of a polynomial trendline.

You can call this method only for a polynomial trendline.

You can call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically

Syntax Function SetOrder(order As Integer)

Parameter **order**
The order of the polynomial trendline. Must be in the range 2 through 6.

Example In the following example, a chart's CustomizeCategoriesAndSeries( ) method has been overridden to set the order of a trendline to 2 less than the number of points in the trendline's parent series:

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   Dim trendline As AcChartTrendline
   Set trendline = series.GetTrendline( 1 )
   Dim order As Integer
   order = series.GetNumberOfPoints( ) - 2
   If (order < 2) Then
      order = 2
   ElseIf (order > 6) Then
      order = 6
   End If
   trendline.SetOrder( order )
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartTrendline::GetOrder method

AcChartTrendline::SetPeriod method

Sets the period of a moving average trendline. You can call this method only for a moving average trendline.

You can call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically

Syntax Function SetPeriod(period As Integer)

Parameter **period**
The number of values to be averaged to calculate each point in the trendline. Must be at least 2.

Example In the following example, a chart's CustomizeCategoriesAndSeries( ) method has been overridden to set the period of a trendline to the value of an integer parameter:

```
Sub CustomizeCategoriesAndSeries( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   Dim trendline As AcChartTrendline
   Set trendline = series.GetTrendline( 1 )
   trendline.SetPeriod( parmTrendlinePeriod )
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartTrendline::GetPeriod method

AcChartTrendline::SetTrendlineType method

Defines how a trendline will be fitted to the points in its parent series.

You can call this method from:

- A chart's CustomizeCategoriesAndSeries( ) method
- Code that is creating a chart dynamically

Syntax Function SetTrendlineType(trendlineType As AcChartTrendlineType)

Parameter **trendlineType**
Defines how the trendline will be fitted to the points in its parent series.

Example In the following example, a chart's CustomizeCategoriesAndSeries( ) method has been overridden to change the way in which a trendline is fitted to the points in its parent series based on the value of a Boolean parameter:

```
Sub CustomizeCategoriesAndSeries(baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
   Dim series As AcChartSeries
   Set series = baseLayer.GetSeries( 1 )
   Dim trendline As AcChartTrendline
   Set trendline = series.GetTrendline( 1 )
   If (parmMakeTrendlinePolynomial) Then
      trendline.SetTrendlineType( ChartTrendlineTypePolynomial )
      trendline.SetOrder( 3 )
   End If
End Sub
```

See also AcChart::CustomizeCategoriesAndSeries method
AcChartSeries::AddTrendline method
AcChartSeries::InsertTrendline method
AcChartTrendline::GetTrendlineType method
AcChartType

Class AcCollection

The abstract base class for the Actuate collection classes. Figure 7-15 shows the class hierarchy of AcCollection.

Figure 7-15 AcCollection

Description A collection holds objects of any type. The methods on a collection pass references to these objects as a special type called AnyClass. You must assign the object reference to a handle of your derived class to access the methods on your object.

You must take care to store the correct kind of objects in each of your collections. For example, if you create a collection of controls, you can also store a data adapter in that collection. If your code expects all the objects in the collection to be controls, however, a run-time error occurs when you try to assign the data adapter to an object reference variable for a control.

Subclassing AcCollection

Because AcCollection is an abstract base class, do not derive directly from it.

See also Class AcIterator
Class AcOrderedCollection

Methods for Class AcCollection

Methods defined in Class AcCollection

Compare, Contains, Copy, FindByValue, GetCount, IsEmpty, NewIterator, RemoveAll, Remove

AcCollection::Compare method

Compares two objects in a collection. A report can subclass the collection and override this method to provide the logic for making the comparison. FindByValue( ) calls Compare( ) to determine whether an object with a matching value exists in the collection.

Syntax Function Compare(obj1 As AnyClass, obj2 As AnyClass) As Variant

Parameters **obj1**
The first object to compare.

obj2
The second object to compare.

Returns 0 if obj1 is the same as obj2.
1 if obj1 is greater than obj2.
-1 if obj2 is greater than obj1.

AcCollection::Contains method

Tests whether an object exists in the collection.

Syntax Function Contains(item as AnyClass) As Boolean

Parameter **item**
The object to test.

Returns True if the object is in the collection.
False if the object is not in the collection.

AcCollection::Copy method

Copies the contents of another collection into the current collection.

Syntax Sub Copy(from As AcCollection)

Parameter **from**
The collection from which to copy the contents.

AcCollection::FindByValue method

Finds an object that has the same value as a specified object.

Syntax Function FindByValue(obj As AnyClass) As AnyClass

Parameter **obj**
The object to find.

AcCollection::GetCount method

Returns the number of objects in the collection. To determine if a collection contains objects, use IsEmpty( ).

Syntax Function GetCount( ) As Integer

Returns The number of objects in the collection.

See also AcCollection::IsEmpty method

AcCollection::IsEmpty method

Determines whether the collection is empty. To determine the number of objects in a collection, use GetCount( ).

Syntax Function IsEmpty( ) As Boolean

Returns True if the collection contains no objects.
False if the collection contains at least one object.

See also AcCollection::GetCount method

AcCollection::NewIterator method

Creates an iterator for the collection. The iterator supports accessing each item in the collection. If the collection is ordered, the iterator accesses each item in the order you established previously.

Syntax Function NewIterator( ) As AcIterator

Returns A reference to the iterator.

See also Class AcIterator

AcCollection::Remove method

Removes a specified item from the collection. If you specify an object that is not in the collection, Remove( ) does nothing. To remove all objects in a collection, use RemoveAll( ).

The framework automatically deletes objects if they are transient and their reference count goes to zero.

Syntax Sub Remove(item As AnyClass)

Parameter **item**
The object in the collection to delete.

Example The following example shows how to override a page's OnRow( ) method to remove a control based on the name of the class:

```
Sub OnRow( row As AcDataRow )
   Super::OnRow( row )
   Dim control As AcControl
   Set control = FindContentByClass( "XXX" )
   control.DetachFromContainer( )
   control.Abandon( )
End Sub
```

See also AcCollection::RemoveAll method

AcCollection::RemoveAll method

Removes all contents from the collection. To remove a single object from a collection, use Remove( ).

The framework automatically deletes objects if they are transient and their reference count goes to zero.

Syntax Sub RemoveAll( )

See also AcCollection::Remove method

Class AcComponent

The principal base class for the Actuate Foundation Classes (AFC). Figure 7-16 shows the class hierarchy of AcComponent.

AcComponent

Figure 7-16 AcCollection

Description AcComponent is the root class for the AFC framework. All classes that appear in Report Structure derive from AcComponent. AcComponent defines the mechanism for creating objects within container objects.

Within the AFC framework, most classes are derived from AcComponent. However, several classes are not derived from AcComponent because they are not part of the report structure, rather they are classes that provide services to the report. Such classes include AcDBCursor, AcDBStatement, and AcIterator.

Subclassing AcComponent

Because AcComponent is the foundation for the AFC framework, do not derive directly from it.

Property

Table 7-12 describes the AcComponent property.

Table 7-12 AcComponent properties

Property	Type	Description
DisplayName	String	Not used by e.Report Designer Professional

See also Class AcConnection
Class AcReport
Class AcReportComponent

Methods for Class AcComponent

Methods defined in Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcComponent::ApplyVisitor method

Starts visitor functions for a component. Visitor functions revisit components in a report to do special processing, such as creating output in a text file or spreadsheet.

Syntax Sub ApplyVisitor(visitor As AcVisitor)

Parameter **visitor**
The visitor component that contains the visit methods to call.

Example The following example shows part of the programming required to produce an Excel spreadsheet from information that is contained in components. For a description of the entire example, see AcVisitor.

The part of the example below shows how to call ApplyVisitor. The Visitor should be invoked during report generation, in the report root component's Finish( ) method, after Super::Finish( ). The class, AcDetailCsvVisitor, contains the visitor methods that are used to create the Excel spreadsheet.

```
Sub Finish( )
   Dim visitor As AcDetailCsvVisitor
   Super::Finish( )
   Set visitor = New AcDetailCsvVisitor
   visitor.FileName = "c:\temp\extract.csv"
   ApplyVisitor( visitor )
   Shell( "d:\Program Files\Microsoft Office\Office\Excel.exe
      c:\Temp\Extract.csv", 1 )
End Sub
```

See also Class AcVisitor

AcComponent::Delete method

The Actuate Basic destructor. In derived classes, Actuate Basic calls Delete( ) when deleting transient objects. The destructor for the most derived class is called first, followed by all the destructors in the ancestor classes in order of their position in the class hierarchy.

In derived classes, you can override Delete( ) to perform cleanup tasks such as closing files. To do this, call Super::Delete( ) as the last line of your method.

Syntax Sub Delete( )

See also AcComponent::New method

AcComponent::IsPersistent method

Indicates whether the component is persistent or transient. A persistent component is stored in the report object instance (.roi) file. A transient component deletes from memory when the report completes.

Syntax Function IsPersistent( ) As Boolean

Returns True if the component is persistent.
False if the component is transient.

AcComponent::New method

Constructor. Initializes all the properties set using the Properties page.

In derived classes, called by Actuate Basic after an object is instantiated. In derived classes, you typically override New( ) to initialize variables you define. If you override New( ), you must call Super::New( ) as the first line of your method so that SetProperties( ) can initialize property values.

Syntax Sub New( )

Class AcConditionalSection

A class that you use in the report design to instantiate a component in a section, depending on a specified condition. Figure 7-17 shows the class hierarchy of AcConditionalSection.

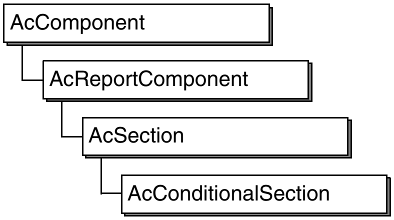


Figure 7-17 AcConditionalSection

Description Use a conditional section to instantiate one of several components conditionally in a section. For example, you can use a conditional section to print a different frame for credit card or cash transactions.

The three general steps to set up a conditional section are:

- Write a Boolean expression in the conditional section's IfExp property.
- Place a component in the conditional section's Then slot.
- Place a component in the conditional section's Else slot.

If the IfExp expression evaluates to True, the conditional section builds the component in the section's Then slot. Otherwise, the conditional section builds the component in the Else slot, as shown in Figure 7-18.

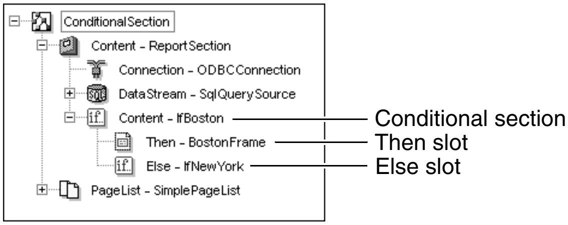

ConditionalSection
Content - ReportSection
Connection - ODBCConnection
DataStream - SqlQuerySource
Content - IfBoston
Then - BostonFrame
Else - IfNewYork
PageList - SimplePageList

Figure 7-18 A conditional section

You can create a case-statement type of structure, in which your report chooses from one of a group of possible components to generate. Perform the following steps to create a case-statement structure:

- Add a sequential section to your report to represent the entire set of possible components.
- Add conditional sections to the sequential section where each conditional section represents one case.

- For each conditional section, specify a Boolean IfExp value to describe the case.
- For each conditional section, place a component in the Then slot. This component instantiates when the IfExp value is True.

Using this structure, you can choose your If expressions so that the cases do not overlap. You can also overlap the cases to instantiate multiple components depending on the condition.

Properties

Table 7-13 lists AcConditionalSection properties.

Table 7-13 AcConditionalSection properties

Property	Type	Description
IfExp	Expression	A Boolean expression. If the expression evaluates to True, the section builds the component in the Then slot. If the expression is False, the conditional section builds the component in the Else slot. The default value for the expression is True.
Then	AcReportComponent slot	Builds this component if IfExp is True. If you enter nothing in this slot and the condition is True, e.Report Designer Professional does not build content for the conditional section.
Else	AcReportComponent slot	Builds this component if IfExp is False. If you enter nothing in this slot and the condition is False, e.Report Designer Professional does not build content for the conditional section.

Methods for Class AcConditionalSection

Method defined in Class AcConditionalSection

ConditionIsTrue

Method inherited from Class AcComponent

ApplyVisitor

AcConditionalSection::ConditionIsTrue method

Returns True if the conditional section should instantiate a component in the Then slot of a report. Returns False if the conditional section should instantiate a component in the Else slot.

You can consider the condition as follows:

```
If ConditionIsTrue( row ) Then
   instantiate Then component
Else
   instantiate Else component
End If
```

The AFC framework generates this method based on the value you enter in the conditional section's IfExp property. Override this method when the expression you want to evaluate is more complex than IfExp can handle.

Syntax Function ConditionIsTrue(row As AcDataRow) As Boolean

Parameter **row**
The data row, if any, associated with this component. This is the data row passed to the BuildFromRow( ) method of the conditional section. The row returns Nothing if the container component calls Build( ) instead.

Returns True if the conditional section should instantiate the component in the Then slot. False if the conditional section should instantiate the component in the Else slot.

Class AcConnection

An abstract base class that defines the core protocol for all connection components. Figure 7-19 shows the class hierarchy of AcConnection.

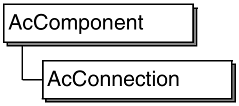


Figure 7-19 AcConnection

Description AcConnection defines the core protocol for connecting to and disconnecting from an input source. Concrete AFC connection classes derived from AcConnection redefine the methods to perform tasks specific to a database connection.

The IsOpen variable contains the current state of the connection. If you derive your own connection classes, keep IsOpen up to date.

Variable

Table 7-14 describes the AcConnection variable.

Table 7-14 AcConnection variable

Variable	Type	Description
IsOpen	Boolean	True establishes a valid connection.

Methods for Class AcConnection

Connect, Disconnect, IsConnected, RaiseError

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcConnection::Connect method

Attempts to connect to the database. You must have previously set the variables needed to describe the connection. Derived classes override this method to establish the connection to the database.

Syntax Function Connect( ) As Boolean

Returns True if the connection is established.
False if the connection is not established.

See also AcConnection::Disconnect method
AcConnection::IsConnected method

AcConnection::Disconnect method

Disconnects from the database. Derived classes override this method to perform the actual disconnect.

Syntax Sub Disconnect( )

See also AcConnection::Connect method
AcConnection::IsConnected method

AcConnection::IsConnected method

Determines whether the connection is established. Use IsConnected( ) in a control structure to execute tasks depending on whether a connection is opened or closed.

Syntax Function IsConnected( ) As Boolean

Returns True if the connection is established.
False if the connection is not established.

See also AcConnection::Connect method
AcConnection::Disconnect method

AcConnection::RaiseError method

An abstract method that derived classes override to obtain error information from the database connection, then raise the error.

Relational databases usually report error conditions using their connection interfaces.

Syntax Sub RaiseError( )

Class AcControl

An abstract base class that defines the core characteristics of all controls. Figure 7-20 shows the class hierarchy of AcControl.

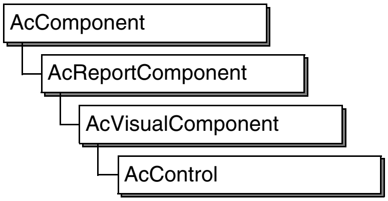


Figure 7-20 AcControl

Description A control is the primary visual component in a report. You typically place controls in a frame to display labels, data, images, drawing elements, and so on. You can also place controls directly on a page, to display the page number or date, for example.

AcControl is the base class for:

- Constant controls, such as labels, lines, and rectangles. Constant controls are fully defined at design time and need no additional data in the Factory.

  You can, however, write code to change the attributes of a constant control in the Factory. For example, you can adjust the size of a line based on the dollar amount of a data row field, or change the color of a label to red to show that a particular customer is 60 days past due.

  To accomplish these tasks, you can override the container frame's OnRow( ) method or the control's BuildFromRow( ) method to add the code.
- Data controls, such as text, integer, and date and time controls, that display values from data rows processed in the Factory. For more information about data controls, see the description of the AcDataControl class.

Subclassing AcControl

Typically, you derive a new control from one of the more specialized subclasses of AcControl, such as AcDoubleControl or AcCrosstabControl. Do not derive directly from AcControl.

Class protocol

AcControl's protocol defines the tasks that all controls perform. Table 7-15 lists the protocol methods for AcControl.

Table 7-15 Class protocol for AcControl

Method	Task
New( )	Initializes the control.
Start( )	Initializes the control.
Build( ) or	Called if the enclosing frame does not have access to a data row.
BuildFromRow( )	Called if the enclosing frame has access to a data row. Sets the value of the control from the data row.
Finish( )	Completes the control.

Property

Table 7-16 describes the AcControl property.

Table 7-16 AcControl property

Property	Type	Description
BalloonHelp	String	The text to display when the user holds the cursor over a control

Methods for Class AcControl

Methods defined in Class AcControl

BalloonHelp, GetControlValue, GetText, GetXMLText, GetValue, IsSummary, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcControl::BalloonHelp method

Returns the text to be displayed when the user hovers the mouse pointer over a control. The default behavior is to display the value of the BalloonHelp property. Override BalloonHelp( ) to display any other fixed or calculated value. You can specify the text programmatically, or display the formatted value of a control. BalloonHelp( ) is available for all subclasses of AcControl except AcLineControl.

Syntax Function BalloonHelp( ) As String

Returns The string to display.

See also AcControl::GetText method

AcControl::GetControlValue method

Returns the value of another control within the same frame. If, for example, you want to change the value or property of one control depending on the value of another control, you can call GetControlValue( ) from the first control to get the value of the second control.

GetControlValue( ) first finds the control, then calls that control's GetValue( ) method to obtain the value. If you call GetControlValue( ) to get the value of a control before its value is set, GetControlValue( ) returns Null.

If your code calls GetControlValue( ), you must be aware of the order in which controls are built. Generally, the controls of a frame are built in the same order that they appear in Report Structure.

Syntax Function GetControlValue(controlName As String) As Variant

Parameter **controlName**
The name of the control for which you want the value. Specify the control using its fully qualified name, such as OrdersReport::ItemFrame::PriceControl, or just the control's name, such as PriceControl.

Returns The value of the specified control.
Null if the specified control's value is not set.

Example In the following example, a label control's Finish( ) method is overridden so that the label's font color is set to red when the value of an integer control, DaysOverdue, is greater than 30. GetControlValue( ) returns the value of the DaysOverdue integer control. Both the label and integer controls are in the same frame.

```
Sub Finish( )
   Super::Finish( )
   If (GetControlValue("DaysOverdue") > 30) Then
      Font.Color = Red
   End If
End Sub
```

Note that you can use Conditional Formatting to achieve the same effect as this example without writing code.

See also AcBaseFrame::GetControlValue method

AcControl::GetXMLText method

Returns the value of a control that has the XMLType property set to XMLText.

Syntax Function GetXMLText( ) As String

Returns The control value as a string.

AcControl::GetText method

Formats a control's value for display. The framework calls GetText( ) to format the data value of data controls and the text value of label controls. In derived classes, such as AcTextControl or AcIntegerControl, GetText( ) formats a control's data value by calling the Actuate Basic Format$ function and using the format pattern you specified in the control's Format property.

You can override a control's GetText( ) method to perform additional formatting, such as translating a numeric value into a string of words for printing on a check.

Use GetText( ) to alter the appearance of a control's value at view time. You cannot modify the actual value of the control by changing the value of its DataValue variable. Overriding DataValue at view time is not a supported operation and can cause unpredictable behavior.

Syntax Function GetText( ) As String

Returns The text to display in the control.

See also AcDataControl::Format method

AcControl::GetValue method

Returns the value of a data control.

Syntax Function GetValue( ) As Variant

Returns The value of the DataValue variable if the control is a data control.
Null if the control is not a data control.

See also AcDataRow::GetValue method

AcControl::IsSummary method

Use the IsSummary( ) method to determine whether the control processes a single row or multiple rows. If IsSummary( ) returns True, the control processes multiple rows.

e.Report Designer Professional generates IsSummary( ) based on the expressions you enter in the value expression properties for a control. If any of those expressions contains an aggregate function, IsSummary( ) returns True.

Syntax Function IsSummary( ) As Boolean

Returns True if the control processes multiple rows.
False otherwise.

AcControl::PageNo method

Returns the position of the page in the report, starting from one. For example, to show the page number in a control, set the value of the control to PageNo( ).

You can use this method for a control directly on a page or for a control in a frame in a PageHeader or PageFooter slot. It is not applicable to other frames because the page is not known when the value of the control is set.

PageNo( ) raises an error if it is called before the frame holding this control is added to a page.

Syntax Function PageNo( ) As Integer

Returns An integer indicating the page index of this control.

See also AcControl::PageNo$ method
AcVisualComponent::GetPageContainer method

AcControl::PageNo$ method

Returns the formatted page number of the control as a string. For example, to show the formatted page number such as vi, 107, or 12-5 in a control, set the value of the control to PageNo$.

You can use this method for a control directly on a page, or for a control in a frame in a PageHeader or PageFooter slot. It is not applicable to other frames because the page is not known when the value of the control is set.

PageNo$( ) raises an error if it is called before the frame holding this control is added to a page.

Syntax Function PageNo$( ) As String

Returns A string containing the formatted page number of this control.

See also AcControl::PageNo method
AcVisualComponent::GetPageContainer method

AcControl::SetDataValue method

Sets the value for a data control within the same frame. SetDataValue( ) simplifies coding by providing a way to assign values to controls that work with AcDataControl or any subclass of AcDataControl.

Syntax Sub SetDataValue(newValue As Variant)

Parameter **newValue**
The value of the control.

Example The following example shows how SetDataValue( ) simplifies the coding required to set the value of a control. The first code snippet shows how to set the value of a control in Actuate Basic without using the SetDataValue( ) method.

```
Dim control As AcControl
Dim textControl As AcTextControl
Dim intControl As AcIntegerControl

Set control = GetControl("Foo")
Set textControl = control
textControl.DataValue = "Text Value"
Set control = GetControl("Bar")
Set intControl = control
intControl.DataValue = 10
```

The following code shows how to use SetDataValue( ) to perform the same task:

```
GetControl("Foo").SetDataValue("Text Value")
GetControl("Bar").SetDataValue( 10 )
```

Using SetDataValue( ) removes the requirement to refer to specific subclasses.

See also AcBaseFrame::GetControl method
AcControl::SetDataValue method

Class AcCrosstab

Displays data in rows and columns. Figure 7-21 shows the class hierarchy of AcCrosstab.

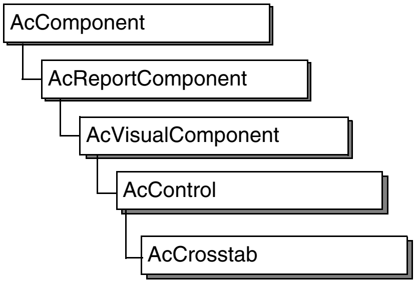


Figure 7-21 AcCrosstab

Description Use the AcCrosstab class to display data in a spreadsheet format. For example, you can use AcCrosstab to display calculated data values and totals of those values. The contents of the cross tab determine its size. A cross tab resizes dynamically both horizontally and vertically to fit its contents.

Variables

Table 7-17 lists AcCrosstab variables.

Table 7-17 AcCrosstab variables

Variable	Type	Description
Background Color	AcColor	The background color of the cross tab, row, column, or cell. This property does not affect subgroups of rows and columns. The background color of the row overwrites the background color of the column because rows are added on top of columns. The default value is Transparent.
Column Headings Border	AcCrosstab BorderStyle	Specifies the border color and thickness of a column heading. The default color is Black. The default thickness is 1 point.
OuterBorder	AcCrosstab BorderStyle	Specifies the color and thickness of the outside border of a cross tab, row, or column. The default color is Black. The default thickness is 1 point for a cross tab, 0.5 point for a row or column.

Table 7-17 AcCrosstab variables

Variable	Type	Description
RowHeadings Border	AcCrosstab BorderStyle	Specifies the border color and thickness of row headings of a cross tab. The default color is Black. The default thickness is 1 point.

Properties

Table 7-18 lists AcCrosstab properties.

Table 7-18 AcCrosstab properties

Property	Type	Description
Background Color	AcColor	The background color of the cross tab, row, column, or cell. This property does not affect subgroups of rows and columns. The background color of the row overwrites the background color of the column because rows are added on top of columns. The default value is Transparent.
Font	AcFont	The default font used in the cross tab.
Column Headings Border	Borders	Specifies the border color and thickness of a column heading. The default color is black. The default thickness is 1 point.
Definition	N/A	Accesses the Crosstab Builder.
LabelMultiple Values	Boolean	Displays labels for each column. The default value is True.
OuterBorder	Borders	Specifies the color and thickness of the outside border of a cross tab, row, or column. The default color is Black. The default thickness is 1 point for a cross tab, 0.5 point for a row or column.
RowHeadings Border	Borders	Specifies the border color and thickness of row headings of a cross tab. The default color is Black. The default thickness is 1 point.

(continues)

Table 7-18 AcCrosstab properties (continued)

Property	Type	Description
Value Placement	AcCrosstab ValueLayout	Specifies whether cross tab values appear horizontal or vertical. Valid values are: ■ ValuesHorizontal ■ ValuesVertical The default value is ValuesHorizontal.

Methods for Class AcCrosstab

Method defined in Class AcCrosstab

FinishBuilding

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue, SetValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcCrosstab::FinishBuilding method

Finishes building the data collector, and creates and populates the visual data structure.

Syntax Sub FinishBuilding( )

Class AcCurrencyControl

Displays a Currency value. Figure 7-22 shows the class hierarchy of AcCurrencyControl.

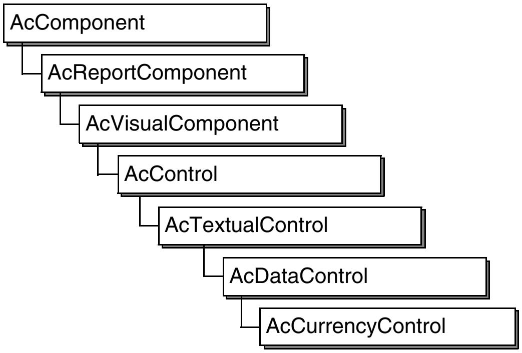


Figure 7-22 AcCurrencyControl

Description Use the currency control to display a Currency value. You can also use a double control or integer control to display numeric values.

See also Class AcControl
Class AcDataControl
Class AcDoubleControl
Class AcIntegerControl
Class AcTextualControl

Variable

Table 7-19 describes the AcCurrencyControl variable.

Table 7-19 AcCurrencyControl variable

Variable	Type	Description
DataValue	Currency	Stores the value of the control

Methods for Class AcCurrencyControl

Methods inherited from Class AcDataControl

Format, GetGroupKey, IsSummary

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcDataAdapter

An abstract base class that defines the logic of classes that form a data stream. The data stream collects, processes, and delivers data to the report. The parts of a data stream are called data adapters. Figure 7-23 shows the class hierarchy of AcDataAdapter.

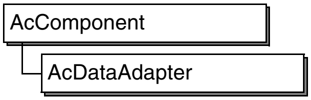


Figure 7-23 AcDataAdapter

Description AcDataAdapter is the abstract base class for the two types of data adapter base classes, AcDataSource and AcDataFilter.

A data source collects data from an input source, such as a database or a spreadsheet. A data filter processes the collected data. Both classes work together to produce and deliver formatted data, called data rows, to the report. You can limit the number of data rows a data adapter retrieves by using the FetchLimit property. This feature is useful for testing and debugging report designs, especially designs that generate large numbers of pages.

AcDataAdapter defines the core logic for how data adapters work with connections and data rows. This class also defines the basic algorithms for random access to data.

Class protocol

Table 7-20 describes AcDataAdapter's protocol, which defines the tasks that all data adapters perform.

Table 7-20 Class protocol for AcDataAdapter

Method	Task
New( )	Initializes the data adapter.
Start( )	Opens the data adapter. For a data source, this method opens the input source from which to read data. For a data filter, this method opens the input adapter. An input adapter is the data adapter that supplies data rows to the data filter.
Fetch( )	Reads one row from the data adapter.
Finish( )	Closes the data adapter. For a data source, this method closes the input source. For a data filter, this method closes the input adapter(s).

Subclassing AcDataAdapter

You typically derive from one of AcDataAdapter's more specialized subclasses, AcDataSource or AcDataFilter.

Variables

Table 7-21 lists AcDataAdapter variables.

Table 7-21 AcDataAdapter variables

Variable	Type	Description
FetchLimit	Integer	The number of data rows the data adapter retrieves.
IsOpen	Boolean	True when the data adapter is open. The IsOpen variable is set to True in Start( ) and False in Finish( ). Use IsStarted( ) to obtain the value of this variable.
Position	Integer	The current input position. Start( ) sets Position to 1. Each call to AddRow( ) increments it by 1. Your other overridden methods must maintain Position. Use GetPosition( ) to obtain the value of this variable.

Properties

Table 7-22 lists AcDataAdapter properties.

Table 7-22 AcDataAdapter properties

Property	Type	Description
Connection	AcConnection Structure Reference	Identifies the connection to use for the data adapter.
DataRow	AcDataRow Structure Reference	Identifies the data row to use with the data adapter.
FetchLimit	Integer	The number of data rows the data adapter retrieves. Limiting the number of data rows is useful for testing and debugging report designs when you need a small data sample.

Methods for Class AcDataAdapter

Methods defined in Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

See also Class AcDataRow
Class AcDataSource
Class AcMultipleInputFilter
Class AcSingleInputFilter

AcDataAdapter::AddRow method

Adds a row to the data adapter. If you override Fetch( ), call AddRow( ) after you instantiate the data row and set its variables. AddRow( ) performs the following tasks:

- Sets the row's RowNumber variable to the current input position
- Advances the input position
- Calls the OnRead( ) method for the data row so the row can compute its computed column values

Syntax Sub AddRow(row As AcDataRow)

Parameter **row**
The new data row.

See also AcDataAdapter::Fetch method
AcDataRow::OnRead method

AcDataAdapter::AddSortKey method

Adds a dynamic sort key column. Group sections provide the ability to automatically update the ORDER BY clause of the SELECT statement for a SQL query source to match the sequence a report needs. The group sections inform the data source of the preferred sort order by calling AddSortKey( ) and passing the group section's Key property value.

If you write code that uses a data source, your custom code can call AddSortKey( ) to customize the sort order of the data. If you create a custom data adapter, you can override this method to support custom run-time sorting.

Unless a subclass specifically overrides AddSortKey( ), the base AcDataAdapter class raises a run-time error if the report calls this method.

Syntax Sub AddSortKey(keyName As String, sortSense As AcSortSense)

Parameters **keyName**
The name of one of the columns in the SELECT clause.

sortSense
The direction of the sort. Specify either SortAscending or SortDescending.

See also AcDataAdapter::CanSortDynamically method

AcDataAdapter::CanSeek method

Determines whether the data adapter supports random access to data. The default setting for CanSeek( ) is False. In derived classes, you can override CanSeek( ) to return True if you want support for random access.

Syntax Function CanSeek( ) As Boolean

Returns True if the data adapter supports random access.
False if the data adapter does not support random access.

AcDataAdapter::CanSortDynamically method

Determines whether the data adapter supports dynamic ordering. AcDataAdapter assumes that the adapter does not support dynamic sorting, so the default setting for CanSortDynamically( ) is False. If a custom data adapter supports sorting, you can override this method to return True, then override AddSortKey( ) to accept the sort columns.

Call CanSortDynamically( ) to ensure that a data adapter supports custom sorting before calling AddSortKey( ).

Syntax Function CanSortDynamically( ) As Boolean

Returns True if the data adapter supports dynamic ordering.
False if the data adapter does not support dynamic ordering.

See also AcDataAdapter::AddSortKey method

AcDataAdapter::CloseConnection method

Closes a local connection that NewConnection( ) returns. For information about overriding this method, see the AcDataAdapter::OpenConnection method.

Syntax Sub CloseConnection(connection As AcConnection)

Parameter **connection**
The connection to close.

See also AcDataAdapter::GetConnection method
AcDataAdapter::NewConnection method
AcDataAdapter::OpenConnection method
AcDataAdapter::SetConnection method

AcDataAdapter::Fetch method

Reads the row at the position identified by GetPosition( ). Fetch( ) reads one row from the data adapter at the position identified by GetPosition( ). Fetch( ) then advances the position by one. If the current position is past the end of the input set, Fetch( ) returns Nothing.

If you create a custom data source or filter, you must override this method to fetch a row. You must handle repeated calls to Fetch( ), even after Fetch( ) reaches the end of the input set. Also, your override should call AddRow( ) each time it instantiates a new row.

If you write a custom Fetch( ) method for a data filter, and your Fetch( ) method simply passes along rows created by an input adapter, then you should not call AddRow( ) because it was already called by the input adapter. Instead, increment Position directly.

When you override a data adapter's Fetch( ) method, the FetchLimit property value has no effect. You need to check this property's value in your code if you want to limit the number of data rows to retrieve.

Syntax Function Fetch( ) As AcDataRow

Returns A reference to the data row fetched.
Nothing if the current position is past the end of the input set.

Example The following example shows how to accumulate the projected and actual cost figures for each month and chart the accumulated results, when the data is not normalized. For example, the report receives the data as one value per row, with one row for January Budget, one row for January Actual, and so on. To chart this data, the report design uses a data filter to normalize the data returned from the query.

The code in the Fetch( ) method of the filter receives a data row from the data source SqlQuerySource, which contains the database query. For each call to Fetch( ), the data filter splits off another amount field and returns it to the report. The dollars accumulate each month. For example, the budget amount for March shows the sum of January, February, and March budgets.

```
Function Fetch( ) As AcDataRow
   Dim aFltrDataRow As FilterDataRow
   ' Get the row returned from the SQL Query
   If queryDataRow is Nothing Then
      Set queryDataRow = InputAdapter.Fetch( )
      If queryDataRow Is Nothing Then Exit Function
      dataPointCount = 0
      actualTotal = 0.0
      planTotal = 0.0
   End If
   ' Run through each of the twelve monthly fields on the query
   ' data row and return one row for each month-and-amount
   ' combination to the report
   Set aFltrDataRow = New FilterDataRow
   aFltrDataRow.month = ( dataPointCount \ 2 ) + 1
   aFltrDataRow.amountType = dataPointCount MOD 2
   'Type of 0 = actual, 1 = plan
   dataPointCount = dataPointCount + 1
   If dataPointCount = 25 Then
      Exit Function
   End If

   Select Case dataPointCount
      Case 1
         aFltrDataRow.amntToChart = queryDataRow.aJanDollars
      Case 2
         aFltrDataRow.amntToChart = queryDataRow.bJanDollars
      Case 3
         aFltrDataRow.amntToChart = queryDataRow.aFebDollars
      Case 4
         aFltrDataRow.amntToChart = queryDataRow.bFebDollars
…
      Case 23
         aFltrDataRow.amntToChart = queryDataRow.aDecDollars
      Case 24
         aFltrDataRow.amntToChart = queryDataRow.bDecDollars
   End Select

   If aFltrDataRow.amountType = 0 Then
      aFltrDataRow.amntToChart = aFltrDataRow.amntToChart +
         actualTotal
      actualTotal = aFltrDataRow.amntToChart
   Else
      aFltrDataRow.amntToChart = aFltrDataRow.amntToChart +
         planTotal
      planTotal = aFltrDataRow.amntToChart
   End If

   Set Fetch = aFltrDataRow
   AddRow(Fetch)
End Function
```

AcDataAdapter::Finish method

Closes the data adapter. If the data adapter is a data source, Finish( ) closes the input source, which can be a query, a file, or another source. If the data adapter is a filter, Finish( ) closes each of the input adapters. An input adapter is the data adapter that supplies data rows to the data filter.

If the connection was created using NewConnection( ), Finish( ) calls CloseConnection( ) to close the connection.

In derived classes, you can override Finish( ) to do additional work when the Factory finishes processing the data adapter. Call Super::Finish after your code.

Example See the AcReportComponent::FindContentByClass method for an example of how to use the Finish( ) method.

Syntax Sub Finish( )

See also AcReportComponent::FindContentByClass method
AcDataAdapter::Start method

AcDataAdapter::FlushBuffer method

If the data adapter uses internal buffering to enable random access, this method flushes all the buffered rows. Call FlushBuffer( ) to reclaim memory. FlushBuffer( ) calls FlushBufferTo( ) to clear the buffer. FlushBuffer( ) does nothing if the data source does not use buffering.

After the buffer flush is complete, the read position moves to the first row past those that were in the buffer.

Derived classes that support buffering should override this method.

Syntax Sub FlushBuffer( )

See also AcDataAdapter::FlushBufferTo method

AcDataAdapter::FlushBufferTo method

Flushes all buffered rows. Call FlushBufferTo( ) to clear all rows, up to and including a specified row. If the data adapter supports buffering and the read position is less than posn, the current read position is set to posn + 1.

Syntax Sub FlushBufferTo(posn As Integer)

Parameter **posn**
The data row to flush to.

See also AcDataAdapter::FlushBuffer method

AcDataAdapter::GetConnection method

Returns a connection. You might need to get a connection if you want to customize the process of selecting and instantiating a connection. For example, you can call GetConnection( ) to return the connection, which you then pass as an argument to SetConnection( ).

Syntax Function GetConnection( ) As AcConnection

Returns A reference to the connection associated with the data adapter.

See also AcDataAdapter::CloseConnection method
AcDataAdapter::NewConnection method
AcDataAdapter::OpenConnection method
AcDataAdapter::SetConnection method

AcDataAdapter::GetPosition method

Returns the position of the next row to fetch, starting with 1. The number of rows fetched to date from a sequential source is one less than GetPosition( ).

Syntax Function GetPosition( ) As Integer

Returns The current row number or 1 when the data adapter is first opened.

AcDataAdapter::IsStarted method

Returns True if you open the adapter using a call to Start( ). Returns False if the data adapter was never started, or if you close the adapter using a call to Finish( ).

Syntax Function IsStarted( ) As Boolean

Returns The value of the IsOpen variable.

AcDataAdapter::NewConnection method

Instantiates the connection class that you place in the Connection slot of the data adapter in Report Structure. You can override NewConnection( ) to customize the process for selecting a connection. For example, if your report needs a different connection depending on the type of data adapter in use, you can override NewConnection( ) to write the conditional logic.

Syntax Function NewConnection( ) As AcConnection

Returns The new connection.

See also AcDataAdapter::CloseConnection method
AcDataAdapter::GetConnection method
AcDataAdapter::OpenConnection method
AcDataAdapter::SetConnection method

AcDataAdapter::NewDataRow method

Instantiates the data row class that appears in the DataRow slot for this adapter. You can override this method to customize the data row to instantiate.

Fetch( ) calls NewDataRow( ) each time it reads a new data row. If you create a custom data source and override Fetch( ) to specify how the data source retrieves data rows, call NewDataRow( ) to instantiate the data row. For an example of creating a custom data source, see *Accessing Data using e.Report Designer Professional.*

Syntax Function NewDataRow( ) As AcDataRow

Returns The new data row.

AcDataAdapter::OpenConnection method

Opens the connection returned by NewConnection( ). If NewConnection( ) returns a local connection, the data adapter calls the OpenConnection( ) method to open the connection. You can override this method to customize the connection before opening it.

For example, suppose you have five data sources in a report. The first and third data sources use an Oracle connection, and the others use an ODBC connection. You can place the ODBC connection in the common section, then create a static variable to hold the Oracle connection. In the data adapters that work with the Oracle connection, override NewConnection( ) to get the shared connection, then override OpenConnection( ) and CloseConnection( ) to do nothing. Code elsewhere must then open and close the Oracle connection.

Syntax Function OpenConnection(connection As AcConnection) As Boolean

Parameter **connection**
The connection to open.

Returns True if the connection opens.
False if the connection does not open.

Example If the data adapter sets the database or user name based on a parameter, your override should call any relevant superclass method and return the value of that method, as shown in the following example:

```
Function OpenConnection( connection As AcConnection ) As Boolean
   Dim conn As AcODBCConnection
   Set conn.DataSource = "testDB"
   OpenConnection = Super::OpenConnection( connection)
End Function
```

Alternatively, you can override OpenConnection( ), along with NewConnection( ) and CloseConnection( ), to implement a custom scheme for sharing connections.

See also AcDataAdapter::CloseConnection method
AcDataAdapter::GetConnection method
AcDataAdapter::NewConnection method
AcDataAdapter::SetConnection method

AcDataAdapter::Rewind method

Moves the fetch position to position one, the beginning of the input set. Rewind( ) is equivalent to:

```
SeekTo( 1 )
```

Syntax Sub Rewind( )

AcDataAdapter::SeekBy method

Moves the fetch position by a given amount, relative to the current position. If you specify an offset of 0, then the position does not move. Negative offsets move the position toward the beginning of the input set. Positive offsets move the position toward the end of the input set. SeekBy( ) is equivalent to:

```
SeekTo( GetPosition( ) + offset )
```

Derived classes need not override this method. They should override SeekTo( ) instead.

Syntax Sub SeekBy(offset As Integer)

Parameter **offset**
The number of rows, relative to the current position, to move.

AcDataAdapter::SeekTo method

Moves the fetch position to a given location. The position is relative to the beginning of the input set. The first row is position one. After a call to SeekTo( ), the next call to GetPosition( ) returns the position you specified. Similarly, the next call to Fetch( ) returns the row to the position you specify.

If you specify a position less than one, the data adapter uses position one instead. Similarly, if you seek a position past the end of the input set, the position will be set to one past the end of the file.

SeekTo( ) is available only in data adapters that support random access. If the adapter provides only sequential access, SeekTo( ) raises a run-time error.

Derived classes that support random access must override this method.

Syntax Sub SeekTo(posn As Integer)

Parameter **posn**
The position from which to read on the next call to Fetch( ).

AcDataAdapter::SeekToEnd method

Reads rows from the current position to the end of the input set. This method is equivalent to calling Fetch( ) in a loop until Fetch( ) returns Nothing.

After a call to SeekToEnd( ), GetPosition( ) returns one greater than the number of rows in the input set.

Derived classes need not override this method.

Syntax Sub SeekToEnd( )

AcDataAdapter::SetConnection method

Sets a connection for the data adapter. The connection must be open and you must call SetConnection( ) before calling Start( ). Any connection NewConnection( ) returns has precedence.

Syntax Sub SetConnection(theConnection As AcConnection)

See also AcDataAdapter::CloseConnection method
AcDataAdapter::GetConnection method
AcDataAdapter::NewConnection method
AcDataAdapter::OpenConnection method

AcDataAdapter::Start method

Opens the data adapter. For a data source, the Start( ) method opens the input source from which to read data. For a data filter, Start( ) opens the input adapter, the data adapter that supplies data rows to the data filter.

Start( ) also calls NewConnection( ) to instantiate the connection. If NewConnection( ) returns a connection, Start( ) calls OpenConnection( ) to open the connection.

You can override Start( ) to add startup code for your class. You should, however, call the superclass method first, then continue with your own initialization only if the superclass method returns True.

```
Function Start( ) As Boolean
   Start = Super::Start( )
   If Not Start Then
      Exit Function
   End If
   ' Custom startup code
End Function
```

Syntax Function Start( ) As Boolean

Returns True if the data adapter opens.
False if the data adapter does not open.

See also AcDataAdapter::Finish method

Class AcDatabaseSource

An abstract base class for data sources that retrieve data from databases. Figure 7-24 shows the class hierarchy of AcDatabaseSource.

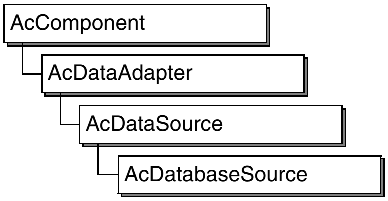


Figure 7-24 AcDatabaseSource

Description AcDatabaseSource is an abstract base class that provides the standard logic for retrieving rows from a relational database cursor. It defines the methods for binding parameters to the database statement, opening the cursor, binding the data row to the cursor, retrieving rows from the cursor, and closing the cursor.

See also Class AcDataAdapter
Class AcDataRow
Class AcDataSource
Class AcSqlQuerySource

Methods for Class AcDatabaseSource

Methods defined in Class AcDatabaseSource

BindDataRow, BindStaticParameters, GetCursor, GetDBConnection, OpenCursor, SetStatementProperty

Method inherited from Class AcDataSource

HasFetchedLast

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDatabaseSource::BindDataRow method

Binds a data row to a SQL query cursor. Figure 7-25 shows the binding relationship between columns in the row that the cursor returns and variables in the data row that you defined.

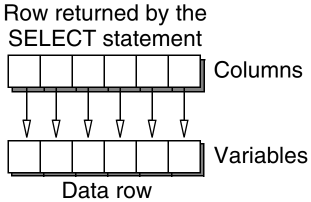


Figure 7-25 The mapping of columns in a row to variables in a data row

Syntax Sub BindDataRow(cursor As AcDBCursor)

Parameter **cursor**

The cursor to which to bind the data row.

Example The following example shows how to set up a SQL query, define a data row, and bind the row fetched by the cursor to the data row.

First, override the Start( ) method on the data stream component to define the SQL SELECT statement.

```
Function Start( ) As Boolean
    Start = Super::Start( )
   Dim selectClause As String
   Dim fromClause As String
   Dim whereClause As String
   Dim aStmt As String
   ' prepare the text for the query statement
   selectClause = "SELECT DISTINCT salesreps.last,
      salesreps.first, orders.orderID"
   fromClause = " FROM customers, orders, salesreps"
   whereClause = " WHERE salesreps.repID = customers.repID AND
      orders.custID = customers.custID"
   aStmt = selectClause & fromClause & whereClause
   ' open a cursor for the above query statement
    OpenCursor( aStmt )
End Function
```

Then, subclass AcDataRow to create a data row by programming in Actuate Basic or using e.Report Designer Professional.

```
Class fSqlDataRow Subclass of AcDataRow
   Dim Salesreps_first As String
   Dim Salesreps_last As String
   Dim orders_orderID As Integer
End Class
```

Override BindDataRow( ) to bind the row the cursor fetched and your data row subclass.

```
Sub BindDataRow( cursor As AcDBCursor )
   ' BindColumn statement must be run for each column in the
   ' SELECT statement
   cursor.BindColumn( 1, "fSqlApp::fSqlDataRow",
      "salesreps_last" )
   cursor.BindColumn( 2, "fSqlApp::fSqlDataRow",
      "salesreps_first" )
   cursor.BindColumn( 3, "fSqlApp::fSqlDataRow",
      "orders_orderID" )
End Sub
```

AcDatabaseSource::BindStaticParameters method

The OpenCursor( ) method of AcDatabaseSource calls BindStaticParameters( ) to bind parameters to the cursor for a SQL statement. You must override BindStaticParameters( ) if the SQL statement uses parameters.

Syntax Sub BindStaticParameters(cursor As AcDBCursor)

Parameter **cursor**
The cursor to which to bind the parameters.

Example The following example shows how to accomplish the following tasks:

- Code a SELECT statement that uses a parameter.
- Override BindStaticParameters( ) to bind the parameter to the statement's cursor.

```
' SELECT statement
SELECT fname, lname FROM Customers WHERE Customer.State = ?
```

```
' BindStaticParameters( ) code
Sub BindStaticParameters( cursor As AcDBCursor )
   cursor.BindParameter( 1, "CA" )
EndSub
```

AcDatabaseSource::GetCursor method

Returns the database cursor associated with the data source. The cursor is available after the Start( ) method calls OpenCursor( ). To get the associated database statement, call GetStatement( ) from the cursor.

Syntax Function GetCursor( ) As AcDBCursor

Returns The database cursor from which the database source retrieves rows.

See also AcDBCursor::GetStatement method

AcDatabaseSource::GetDBConnection method

Returns the database connection associated with the data source. This method is equivalent to GetConnection( ), except the type of the returned connection is the more derived AcDBConnection class. You can use this connection, for example, to raise an error if a database error occurs.

Syntax Function GetDBConnection( ) As AcDBConnection

Returns The database connection.

See also AcDataAdapter::GetConnection method
Class AcDBConnection

AcDatabaseSource::GetPreparedStatement method

Gets the statement on which to execute the database cursor. A prepared statement is one on which the Prepare( ) method has been called.

Syntax Function GetPreparedStatement( ) As AcDBStatement

See also AcDBConnection::Prepare method

AcDatabaseSource::OpenCursor method

Opens the database cursor. OpenCursor( ) is a helper method called by your implementation of Start( ). The Start( ) method must create the SQL statement. Start( ) then calls OpenCursor( ) to prepare the statement, bind the parameters, allocate and open a cursor, and bind the data row to the cursor. OpenCursor( ) calls BindStaticParameters( ) to bind static parameters to the statement. If your SQL statement already has parameters, you must override BindStaticParameters( ). OpenCursor( ) also calls BindDataRow( ), which you must override to bind the data row to the cursor.

Syntax Sub OpenCursor(stmt As String)

See also AcDatabaseSource::BindDataRow method
AcDatabaseSource::BindStaticParameters method

AcDatabaseSource::SetStatementProperty method

Assigns a string value to the specified property. The ODA driver interprets this value when the report runs.

Syntax Sub SetStatementProperty(propName As String, propValue As String)

Parameters **propName**
The name of the property to which to assign the value.

propValue
The value to assign to the property.

Class AcDataControl

The base class for controls that display data from data rows. Figure 7-26 shows the class hierarchy for AcDataControl.

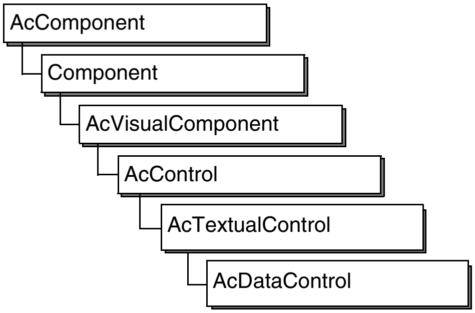


Figure 7-26 AcDataControl

Description AcDataControl defines the logic for setting the values of data controls, which display data from the input source. Each data control displays one piece of data, such as a name, a date, a quantity, or a total. To specify what data a data control should display, you assign a value expression to the control's ValueExp property. If you leave this property blank, the default value of the data control is Null.

Subclassing AcDataControl

Do not derive directly from AcDataControl. Actuate products provide a different type of data control for each data type, as follows:

- AcCurrencyControl displays currency data.
- AcDateTimeControl displays dates.
- AcDoubleControl displays double precision floating point numbers.
- AcDynamicTextControl displays text that has variable size and optionally contains HTML or RTF formatting information.
- AcIntegerControl displays numeric data other than currencies, floating-point numbers, or dates and times.
- AcTextControl displays string data.

These specialized classes derive from AcDataControl. You derive a new control from one of the specialized subclasses of AcDataControl.

Building a control without a data row

The Build( ) method of the frame that contains a control calls the control's Build( ) method instead of BuildFromRow( ). The call to the frame's Build( ) method occurs when you place a frame in a slot where the frame does not receive data rows. For information about when the framework calls a frame's Build( ) method, see AcBaseFrame.

Build( ) sets the value of a control using data from a source other than a data row. The value can be from an Actuate Basic function call, a method call, a variable, or a constant. The default behavior for Build( ) is to call SetValue( ) to set the value of the control.

You can override the data control's Finish( ) method to perform custom processing, such as changing the value or property of the control depending on a condition.

The following example overrides a text control's Finish( ) method to change the control's data value to a different string. This code also changes the display text to a different color when Date$( ) returns "04-15-2010". The report developer assigned Date$( ) to the control's ValueExp property.

```
Sub Finish( )
   If DataValue = "04-15-2010" Then
      DataValue = "Tax day"
      Font.Color = Red
   End If
   Super::Finish( )
End Sub
```

Building a control from the data row

BuildFromRow( ) sets the value of a control using data from a data row. The enclosing frame's BuildFromRow( ) method calls the control's BuildFromRow( ) method.

Controls can have one of the following three relationships to a data row:

- The control needs no data. Some controls, such as graphic images and lines, require no data from the data row. Controls that do not take data from a data source are called constant controls. A label control is a constant control.
- The control uses data from a single row. A control that returns a customer's name from the data source displays data from a single data row.
- The control uses data from multiple rows. Some controls summarize data from a set of rows. These controls are called aggregate controls. A currency control that sums a customer's payments for the past three quarters is an example of an aggregate control.

The BuildFromRow( ) method provides a general mechanism for handling all three relationships. If the control does not need the data row, BuildFromRow( ) returns Finished Building. If the control uses a single row, BuildFromRow( ) sets the control's value and returns Finished Building. If the control uses multiple rows, BuildFromRow( ) returns Continue Building.

You can override a control's Finish( ) method to perform custom processing, such as changing the value or a property of the control depending on a condition.

About controls that use a single data row

If a control uses a single data row, BuildFromRow( ) performs the following tasks:

- Calls SetValue( ) on the first row to set the value of the control
- Calls SetTocEntry( ) to set the Table of Contents entry for the control
- Calls OnRow( ), which you can override to do further processing on the row
- Returns Finished Building

Controls that need no data rows work as if they need only one row. These controls simply ignore the row. In this case, SetValue( ) does nothing.

About controls that use multiple data rows

Aggregate controls work with any number of rows and summarize data from those rows. For example, an aggregate control might show the minimum, maximum, sum, or average of a group of sales records. In this case, BuildFromRow( ) processes rows a bit differently than in the single row case. On the first row, BuildFromRow( ) calls SetTocEntry( ) to set the Table of Contents entry. For all rows, BuildFromRow( ) calls both SetValue( ) and OnRow( ). For aggregate controls, BuildFromRow( ) always returns Continue Building.

The following code overrides an integer control's Finish( ) method so that the data value is green when the value is greater than 20:

```
Sub Finish
   If DataValue > 20 Then
      Font.Color = green
   End If
   Super::Finish
End Sub
```

See also

Class AcControl
Class AcCurrencyControl
Class AcDateTimeControl
Class AcDoubleControl
Class AcIntegerControl
Class AcTextControl
Class AcTextualControl

Class protocol

The protocol for AcDataControl is the same as for AcControl, except that it adds the capability to set the value of a control. Table 7-23 lists methods for AcDataControl.

Table 7-23 Class protocol for AcDataControl

Method	Task
New( )	Initializes the control
Start( )	Prepares the data control for building
SetValue( )	Called by BuildFromRow( ) to set the value of the control, typically from the expression in the ValueExp property
OnRow( )	Called by BuildFromRow( ) to let the control do additional processing for the row

Properties

Table 7-24 lists AcDataControl properties.

Table 7-24 AcDataControl properties

Property	Type	Description
Format	String	Formats the data control
ValueExp	Expression	Specifies the value of the control
ValueType	AcControl ValueType	Specifies how many data rows the control will process

Methods for Class AcDataControl

Methods defined in Class AcDataControl

Format, GetGroupKey

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue, SetValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class Component

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDataControl::Format method

Returns the format pattern specified in the control's Format property. The Format property accepts any of the format patterns available to the Actuate Basic Format$ function. Format( ) returns the format pattern as a string. For example, if you specify (@@@) @@@-@@@@ as the format pattern for a text control that displays telephone numbers, Format( ) returns the string (@@@) @@@-@@@@. The GetText( ) method uses this return value to format the control's value for display.

Syntax Function Format( ) As String

Returns The format pattern that formats the control's value.

See also AcControl::GetText method

AcDataControl::GetGroupKey method

Returns the key for the group section, if any, that contains the control.

If the GroupOn property is set to the default value of GroupOnEachValue, GetGroupKey( ) returns the value of the column key.

If the GroupOn property is set to GroupOnCustom, GetGroupKey( ) returns the group key set by the GetKeyValue( ) method.

For all other values of GroupOn, GetGroupKey( ) returns the first value in the range of values for the key. For example, if GroupOn is set to GroupOnYear, GetGroupKey( ) returns values such as 1/1/2004, 1/1/2009.

Syntax Function GetGroupKey( ) As Variant

Returns The group key. Nothing if there is no group section.

Class AcDataFilter

The abstract base class for all data filter classes. Figure 7-27 shows the class hierarchy of AcDataFilter.

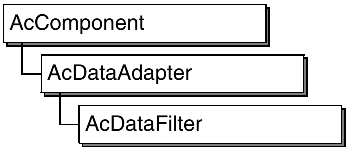


Figure 7-27 AcDataFilter

Description AcDataFilter is the base class for the two general types of data filter classes: AcSingleInputFilter and AcMultipleInputFilter. A single-input filter accepts input from one data adapter, processes the data, then passes it to the next data adapter or the report section. A multiple-input filter performs the same tasks but accepts input from any number of data adapters.

Subclassing AcDataFilter

You typically do not derive directly from AcDataFilter. To customize a data filter, use one of the data filters derived from AcDataFilter and override Fetch( ) to implement the filtering algorithm.

See also Class AcDataAdapter
Class AcMultipleInputFilter
Class AcSingleInputFilter

Methods for Class AcDataFilter

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcDataFrame

Defines the logic for how frames work with data rows. Figure 7-28 shows the class hierarchy of AcDataFrame.

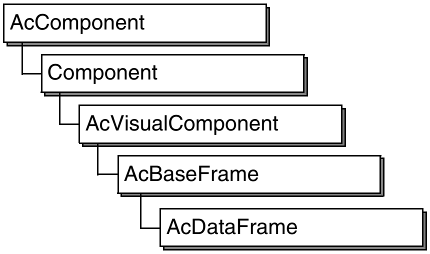


Figure 7-28 AcDataFrame

Description AcDataFrame provides the mechanism for building a frame's contents using values from a data row. The mechanism includes passing a data row to each of the frame's contents and accumulating aggregates.

Building a frame

The framework calls BuildFromRow( ) to set the data value of a control in a frame using a value from a data row. The frame's BuildFromRow( ) method calls BuildFromRow( ) for each of the frame's components.

The return value of BuildFromRow( ) indicates whether the frame processed the row. The frame's BuildFromRow( ) method determines its return value based on the return values of each component's BuildFromRow( ) method. The frame uses this return value to determine whether to continue passing rows to the frame's components or instantiate a new frame to process a row.

A frame can process either a single row or an unlimited set of rows.

If the frame processes only a single row, BuildFromRow( ) returns Finished Building after the frame's contents set their values. This return value indicates to the frame's container that it should instantiate a new frame to process the next data row.

If a frame processes multiple rows, as when the frame contains a chart, BuildFromRow( ) always returns Continue Building. This return value indicates to the frame's container that it should send further data rows to the frame.

If any one of the frame's contents returns Continue Building, the frame's BuildFromRow( ) method returns Continue Building. This process supports including, for example, a line control, which requires no rows, a text control, which requires one row, and a chart, which processes multiple rows, within a single frame. The controls that process only one row processes only the first row and ignores all subsequent rows.

You can override a frame's BuildFromRow( ) method to perform custom processing, such as conditionally accepting rows or accepting a limited number of rows.

Subclassing AcDataFrame

You typically do not subclass AcDataFrame unless you must change the way a frame builds its contents.

Methods for Class AcDataFrame

Methods inherited from ClassAcBaseFrame

AddToAdjustSizeList, BindToFlow, FindContentByClass, FindContentByClassID, GetControl, GetControlValue, GetPageNumber, GetSearchValue, IsDataFrame, IsFooter, IsHeader, MakeContents, RebindToFlow, SearchAttributeName

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained,ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class Component

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcDataRow

A class that defines the characteristics of a data row. A data row is a record structure that contains data from a single input record, processed into a format that the report accepts. Typically, each variable in a data row maps to a single column, or field, of a record. Figure 7-29 shows the class hierarchy of AcDataRow.

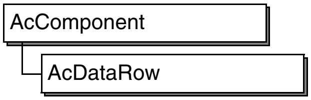


Figure 7-29 AcDataRow

Description AcDataRow works with data adapter classes, such as AcDataSource, AcSingleInputFilter, and AcMultipleInputFilter, to produce formatted data for the report. If you use the Query Editor or Textual Query Editor to build a SQL query, the framework creates the data row. If you create a custom data source or a custom data filter, you must create a custom data row that works with the data source or filter. Data rows are transient. The Factory creates them, passes them to the report, and deletes them.

The data source retrieves data from an input source and creates an instance of a subclass of AcDataRow for each record. The data filter filters and sorts the data as needed. Data filters are optional.

Figure 7-30 gives a high-level view of how the data from an input source is processed into data rows and sent to the report.

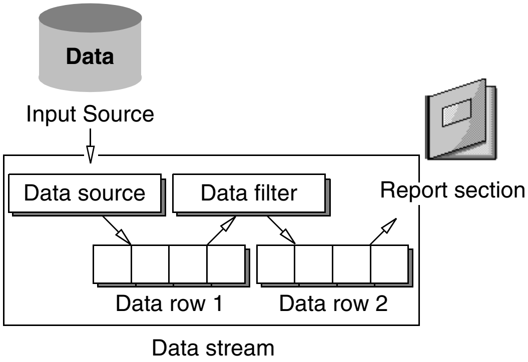


Figure 7-30 Overview of how data goes from an input source to a report

Subclassing AcDataRow

You must create a subclass of AcDataRow when you create a custom data source or filter. To derive a subclass from AcDataRow, take the following steps:

- Add the variables that represent the fields of the data row.

- Override the OnRead( ) method to compute the variable values based on the input fields.

The following example shows a derived AcDataRow class that defines the variables to hold the result of a query. The query returns four columns, AccountName, Address, CreditLimit, DueDate.

```
Class AccountSummary Subclass of AcDataRow
   Dim AccountName As String
   Dim Address As String
   Dim CreditLimit As Currency
   Dim DueDate As Date
End Class
```

Working with columns stored in a data row

Conceptually, data rows are composed of columns. The Actuate framework provides the following options for defining columns:

- A data row variable
- A table.column alias for a data row variable
- A method
- A member of a class or structure member variable
- A variable index

The framework provides two methods, GetValue( ) and SetValue( ) to help you work with columns stored in a data row. GetValue( ) retrieves the value of a column stored in a data row. SetValue( ) updates the value of a column stored in a data row. Using GetValue( ) and SetValue( ) simplifies programming by reducing the need to set up object reference variables for the data row columns that you need to retrieve or update. See the description of the AcDataRow::GetValue method for examples of using the various options for defining columns.

Variable

Table 7-25 describes the AcDataRow variable.

Table 7-25 AcDataRow variables

Variable	Type	Description
RowNumber	Integer	The number of the current row, starting with 1, within the data source

See also Class AcDataAdapter
Class AcDataSource
Class AcReportComponent

Methods for Class AcDataRow

Methods defined in Class AcDataRow

GetValue, OnRead, SetValue

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDataRow::GetValue method

Returns the value of the specified column or variable. Use GetValue( ) to write generic code that accesses the value of a column or variable within your data row. If you write code for a specific data row type, use an object reference variable for the data row and Actuate Basic's dot notation to access the variable directly. This code runs faster than calling GetValue( ) to access a data row variable.

You can use the following five types of values to pass the column name:

- A data row variable name
- A database column alias
- A method name
- A member of a class or a structure member variable
- A variable index

Using a data row variable name

The simplest approach is to call GetValue( ) using a data row variable name. If you create your own data row, you can use GetValue( ) to find the value of a variable. The following example shows how to get the account name from a data row variable named AccountName in the AccountSummary data row:

```
Sub AccountInfo( row As AcDataRow )
   DataValue = row.GetValue( "AccountName" )
End Sub
```

Using a database column alias

When you use Query Editor or Textual Query Editor to build a query data stream, the framework builds a data row for you. The framework also maps from database table.column names to data row variables. You can then pass one of these database names to GetValue( ) to obtain the value of the corresponding data row variable. This feature of GetValue( ) enables dynamic binding, the ability to work with a row at run time, even if you do not know the exact data type of the row. To write code that operates independently of the data row structure, use table.column names to refer to columns.

The following example shows how to retrieve a customer name derived from the Name column of the Customer table:

```
Sub AccountInfo( row As AcDataRow )
   DataValue = row.GetValue( "Customer.Name" )
End Sub
```

Using a method name

Sometimes you need to create a set of computations on columns. For example, if you have a data row for a customer invoice, you might want to know how much of the total amount of that invoice is not yet due, how much is now due, or how much is 30, 60, or 90+ days past due. You can create a set of variables to hold these amounts. It is easier, however, to provide a set of methods that perform the calculation, as shown in the following example:

```
Function Amount30DaysLate( ) As Currency
   If DueDate + 30 <= Date( ) And Date( ) < DueDate + 60 Then
      Amount30DaysLate = InvoiceAmount
   Else
      Amount30DaysLate = 0.0
   End If
End Function
```

You can refer to such methods using square bracket notation in value expressions for controls or by using the GetValue( ) method. You can create methods that act like data row columns. These methods have the following restrictions:

- They must return a value consistent with GetValue( ) return values.
- They must accept no arguments.

The following example accesses the Amount30DaysLate( ) method:

```
[Amount30DaysLate]
row.GetValue( "Amount30DaysLate" )
```

Using a structure or an object

You can use square bracket notation in a value expression or GetValue( ) in code to access members of structures or objects nested inside the data row. In the following example, the data row contains an AddressStruct structure that declares the FullName, Street, City, State, and Zip variables:

```
Type AddressStruct
   FullName As String
   Street As String
   City As String
   State As String
   Zip As String
End Type
```

If the customer data row also has a variable Address of type AddressStruct, you can access members of that structure using dot notation as shown in the following example:

```
[Address.FullName]
[Address.Street]
…
row.GetValue( "Address.FullName" )
row.GetValue( "Address.Street" )
…
```

Similarly, if the data row contains an object reference variable to another object, you can access members of that object using the same dot syntax. You can still convert the AcDataRow variable to point to your particular data row class, then access the variable directly, as shown in the following example:

```
Sub AccountInfo( row As AcDataRow )
   Dim accRow As AccountSummary
   Set accRow = row
   DataValue = accRow.AccountName
End Sub
```

If you access the data row variable directly, your control works with only one specific type of data row. To ensure the code works with any data row that has the correct column or variable name, access data in a data row using the GetValue( ) method.

Using a variable index

You can use a variable index to access data row variables. For example, to iterate over variables in a data row, you can access the value of any data row variable by using an index corresponding to the variable’s position in the data row. The following code sample uses a variable index to access the values of data row variables. The Actuate Basic function, GetVariableCount, returns the total number of variables in the data row.

```
Sub AccountInfo( row As AcDataRow )
   Dim accRow As AccountSummary
   Dim colIndex As Integer
   Dim DataValue As Variant
   ' Compute the index of the first local variable in a
   ' data row subclassed from AcDataRow.
   Static firstRowVariableIndex As Integer
   if (firstRowVariableIndex = 0) Then
      Dim r As AcDataRow
      Set r = New AcDataRow
      firstRowVariableIndex = GetVariableCount( r ) + 1
      Set r = Nothing
   End If
…
```

```
   Set accRow = row
   For colIndex = firstRowVariableIndex to
+     GetVariableCount( accRow )
…
      DataValue = accRow.GetValue( colIndex )
…
   Next
…
End Sub
```

About the order of evaluation

Typically, the name you provide to GetValue( ) uniquely identifies one column alias, variable, function, structure, or object. If you have a column alias and a variable, function, structure, or object with the same name, the framework uses the column alias.

Syntax Function GetValue(colName As String) As Variant

Parameters **colName**
The column or variable name with the value to return.

Function GetValue(index As Integer) As Variant

index
The index of the variable that holds the value to return.

Returns The value of the given column or variable.

See also AcDataRow::GetValue method
AcDataRow::SetValue method

AcDataRow::OnRead method

Called by the data adapter after it creates the data row and sets the data row values. You can override OnRead( ) to manipulate variables in a data row. For example, you can set the value of a calculated variable based on other variables in the data row.

Syntax Sub OnRead( )

Example The following example overrides the data row's OnRead( ) method to calculate a value for the ExtendedCost variable. The calculation uses values in two other variables, Cost and Quantity.

```
Sub OnRead( )
   Super::OnRead( )
   ExtendedCost = Cost * Quantity
End Sub
```

AcDataRow::SetValue method

Sets the value of the specified column or variable. Use this method in generic code to set the value of a column or variable in a data row. If you write code for a specific data row type, use an object reference variable for the data row and Actuate Basic's dot notation to access the variable directly. This code runs faster than calling SetValue( ) to access a data row variable.

Syntax

```
Function SetValue( colName As Any, value As Any ) As Boolean
```

Parameters

colName
The column or variable name with the value to set.

value
The data value for the column or variable. The data type for this value must be the same as the type of the column. If the types do not match, it must be possible to convert the value's data type to the type of the column.

```
Function SetValue( index As Integer, value As Any ) As Boolean
```

index
The index of the variable with the value to set.

Returns

True if the value is set.
False if the data type for the value is not the same as the type for the column and it is not possible to convert the value's data type to the column's type.

See also

AcDataRow::GetValue method
Class AcDataRowBuffer
Class AcDataRowSorter

Class AcDataRowBuffer

A data filter that converts a sequential data stream into one that supports random access by buffering data rows. Figure 7-31 shows the class hierarchy of AcDataRowBuffer.

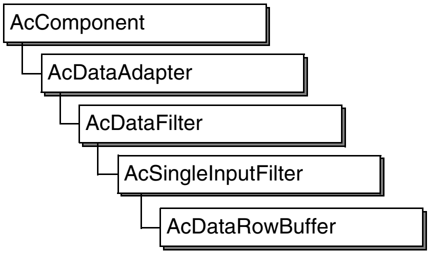


Figure 7-31 AcDataRowBuffer

Description Many data sources provide only the ability to read rows in a sequential first-to-last order. In some cases, you need to move through the data in random order, or make multiple passes over certain groups of data to print the data first in a chart, then in a table. The AcDataRowBuffer class acts as a converter to change a sequential data source into a random-access data stream. It does so by storing, or buffering, data rows as you read them so that you can return to them later. You do not need to use this class if the data source already provides random access.

Because this class supports random access, the CanSeek( ) method returns True. All the random-access methods, such as SeekTo( ) and Rewind( ), are available.

AcDataRowBuffer lets you manipulate data as if you had direct access to the input source. You can locate rows by specifying a row number with SeekTo( ) or by specifying a relative position using SeekBy( ). You can rewind to the beginning of the data buffer using Rewind( ) or advance to the end using SeekToEnd( ). The row number starts at 1. See class AcDataAdapter for details.

Optionally, you can flush the buffer as needed to recover disk space. For example, if you must make two passes over data for each customer, you can flush the rows for each customer as you complete the process. Flushing the buffer does not change the way you access rows. You cannot seek back to revisit the flushed rows.

You can also use the data row buffer class to gather data rows the report creates programmatically. For example, you can produce a report of account activity. For every account with exceptional items, you can create a second data row to print in a second report. To process these rows, create a data row buffer without an input adapter. Call AddRowToBuffer( ) to add each exception row to the buffer. Then, in a later report section, you can read and process these rows in the usual way.

See also Class AcDataAdapter
Class AcDataSource
Class AcSingleInputFilter

Methods for Class AcDataRowBuffer

Methods defined in Class AcDataRowBuffer

AddRowToBuffer, GetBufferCount, GetBufferStart

Methods inherited from Class AcSingleInputFilter

SetInput, GetInput, NewInputAdapter

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDataRowBuffer::AddRowToBuffer method

Adds a row to the end of the data row buffer. Use this method to add a row to the buffer programmatically. This method does not modify the current read position.

Syntax Sub AddRowToBuffer(row As AcDataRow)

Parameter **row**
The data row to add to the buffer.

AcDataRowBuffer::GetBufferCount method

Gets the number of rows currently in the buffer. Rows that have been flushed are not counted.

Syntax Function GetBufferCount( ) As Integer

Returns The number of rows currently in the buffer.

AcDataRowBuffer::GetBufferStart method

Gets the position of the first row in the buffer, relative to the beginning of the input set. If the buffer is empty, this method returns the position of the row that will become the first row in the buffer the next time you call Fetch( ).

Syntax Function GetBufferStart( ) As Integer

Returns Returns the position of the first row in the buffer, relative to the beginning of the input set.

Class AcDataRowSorter

A data filter that has the capability to sort rows in a buffer. Figure 7-32 shows the class hierarchy of AcDataRowSorter.

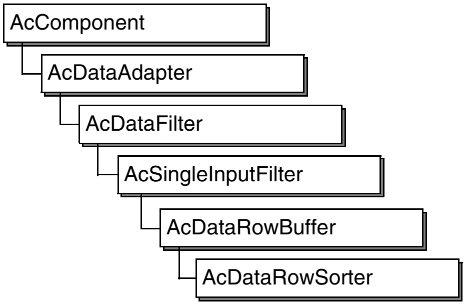


Figure 7-32 AcDataRowSorter

Description AcDataRowSorter is a data filter that uses the buffering capabilities of its superclass, AcDataRowBuffer, to read and store data rows. In addition, AcDataRowSorter provides a framework for subclasses to implement a sort algorithm.

To implement the sort functions and process data rows:

- Derive a class from AcDataRowSorter.
- Override the Compare( ) method, which implements the sort algorithm.

See also Class AcDataAdapter
Class AcDataSource
Class AcDataRowBuffer
Class AcSingleInputFilter

Methods for Class AcDataRowSorter

Methods defined in Class AcDataRowSorter

Compare, CompareKeys

Methods inherited from Class AcDataRowBuffer

GetBufferCount, GetBufferStart, AddRowToBuffer

Methods inherited from Class AcSingleInputFilter

SetInput, GetInput, NewInputAdapter

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDataRowSorter::Compare method

Compares two data rows. The Compare( ) method determines whether one row comes before or after another in the sort order. The comparison is based on sort keys. A sort key is a column on which you want to base the sort.

Do not use a non-integer number field as a sort key because of the rounding errors that can result from converting from a floating point to binary form.

When writing the Compare( ) method, cascade the comparisons as follows:

- If the first sort key column differs between the two rows, return 1 if key 1 > key 2 or -1 if key 1 < key 2.
- If the first sort key columns are the same, then repeat the process on the second key and any subsequent keys.

You must override the Compare( ) method when you create a custom sort filter. If you fail to override this method, you get a run-time error when the sorter attempts to sort the data.

Syntax Function Compare(row1 As AcDataRow, row2 As AcDataRow) As Integer

Parameters **row1**
A reference to the first row to compare.

row2
A reference to the second row to compare.

Returns A positive number if row1 goes after row2.
0 if row1 equals row2.
A negative number if row1 goes before row2.

Example The following example compares two customers by state. If the states are identical, then Compare( ) compares the customer names.

```
Function Compare( row1 As AcDataRow, row2 As AcDataRow ) As
   Integer
   Dim Cust1 As CustomerRow
   Dim Cust2 As CustomerRow
   Set Cust1 = row1
```

```
      Set Cust2 = row2
      Compare = CompareKeys( Cust1.State, Cust2.State )
      If Compare = 0 Then
         Compare = CompareKeys( Cust1.CustName, Cust2.CustName )
      End If
   End Function
```

See also AcDataRowSorter::CompareKeys method

AcDataRowSorter::CompareKeys method

Compares two sort keys in a column. You typically call CompareKeys( ) from Compare( ), which defines the sort algorithm. For an example of using CompareKeys( ), see the example in AcDataRowSorter::Compare method.

Syntax Function CompareKeys(key1 As Variant, key2 As Variant) As Integer

Parameters **key1**
A reference to the first key to compare.

key2
A reference to the second key to compare.

Returns -1 if key1 is less than key2.
0 if key1 equals key2.
1 if key1 is more than key2.

See also AcDataRowSorter::Compare method

Class AcDataSection

An abstract base class that defines the logic sections use to process a group of data rows. Figure 7-33 shows the class hierarchy of AcDataSection.

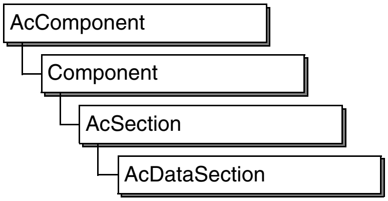


Figure 7-33 AcDataSection

Description A data section processes a group of data rows. AcDataSection is the base class for the two types of data sections, Section and AcGroupSection.

A report section defines a group as the entire set of data rows the section reads from a data stream. A report section opens a data stream and retrieves data rows from it.

A group section defines a group as a set of data rows that have the same key value, such as data rows with a state field value of CA. A group section relies on another component to provide it with data rows.

Both types of data sections process groups of data rows the same way. The processing involves the five component references that AcDataSection defines. Table 7-26 describes how the data section processes rows for components in these component references.

Table 7-26 Overview of how a data section processes component references

Component reference	Description	Process
PageHeader	Contains a frame that appears at the top of each page, except the first page	The data section keeps track of the page and flow start and end events to build the page header frame. The data section passes the current row to the page header frame when the frame is built.
Before	Contains a frame that appears before the first row in a group	The data section's Start( ) method instantiates the Before frame. The Before frame's BuildFromRow( ) method is called to process each row the section processes. The data section finishes the Before frame after the frame processes the last row in the group.

(continues)

Table 7-26 Overview of how a data section processes component references (continued)

Component reference	Description	Process
Content	Contains a section or a frame that processes each data row in a group	The data section instantiates the content component when it processes the first row. The component's BuildFromRow( ) method is called to process each row until it returns Finished Building.
After	Contains a frame that appears after the last row in a group	The data section's Start( ) method instantiates the After frame. The After frame's BuildFromRow( ) method is called to process each row the section processes. The data section finishes the After frame after the frame processes the last row in the group.
PageFooter	Contains a frame that appears at the bottom of each page, except the last	The data section keeps track of the page and flow start and end events to build the page footer frame. The data section passes the current row to the page footer frame when the frame is built.

The processes described in the preceding table explain what occurs when a data section uses running, or one-pass, aggregates, such as an After frame that calculates the total orders for a group of rows. The process changes if the section uses lookahead, or two-pass, aggregates, such as an After frame that calculates the order value for a group of rows as a percentage of all totals, across all groups. For lookahead aggregates, the data section has to process the data rows twice. The first pass calculates the aggregates. The second pass builds the contents as described in the preceding table.

Subclassing AcDataSection

Because AcDataSection is an abstract class, do not subclass AcDataSection.

Variables

Table 7-27 lists AcDataSection variables.

Table 7-27 AcDataSection variables

Variable	Type	Description
ContiguousPageFooter	Boolean	Determines whether the page footer appears directly under the last frame on the page or at the bottom of the page.
ShowFooterOnLast	Boolean	Determines whether the page footer appears on the last page. The default setting places the page footer on every page except the last.

Table 7-27 AcDataSection variables

Variable	Type	Description
ShowHeaderOnFirst	AcPage Header Options	Determines whether the page header appears on the first page. The default setting places the page header on every page except the first.

Properties

Table 7-28 lists AcDataSection properties.

Table 7-28 AcDataSection properties

Property	Type	Description
ContiguousPage Footer	Pagination	Determines whether the page footer appears directly under the last frame on the page or at the bottom of the page.
PageBreakBetween	Pagination	Determines whether the section should start each Content component except page headers and footers at the top of a new page.
ShowFooterOnLast	Pagination	Determines whether the page footer appears on the last page. The default setting displays the page footer on every page except the last.
ShowHeaderOnFirst	Pagination	Determines whether the page header appears on the first page. Valid values are: ■ AsColumnHeader. The column headers appear after the Before frame and immediately before the first set of columns. ■ AsPageHeader. The page header appears before the Before frame. ■ NoHeaderOnFirst. The page header does not appear on the first page. The default value is NoHeaderOnFirst.

Methods for Class AcDataSection

Methods defined in Class AcDataSection

GetAfter, GetBefore, GetFirstPageFooter, GetFirstPageHeader, GetPageFooter, GetPageHeader, NewAfter, NewBefore, NewContent, NewPageFooter, NewPageHeader, OnEmptyGroup

Methods inherited from Class AcSection

CommittedToFlow, DeletePageFrame, FinishConnection, FinishFlow, FinishPage, GetComponentACL, GetCurrentRow, GetSearchValue, NewPage, ObtainConnection, PageBreakAfter, PageBreakBefore, SetSearchValue, SetSecurity, StartFlow, StartPage, StopAfterCurrentFrame, StopAfterCurrentRow, StopNow, TocAddComponent, TocAddContents

Methods inherited from Class Component

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDataSection::GetAfter method

Retrieves a reference to the component in the After slot. You need a reference to the component if, for example, you want to change properties of an existing After component.

Syntax Function GetAfter( ) As Component

Returns The component in the After slot.
Nothing if there is no component in the After slot.

See also AcDataSection::GetBefore method
AcDataSection::GetPageFooter method
AcDataSection::GetPageHeader method

AcDataSection::GetBefore method

Retrieves a reference to the component in the Before slot of this report or group section. You need a reference to the component if, for example, you want to change properties of an existing Before component.

Syntax Function GetBeforeFrame( ) As Component

Returns The component in the Before slot.
Nothing if there is no component in the Before slot.

See also AcDataSection::GetAfter method
AcDataSection::GetPageFooter method
AcDataSection::GetPageHeader method

AcDataSection::GetFirstPageFooter method

Returns the page footer of the first page in a report or group section. The page footer component exists even if the section does not contain a visible page footer. GetFirstPageFooter( ) is a viewing method.

When performing a search on page footer controls, the search engine examines the instance of the page footer GetFirstPageFooter( ) returns.

Syntax Function GetFirstPageFooter( ) As AcBaseFrame

Returns The page footer component.
Nothing if the page footer does not exist in the report design.

AcDataSection::GetFirstPageHeader method

Returns the page header of the first page in a report or group section. The page header component exists even if the section does not contain a visible page header, such as when if the ShowHeaderOnFirst property is set to NoHeaderOnFirst and the section starts and ends on the same page. GetFirstPageHeader( ) method is a viewing method.

When performing a search on page header controls, the search engine examines the instance of the page header GetFirstPageHeader( ) returns.

Syntax Function GetFirstPageHeader( ) As AcBaseFrame

Returns The page header object.
Nothing if the page header does not exist in the report design.

AcDataSection::GetPageFooter method

Returns a reference to the PageFooter component for the currently active flow. You need a reference to the component if, for example, you want to change properties of an existing PageFooter component.

Syntax Function GetPageFooter( ) As AcBaseFrame

Returns The reference to the PageFooter component for the currently active flow.
Nothing if there is no PageFooter component.

See also AcDataSection::GetAfter method
AcDataSection::GetBefore method
AcDataSection::GetPageHeader method

AcDataSection::GetPageHeader method

Returns a reference to the PageHeader component in the current flow. You need a reference to the component if, for example, you want to change properties of an existing PageHeader component.

Syntax Function GetPageHeader( ) As AcBaseFrame

Returns Returns the page header for the currently active flow.
Nothing if there is no component in the PageHeader slot.

See also AcDataSection::GetAfter method
AcDataSection::GetBefore method
AcDataSection::GetPageFooter method

AcDataSection::NewAfter method

Instantiates the component in the After slot. You can override NewAfter( ) to conditionally instantiate an After component. For example, to display a different After frame depending on the value of a data row variable, you can override NewAfter( ) to write the conditional logic.

Syntax Function NewAfter( ) As Component

Returns The component in the After slot.

See also AcDataSection::NewContent method

AcDataSection::NewBefore method

Instantiates the component in the Before slot. You can override NewBefore( ) to conditionally instantiate a Before component. For example, to display a different Before frame depending on the value of a data row variable, you can override NewBefore( ) to write the conditional logic.

Syntax Function NewBefore( ) As Component

Returns The component instantiated in the Before slot.

See also AcDataSection::NewContent method

AcDataSection::NewContent method

Instantiates the component in the Content slot of the report or group section. You can override NewContent( ) to conditionally instantiate a Content component. For example, to display a different frame depending on the value of a data row variable, you can override NewContent( ) to write the conditional logic.

Syntax Function NewContent( ) As Component

Returns The component instantiated in the Content slot.

Example The following example shows how to override NewContent( ) to instantiate one of three frames depending on the type of customer. One frame is for business customers, another frame is for residential customers, and another frame is for government customers.

```
Function NewContent( ) As Component
   Dim cust As CustomerRow
   Set cust = GetCurrentRow( )

   If row Is Nothing Then
       'Creating a content for use in detecting two-pass
         aggregates
       'This report has no aggregates, so just return Nothing
      Exit Function
   End If

   Select Case cust.CustType
      Case "R"
         Set NewContent = New Persistent ResidentialCustomerFrame
      Case "S"
         Set NewContent = New Persistent BusinessCustomerFrame
      Case "G"
         Set NewContent = New Persistent GovtCustomerFrame
   End Select
End Function
```

See also AcDataSection::NewAfter method
AcDataSection::NewBefore method
AcDataSection::NewPageFooter method
AcDataSection::NewPageHeader method

AcDataSection::NewPageFooter method

Instantiates the component in the PageFooter slot. You can override NewPageFooter( ) to conditionally instantiate a PageFooter component. For example, to display a different frame depending on the value of a data row variable, you can override NewPageFooter( ) to write the conditional logic.

Syntax Function NewPageFooter( ) As AcBaseFrame

Returns The component instantiated in the PageFooter slot.

See also AcDataSection::NewContent method

AcDataSection::NewPageHeader method

Instantiates the component in the PageHeader slot of the report or group section. You can override NewPageHeader( ) to conditionally instantiate a PageHeader component. For example, to display a different frame depending on the value of a data row variable, you can override NewPageHeader( ) to write the conditional logic.

Syntax Function NewPageHeader( ) As AcBaseFrame

Returns The component instantiated in the PageHeader slot.

See also AcDataSection::NewContent method

AcDataSection::OnEmptyGroup method

The report or group section calls OnEmptyGroup( ) when the section finishes processing the current group and the group contains no data rows. Override OnEmptyGroup( ) to change this behavior. For example, you can output a custom frame to describe the case or raise an error.

Syntax Sub OnEmptyGroup( )

Class AcDataSource

A base class that defines how a data source retrieves data from an input source and creates data rows. Figure 7-34 shows the class hierarchy of AcDataSource.

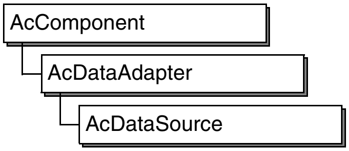


Figure 7-34 AcDataSource

Description AcDataSource is the base class for data adapters that read data from an input source such as a query, a file, or another external source. AcDataSource adds to the base data adapter class some general functionality that is useful when creating data sources.

AcDataSource defines and maintains a variable, IsAtEnd, that keeps track of a data source's state. You can set IsAtEnd to True when the report detects that the data source has read the last input row.

Subclassing AcDataSource

Create a subclass directly from AcDataSource to retrieve data from an input source that is not a database. For example, if a report uses data from a spreadsheet or a text file, you need to create a data source that can read from a spreadsheet or text file. To create a custom data source, take the following steps:

- Override Start( ) to open an input source, such as a flat file.
- Override Fetch( ) to read data rows from an input source.
- Override Finish( ) to close an input source.

Variable

Table 7-29 describes the AcDataSource variable.

Table 7-29 AcDataSource variables

Variable	Type	Description
IsAtEnd	Boolean	The status of the data source state

See also Class AcDataAdapter
Class AcDataRow
Class AcMultipleInputFilter
Class AcSingleInputFilter

Methods for Class AcDataSource

Method defined in Class AcDataSource

HasFetchedLast

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDataSource::HasFetchedLast method

Determines whether the data source has fetched the last row. HasFetchedLast( ) returns the value of the IsAtEnd variable. It is the responsibility of derived classes to ensure that this variable is set correctly. Instead of calling HasFetchedLast( ) to determine whether the data source has all the data rows, it is better to use Fetch( )'s return value. Fetch( ) returns Nothing when the last data row returns.

HasFetchLast( ) is primarily used when data sources must prevent reading past the end of their input sources.

Syntax Function HasFetchedLast( ) As Boolean

Returns True if the data source retrieved the last data row.
False if there are more data rows to retrieve.

Class AcDateTimeControl

A class that you use in the report design to display a date or time. Figure 7-35 shows the hierarchy of AcDateTimeControl.

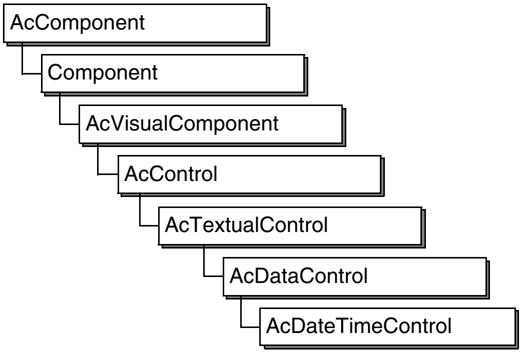


Figure 7-35 AcDateTimeControl

Description Use the DateTime control to store and display a date or time numeric value. The value you assign to the control's ValueExp property must be a date type. If, for example, you specify Date$( ) in ValueExp, you get an error message because Date$( ) returns a string. To get the current date as a date type, use Now( ) in ValueExp. Similarly, if the DateTime control gets its value from a data row column, make sure the date is a date value and not a string.

Variable

Table 7-30 describes the AcDateTimeControl variable.

Table 7-30 AcDateTimeControl variables

Variable	Type	Description
DataValue	Date	Stores the date and time value. The range is 1 January 100 to 31 December 9999 for dates. The range is 0:00:00 to 23:59:59 for times. The default value is Null.

Methods for Class AcDateTimeControl

Methods inherited from Class AcDataControl

Format, GetGroupKey, IsSummary

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class Component

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcDBConnection

A base class that defines the basic protocol for establishing database connections. Figure 7-36 shows the class hierarchy of AcDBConnection.

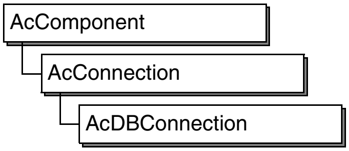


Figure 7-36 AcDBConnection

Description AcDBConnection class is the base class for the following connection classes:

- AcDB2Connection
- AcMSSQLConnection
- AcOdaConnection
- AcODBCConnection
- AcOracleConnection
- AcProgressSQL92Connection

AcDBConnection defines the basics of connecting to and disconnecting from a database, and the logic for creating the database statement object required to execute a SQL statement. For information about database statements, see Class AcDBStatement.

AcDBConnection also defines error-handling methods, such as GetGeneralError( ), GetSpecificError( ), GetGeneralErrorText( ), and GetSpecificErrorText( ). You can call these methods to display error messages when the connect or disconnect operations fail.

Properties

Table 7-31 lists AcDBConnection properties.

See also Class AcDBStatement

Methods for Class AcDBConnection

Methods defined in Class AcDBConnection

GetGeneralError, GetGeneralErrorText, GetSpecificError, GetSpecificErrorText, Prepare

Table 7-31 AcDBConnection properties

Property	Type	Description
ConfigKey	String	Specifies the run-time connection properties for a report. The value of the ConfigKey property must match the value of the Type attribute for the connection's ConnectOptions element in the configuration file. If the ConfigKey property is not set, the framework uses the fully qualified name of the connection component.
Maximum StringLength	String Variable	The maximum length of a table field. The maximum length is 32,672 characters. The default value is 8,000 characters.

Methods inherited from Class AcConnection

Connect, Disconnect, IsConnected, RaiseError

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDBConnection::GetGeneralError method

Checks for error conditions and returns a general error code. General error codes are generated by Actuate software and are used for all databases. To return error codes generated by a specific SQL database, use GetSpecificError( ).

Table 7-32 lists the general error code constants.

Table 7-32 General error code constants

DB_BadParamTypeForFunc	DB_InvalidProcedure
DB_CannotLoadDLL	DB_InvalidStatement
DB_CantConvertParameter	DB_LoginFailed
DB_CursorNotOpen	DB_MaxCursorsOnParm
DB_CursorOnSprocStmtErr	DB_MaxCursorsOnStatement
DB_DescNotAvailable	DB_NoColumnInfo
DB_EndOfLife	DB_NoCurrentConnection
DB_EndOfResults	DB_NoError
DB_FuncNotForDB	DB_NoResultSetAvailable
DB_FuncNotForDBServer	DB_NotSupportedPlatform
DB_FuncNotForDS	DB_OutOfCursors

Table 7-32 General error code constants

DB_IncompatibleClient	DB_OutOfMemory
DB_InternalError	DB_OverloadedStoredProc
DB_InvalidConnProperty	DB_ParameterNotBound
DB_Invalid_DataType	DB_Specific
DB_InvalidDescId	DB_TimeOut
DB_InvalidLogin	DB_UnauthorizedConnection
DB_InvalidColumn	DB_UnboundVariable
DB_InvalidParameter	DB_VariableDescMismatch
DB_InvalidParamId	

Syntax Function GetGeneralError( ) As Integer

Returns The error code.

See also AcDBConnection::GetGeneralErrorText method
AcDBConnection::GetSpecificError method
AcDBConnection::GetSpecificErrorText method

AcDBConnection::GetGeneralErrorText method

Checks for errors and returns a description of the error. Actuate software generates general error messages for all databases. To return an error message from a specific SQL database, use GetSpecificErrorText( ).

Syntax Function GetGeneralErrorText( ) As String

Returns The text of the Actuate error code.

See also AcDBConnection::GetGeneralError method
AcDBConnection::GetSpecificError method
AcDBConnection::GetSpecificErrorText method

AcDBConnection::GetSpecificError method

Checks for error conditions and returns an error code from a SQL database. To return general error codes from Actuate software, use GetGeneralError( ).

Syntax Function GetSpecificError( ) As Integer

Returns The error code generated by the SQL server.

See also AcDBConnection::GetGeneralError method
AcDBConnection::GetGeneralErrorText method
AcDBConnection::GetSpecificErrorText method

AcDBConnection::GetSpecificErrorText method

Checks for error conditions and returns a description of the error from the SQL server. To return general error messages from Actuate software, use GetGeneralErrorText( ).

Syntax Function GetSpecificErrorText( ) As String

Returns The text of the SQL server error code.

See also AcDBConnection::GetGeneralError method
AcDBConnection::GetGeneralErrorText method
AcDBConnection::GetSpecificError method

AcDBConnection::Prepare method

Creates and prepares a database statement object to execute a SQL statement.

If you use the standard SQL query data source to retrieve data from the database, the framework calls Prepare( ). If you create and execute your own SQL statements, you must call Prepare( ), then call Execute( ) or open a cursor on the statement.

Syntax Function Prepare(stmtText As String) As AcDBStatement

Returns The database statement that was prepared.
Nothing if there is an error in the statement.

See also AcDBStatement::Execute method
AcDBStatement::Prepare method

Class AcDB2Connection

Establishes a connection to a DB2 database. Figure 7-37 shows the class hierarchy of AcDB2Connection.

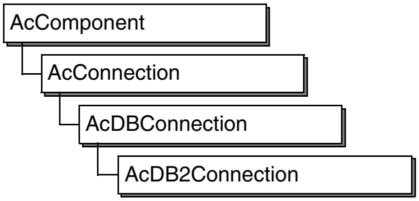


Figure 7-37 AcDB2Connection

Description Use the AcDB2Connection class to establish a connection to a DB2 database. The report must set the DLL path, user name, password, and data source prior to connecting. Once connected, the report should not change these values.

Variables

Table 7-33 lists AcDB2Connection variables.

Table 7-33 AcDB2Connection variables

Variable	Type	Description
DataSource	String	The DB2 data source
DllPath	String	The name of the DLL providing the client database
Password	String	The client password for the connection
UserName	String	The client user name for the connection

Properties

Table 7-34 lists AcDB2Connection properties.

Table 7-34 AcDB2Connection properties

Property	Type	Description
DataSource	String	The DB2 data source
DllPath	String	The name of the DLL providing the client database
Password	String	The client password for the connection
UserName	String	The client user name for the connection

About DB2 data types

Table 7-35 describes the default conversion between DB2 and Actuate data types.

Table 7-35 Default mapping of DB2 to Actuate data types

DB2 data type	Maps to
Bigint	Actuate Long. Can map to Actuate Currency, Double, Integer, Single, or String.
Binary	Actuate String.
Bit	Actuate Integer. Can also map to Actuate Double, Long, Single, or String.
Blob	Actuate String.
Char	Actuate String.
Clob	Actuate String.
Date	Actuate Date. Can also map to Actuate String.
Dbclob	Actuate String.
Decimal	Actuate Double. Can also map to Actuate Currency, Integer, Long, Single, or String.
Double	Actuate Double. Can also map to Actuate Currency, Single, or String.
Float	Actuate Double. Can also map to Actuate Currency, Single, or String.
Graphic	Actuate String.
Integer	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.
Longvarbinary	Actuate String.
Longvarchar	Actuate String.
Longvargraphic	Actuate String.
Numeric	Actuate Double. Can also map to Actuate Currency, Integer, Long, Single, or String.
Real	Actuate Single. Can also map to Actuate Currency, Double, or String.
Smallint	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.
Time	Actuate Date. Can also map to Actuate String.
Timestamp	Actuate Date. Can also map to Actuate String.
Tinyint	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.

Table 7-35 Default mapping of DB2 to Actuate data types

DB2 data type	Maps to
Type_date	Actuate Date. Can also map to Actuate String.
Type_time	Actuate Date. Can also map to Actuate String.
Type_timestamp	Actuate Date. Can also map to Actuate String.
Varbinary	Actuate String.
Varchar	Actuate String.
Vargraphic	Actuate String.

Methods for Class AcDB2Connection

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Methods inherited from Class AcConnection

Connect, Disconnect, IsConnected, RaiseError

Methods inherited from Class AcDBConnection

GetGeneralError, GetGeneralErrorText, GetSpecificError, GetSpecificErrorText, Prepare

Class AcDBCursor

Provides an Actuate Basic interface to a database cursor for a SQL statement. Figure 7-38 shows the class hierarchy of AcDBCursor.

Figure 7-38 AcDBCursor

Description A database cursor is an identifier associated with a set of data rows that a SQL query returns. The cursor manages the retrieval of rows and acts as intermediary between the data that returns and the data stream component of a report. The cursor also keeps track of the row position in the set as the database sends each row to the data source. SELECT statements that return more than one row of data require a database cursor.

When you use Actuate's SQL query data source to retrieve data, the framework executes all the necessary tasks, including creating an instance of AcDBCursor to manage row retrieval. If you write custom code to handle data retrieval from, for example, a stored procedure, you must create a connection, a database statement, and a cursor.

Use the following steps to retrieve rows using a cursor:

- Use a subclass of AcDBConnection to connect to your database.
- Prepare the statement object using the connection's Prepare( ) method. For more information about database statement objects, see Class AcDBStatement.
- Create a cursor using the statement's AllocateCursor( ) method.
- Open the cursor using the cursor's OpenCursor( ) method.
- Bind the cursor to a data row class using the cursor's BindColumn( ) method.
- Instantiate a data row to hold the first row of data.
- Call the cursor's Fetch( ) method to retrieve the first row.
- Repeat the previous two steps to retrieve each row until Fetch( ) returns False, indicating that the cursor has read all available rows.

The framework deletes the statement and cursor when they complete their tasks. Actuate software generates an error if you call the Delete( ) method to delete a cursor.

Example You can use the New( ) method to create a cursor. In the following code example, the two Set statements are equivalent:

```
Sub Example( stmt As AcDBStatement )
   Dim cursor1 As AcDBCursor
   Dim cursor2 As AcDBCursor
   Set cursor1 = stmt.AllocateCursor( )
   Set cursor2 = New AcDBCursor( stmt )
End Sub
```

You can also use the AcDBStatement::AllocateCursor method to create a cursor.

Methods for Class AcDBCursor

Methods defined in Class AcDBCursor

BindColumn, BindParameter, CloseCursor, DefineProcedureInputParameter, DefineProcedureOutputParameter, DefineProcedureReturnParameter, Delete, Fetch, GetConnection, GetOutputParameter, GetProcedureStatus, GetStatement, IsOpen, New, OpenCursor, SetProperty, StartNextSet

AcDBCursor::BindColumn method

Binds a database column to a data row variable. Use the BindColumn( ) method to specify how the framework copies column data to the data row. Call BindColumn( ) repeatedly until you have bound each column to a data row variable. All columns must be bound to variables of a single class. Figure 7-39 shows conceptually how columns are bound to variables. In the following example, you call BindColumn( ) six times.

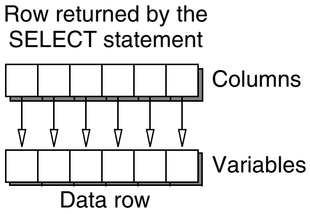


Figure 7-39 Column binding

After binding the columns to the data row variables, call Fetch( ) to retrieve each row from the database.

Syntaxes Sub BindColumn(columnID As Integer, className As String, memberName As String)

Sub BindColumn(columnName As String, className As String, memberName As String)

Parameters **columnID**
The index of the column to which to bind the data row variable. The first column has an index of 1. The index of each column is determined by its position in the SELECT clause.

columnName
The name of the column to which to bind the data row variable. The name must be the same as the column name or alias used in the SELECT clause.

className
The name of the data row class. This class is typically a subclass of AcDataRow. You can, however, bind column data to variables in any class.

memberName
The name of the variable in the data row to hold the output data.

Example In the following example, a SELECT statement gets the names of contacts whose last names are Franco. A cursor is required because the SELECT statement returns data rows. The calls to BindColumn( ) set up the association between the columns and the FirstName and LastName variables in MyRow.

```
Class MyRow
   Dim FirstName As String
   Dim LastName As String
End Class

Sub Example( connection As AcDBConnection )
   Dim stmt As AcDBStatement
   Dim cursor As AcDBCursor
   Dim row As MyRow

   ' Prepare the statement
   Set stmt = connection.Prepare( "SELECT contact_first,
      contact_last FROM Customers WHERE contact_last =
      'Franco'" )

   ' Open the cursor
   Set cursor = stmt.AllocateCursor( )
   cursor.OpenCursor( )

   ' Bind the columns to the data row variables
   cursor.BindColumn( 1, "MyRow", "FirstName" )
   cursor.BindColumn( 2, "MyRow", "LastName" )

   ' Instantiate the data row and retrieve data
   Do While True
      Set row = New MyRow
      If Not cursor.Fetch( row ) Then
         Exit Do
      End If
      ' Process the row
   Loop
End Sub
```

See also AcDBCursor::Fetch method

AcDBCursor::BindParameter method

Assigns the value of an Actuate Basic variable to a cursor parameter. You must assign a value to all the cursor parameters specified in the associated database statement text.

Syntaxes Sub BindParameter(parameterId As Integer, var As Any)

Sub BindParameter(parameterName As String, var As Any)

Parameters **parameterId**
The position of the cursor parameter. The first parameter in the statement is position 1, the second is position 2, and so on.

var
The variable with the value assigned to the parameter. Its data type should be appropriate for the parameter.

parameterName
The name of the cursor parameter.

AcDBCursor::CloseCursor method

Closes the cursor. Use CloseCursor( ) only if you need to reopen the same cursor later. The framework closes the cursor automatically when it deletes the cursor object.

Syntax Sub CloseCursor( )

See also AcDBCursor::OpenCursor method

AcDBCursor::DefineProcedureInputParameter method

Defines an input parameter used by a stored procedure. If your report accesses a stored procedure that uses only input parameters, you must call DefineProcedureInputParameter( ) for each parameter to specify the parameter name and Basic data type that matches the parameter’s type. If the parameter both accepts an input value and returns an output value, specify the input and output parameters using DefineProcedureOutputParameter( ).

Syntax Function DefineProcedureInputParameter(pname As String, val As Variant) As Boolean

Parameters **pname**
The name of the input parameter.

val
The value to pass to a stored procedure input parameter.

Returns True if the parameter is defined successfully.
A database error is raised if errors are found.

See also AcDBCursor::DefineProcedureOutputParameter method
AcDBCursor::DefineProcedureReturnParameter method

AcDBCursor::DefineProcedureOutputParameter method

Defines an input and output parameter or an output only parameter used by a stored procedure. If your report accesses a stored procedure that uses output parameters, you must call DefineProcedureOutputParameter( ) for each parameter to specify the parameter name and Basic data type that matches the parameter's type. If the parameter both accepts an input value and returns an output value, you must also specify the input value to pass.

After defining the stored procedure's output parameters and executing the stored procedure, call StartNextSet( ) to get the value of each output parameter. Output parameters with a V_CPOINTER Actuate Basic type code cannot be accessed using GetOutputParameter( ). To get a reference to the cursor, call AcDBStatement::AllocateCursor method.

Syntaxes For parameters that only return output:

```
Function DefineProcedureOutputParameter( pname As String, tcode As Integer )
    As Boolean
```

For parameters that receive input values and return output:

```
Function DefineProcedureOutputParameter( pname As String, tcode As Integer,
    val as Variant ) As Boolean
```

Parameters **pname**
The name of the output parameter.

tcode
The Actuate Basic type code that maps to the data type of the stored procedure output parameter. Valid data types are:

- V_CURRENCY
- V_DATE
- V_DOUBLE
- V_INTEGER
- V_LONG
- V_SINGLE
- V_STRING

val
The value to pass to a stored procedure output parameter that also takes input. If the corresponding Actuate Basic type code is V_CPOINTER, specify a Null value.

Returns True if parameter is defined successfully.
A database error is raised if errors are found.

See also AcDBCursor::DefineProcedureInputParameter method
AcDBCursor::DefineProcedureReturnParameter method
AcDBStatement::AllocateCursor method

AcDBCursor::DefineProcedureReturnParameter method

Specifies the data type of the return value from a stored procedure.

Syntax

```
Function DefineProcedureReturnParameter( pname As String, tcode As Integer )
    As Boolean
```

Parameters

pname
The name of the parameter that represents the return value.

tcode
The Actuate Basic type code that maps to the data type of the stored procedure return value. Valid data types are:

- V_CURRENCY
- V_DATE
- V_DOUBLE
- V_INTEGER
- V_LONG
- V_SINGLE
- V_STRING

Returns True if return value is defined successfully.
A database error is raised if there are errors.

See also AcDBCursor::DefineProcedureInputParameter method
AcDBCursor::DefineProcedureOutputParameter method

AcDBCursor::Delete method

Deletes the cursor object.

Syntax

```
Sub Delete( )
```

AcDBCursor::Fetch method

Retrieves one row from a database cursor. To retrieve all rows, execute Fetch( ) in a Do loop. When Fetch( ) finishes retrieving rows, it returns False.

Syntax Function Fetch(dataRow As AnyClass) As Boolean

Parameter **dataRow**
The data row to which to copy the data. The data row's variables should already be bound to the columns with BindColumn( ).

Returns True if a row is available.
False if there are no more rows. If you are reading from a stored procedure that returns more than one set of rows, Fetch( ) returns False at the end of each set.

See also AcDataAdapter::Fetch method
AcDBCursor::BindColumn method

AcDBCursor::GetConnection method

Returns a reference to the connection against which the cursor operates. You need this reference if, for example, you want to call the connection's error-handling methods, such as GetGeneralError( ) or GetSpecificError( ).

Syntax Function GetConnection( ) As AcDBConnection

Returns The connection this cursor uses.

AcDBCursor::GetOutputParameter method

Returns the value of a stored procedure's output parameter. You should already have defined each output parameter using DefineProcedureOutputParameter( ).

GetOutputParameter( ) returns a single value. To get rows of data, use the cursor's Fetch( ) method.

Syntaxes Function GetOutputParameter(columnName As String) As Variant

Function GetOutputParameter(columnIndex As Integer) As Variant

Parameters **columnName**
The name of the database column from which the data for the output parameter is fetched. This argument must be used for output parameters on Oracle stored procedures. You cannot use output parameters with a data type of V_CPOINTER.

columnIndex
The position of the column from which the data for the output parameter is fetched. This argument cannot be used for Oracle stored procedures.

Returns Value of the output parameter.

See also AcDBCursor::DefineProcedureOutputParameter method

AcDBCursor::GetProcedureStatus method

Returns a preset value that indicates the status of a stored procedure, if status values were previously defined. GetProcedureStatus( ) is typically used to monitor the execution and termination of the stored procedure.

Syntax Function GetProcedureStatus( ) As Integer

Returns A preset status value.

AcDBCursor::GetStatement method

Returns a reference to the database statement for which the cursor was created.

Syntax Function GetStatement( ) As AcDBStatement

Returns A reference to the statement for which the cursor was created.

AcDBCursor::IsOpen method

Determines whether the database cursor is open. The return value is useful for checking the status of the cursor before closing or reopening it, or before executing a task.

Syntax Function IsOpen( ) As Boolean

Returns True if the cursor is open.
False if the cursor is not open.

See also AcDBCursor::OpenCursor method
AcDBStatement::AllocateCursor method

AcDBCursor::New method

Constructor method for this class. You cannot call Sub New( ), the default constructor method that has no parameters. Instead, use one of the constructors that takes parameters.

Syntaxes Sub New( )

Sub New(theStatement As AcDBStatement)

Sub New(theStatement As AcDBStatement, parameterName as String)

Parameters **theStatement**
The statement to use to create the new cursor.

parameterName
The name of a parameter the statement uses.

AcDBCursor::OpenCursor method

Opens the database cursor. You can also call OpenCursor( ) to reopen a cursor previously closed with CloseCursor( ).

Another way to open a cursor is to use the statement's OpenCursor( ) method. The difference between the cursor's OpenCursor( ) method and the statement's OpenCursor( ) method is that the latter both allocates and opens a cursor.

Syntax Function OpenCursor( ) As Boolean

Example The following code illustrates the two ways to allocate and open a cursor. The code assumes the statement is created and prepared.

```
Dim cursor1 As AcDBCursor
Dim cursor2 As AcDBCursor
' Using the cursor's AllocateCursor( ) and OpenCursor( )
' methods
Set cursor1 = stmt.AllocateCursor( )
cursor1.OpenCursor( )
' Using the statement's OpenCursor( ) method
Set cursor2 = stmt.OpenCursor( )
```

Returns True if the cursor opens successfully.
False if an error occurs.

See also AcDBCursor::CloseCursor method
AcDBStatement::AllocateCursor method
AcDBStatement::OpenCursor method

AcDBCursor::SetProperty method

Sets a parameter property for a stored procedure.

Syntax Function SetProperty(parameterName As String, parameterValue As Variant) As Boolean

Parameters **parameterName**
The name of the parameter.

parameterValue
The parameter value.

Returns True if the property is set.
False if there are errors.

AcDBCursor::StartNextSet method

Starts a new set of rows in a stored procedure. If you are accessing a stored procedure that returns more than one set of rows, you typically must call StartNextSet( ) after Fetch( ) finishes retrieving a set.

StartNextSet( ) prepares the cursor to read a new set of rows. After calling StartNextSet( ), you go through another process of binding the row columns to variables of a new data row using BindColumn( ), then retrieving rows using Fetch( ).

Syntax Function StartNextSet( ) As Boolean

Returns True if there is another set of data.
False if there are no more data sets.

Example The following example shows how to use a cursor to read rows from a stored procedure that returns two sets of rows:

```
Sub Example( connection As AcDBConnection )
   Dim stmt As AcDBStatement
   Dim cursor As AcDBCursor
   Dim order As OrderRow
   Dim payment As PaymentRow
   ' Prepare the statement and open the cursor.
   Set stmt = connection.Prepare( "CustomerInfo Jones" )
   Set cursor = stmt.AllocateCursor( )
   cursor.OpenCursor( )
   ' Prepare for the first set; bind each database column
   ' to a data row variable.
   cursor.BindColumn( 1, "OrderRow", "OrderNumber" )
   ' <bind other columns>
   ' Read the first set until Fetch( ) returns False.
   Do While True
      Set order = New OrderRow
      If Not cursor.Fetch( order ) Then
         Exit Do
      End If
      ' <Process the order row>
   Loop
   ' Prepare for the second set, then bind each database column
   ' to a data row variable.
   Cursor.StartNextSet( )
   Cursor.BindColumn( 1, "PaymentRow", "PaymentDate" )
   ' <bind other columns>

   ' Read the second set until Fetch( ) returns False.
   Do While True
      Set order = New PaymentRow
      If Not cursor.Fetch( order ) Then
         Exit Do
      End If
      ' <Process the payment row>
   Loop
End Sub
```

Class AcDBStatement

A class that provides a Basic interface to a SQL statement. Figure 7-40 shows the class hierarchy of AcDBStatement.

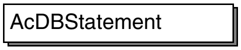


Figure 7-40 AcDBStatement

Description A database statement provides a way to execute a SQL statement. Actuate software supports two kinds of database statements. One kind of statement executes and returns no data. Examples of such statements include the SQL CREATE TABLE, INSERT, and UPDATE statements. The other kind of statement executes and returns one or more rows of data. The SELECT statement is a typical example.

A statement that returns more than one row of data requires a database cursor. A database cursor manages the retrieval of rows. For more information about database cursors, see AcDBCursor.

When you use Actuate's SQL query data source to retrieve data, the framework executes all the necessary tasks, including creating the database statement. To create and execute other SQL statements, you must create an instance of AcDBStatement.

Using a database statement

The following steps show how to create and execute a SQL statement that does not return data rows:

- Establish a connection to the database. You can use the one that a report creates for you, or you can create your own connection.
- Prepare the statement object using the connection's Prepare( ) method.
- If the SQL statement accepts parameters with values that are provided later, use BindParameter( ) to assign the value of a variable to each parameter.
- Execute the statement any number of times by calling Execute( ).

The framework deletes the statement and cursor when they complete their tasks. The framework generates an error if you call the Delete method to delete a cursor.

For information about executing a SQL statement that returns data rows, see Class AcDBCursor.

Creating a database statement

You can use the New( ) method to create a statement. You also can call the connection's Prepare( ) method. The difference between the two methods is that the connection's Prepare( ) method creates and also prepares a statement, whereas New( ) only creates the statement. If you use New( ), you must call the statement's Prepare( ) method after New( ) to prepare the statement. The following example illustrates the two ways to create and prepare a statement:

```
Sub Example( connection As AcDBConnection )
   Dim stmt1 As AcDBStatement
   Dim stmt2 As AcDBStatement

   ' Using the connection's Prepare( ) method
   Set stmt1 = connection.Prepare( "DROP TABLE MyTable" )

   ' Using the statement's New( ) and Prepare( ) methods
   Set stmt2 = New AcDBStatement( connection )
   stmt2.Prepare( "DROP TABLE MyTable" )
End Sub
```

Methods for Class AcDBStatement

Methods defined in Class AcDBStatement

AllocateCursor, BindParameter, DefineProcedureInputParameter, DefineProcedureOutputParameter, DefineProcedureReturnParameter, Delete, Execute, GetOutputCount, GetOutputParameter, GetParameterCount, GetProcedureStatus, GetStatementText, OpenCursor, Prepare

AcDBStatement::AllocateCursor method

Creates a cursor to read the rows that the statement returns. After you create the cursor, call the cursor's OpenCursor( ) method to open the cursor. This technique allows you to reuse a cursor multiple times for the same database statement.

If you are using Oracle stored procedures, use AllocateCursor( ) to create an AcDBCursor object for an Oracle cursor variable. The cursor variable is an output parameter on the Oracle stored procedure call statement.

In most cases, you can use OpenCursor( ) to allocate and open a cursor in one step.

Syntaxes Function AllocateCursor( ) As AcDBCursor

Function AllocateCursor(parameterName As String) As AcDBCursor

Parameter **parameterName**
The name of a cursor variable parameter specified in the Oracle stored procedure. ParameterName must be enclosed in quotation marks (").

Returns The database cursor that was created.

Example The example shows how to allocate and then open a cursor on an Oracle database accessed using a stored procedure. EmpCursor is the name of the cursor parameter on the Oracle stored procedure Call statement.

```
Dim theEmpCursor As AcDBCursor
' Using the cursor’s AllocateCursor( ) and OpenCursor( )
' methods
Set theEmpCursor = stmt.AllocateCursor( "EmpCursor" )
If theEmpCursor Is Nothing Then
   If Not theEmpCursor.OpenCursor( ) Then
      GetDBConnection( ).RaiseError( )
      Exit Function
   EndIf
End If
```

See also AcDBCursor::OpenCursor method
AcDBStatement::OpenCursor method

AcDBStatement::BindParameter method

Assigns the value of an Actuate Basic variable to a cursor parameter. You must assign a value to all the cursor parameters specified in the database statement.

Before calling BindParameter( ), you must already have created the statement using Prepare( ).

Syntax Sub BindParameter(parameterId As Variant, var As Any)

Parameters

parameterId
The position of the parameter to bind. The first parameter in the statement is position 1, the second is position 2, and so on.

var
The variable with the value assigned to the parameter. Its data type should be appropriate for the parameter.

Example The following example shows how to execute a parameterized INSERT statement twice, each time using a different set of parameter values. The example assumes you have established a connection to the database.

Note that you assign values to the local variables, then bind each parameter to the corresponding local variable. To execute the statement again with different values, you must assign the new values to the local variables, and again bind each parameter to the corresponding local variable.

```
Sub AnExample( connection As AcDBConnection )
   Dim statement As AcDBStatement
   Dim val1 As Integer
   Dim val2 As Integer
   ' Prepare the statement with two parameters, :val1 and :val2
   Set statement = connection.Prepare( "INSERT INTO MyTable
      (col1, col2) VALUES (:val1, :val2)" )
   If statement Is Nothing Then
      MsgBox "Failed to prepare the statement"
      MsgBox connection.GetSpecificErrorText( )
      Exit Sub
   End If

   ' Assign values to the variables.
   val1 = 100
   val2 = 200

   ' Bind each parameter to a variable.
   Statement.BindParameter( 1, val1 )
   Statement.BindParameter( 2, val2 )

   ' Execute the statements.
   If Not statement.Execute( ) Then
      MsgBox "Failed to insert data"
      MsgBox connection.GetSpecificErrorText( )
      Exit Sub
   End If

   ' Execute the statement again with different parameter
   ' values.
   val1 = 500
   val2 = 600
   ' Again, bind each parameter to a variable.
   Statement.BindParameter( 1, val1 )
   Statement.BindParameter( 2, val2 )

   If Not statement.Execute( ) Then
      MsgBox "Failed to insert data"
      MsgBox connection.GetSpecificErrorText( )
      Exit Sub
   End If
   ' When this function exits, the framework deletes the
   ' statement, freeing the statement resources.
End Sub
```

See also AcDBStatement::Prepare method

AcDBStatement::DefineProcedureInputParameter method

Defines parameter information for an input parameter used by a stored procedure. If your report accesses a stored procedure that uses only input parameters, you must call the DefineProcedureInputParameter( ) method for each parameter to specify the parameter name and Basic data type that matches the parameter’s type. If the parameter both accepts an input value and returns an output value, specify the input and output parameters using the method, AcDBStatement::DefineProcedureOutputParameter method.

Syntax

Function DefineProcedureInputParameter(pname As String, value as Variant) As Boolean

Parameters

pname
The name of the input parameter.

value
The value to pass to the stored procedure.

Returns

True if input parameter is defined successfully.
A database error is raised if errors are found.

See also

AcDBStatement::DefineProcedureOutputParameter method

AcDBStatement::DefineProcedureOutputParameter method

Provides parameter information for an output parameter used by a stored procedure. If your report accesses a stored procedure that uses output parameters, you must call the DefineProcedureOutputParameter( ) method for each parameter to specify the parameter name and Actuate Basic data type that matches the parameter's type. If the parameter both accepts an input value and returns an output value, you must additionally specify the input value to pass.

After defining the stored procedure's output parameters and executing the stored procedure, call GetOutputParameter( ) to get the value of each output parameter. Output parameters with a V_CPOINTER data type cannot be accessed using GetOutputParameter( ). To get a reference to the cursor, call the AcDBStatement::AllocateCursor method.

Syntaxes

For parameters that only return output:

Function DefineProcedureOutputParameter(pname As String, tcode As Integer) As Boolean

For parameters that receive input values and return output:

Function DefineProcedureOutputParameter(pname As String, tcode As Integer, val as Variant) As Boolean

Parameters

pname
The name of the output parameter.

tcode
The Actuate Basic type code that maps to the data type of the stored procedure input or output parameter. Valid data types are:

- V_CPOINTER
- V_CURRENCY
- V_DATE

- V_DOUBLE
- V_INTEGER
- V_LONG
- V_SINGLE
- V_STRING

val
The value to pass to a stored procedure output parameter that also takes input.

Returns True if input parameter is defined successfully.
A database error is raised if errors are found.

Example The following statement shows how to define an output parameter. Stmt contains a reference to the database statement:

```
stmt.DefineProcedureOutputParameter( "deptAcct", V_INTEGER )
```

The following statement shows how to declare an input and output parameter on an Oracle stored procedure as a cursor variable:

```
stmt.DefineProcedureOutputParameter( "empCursor", V_CPOINTER,
   NULL )
```

See also AcDBStatement::DefineProcedureInputParameter method
AcDBStatement::GetOutputParameter method

AcDBStatement::DefineProcedureReturnParameter method

Specifies the data type of a return parameter.

Syntax Function DefineProcedureReturnParameter(pname As String, tcode As Integer) As Boolean

Parameters **pname**
The name of the return parameter.

tcode
The Actuate Basic type code that maps to the data type of the stored procedure return parameter. Valid data types are:

- V_CPOINTER
- V_CURRENCY
- V_DATE, V_DOUBLE
- V_INTEGER
- V_LONG

- V_SINGLE
- V_STRING

Returns True if the return parameter is defined successfully.
A database error is raised if errors are found.

Example The following statement shows how to define a cursor variable as a return parameter for an Oracle stored procedure:

```
stmt.DefineProcedureReturnParameter( "mgrCursor", V_CPOINTER )
```

See also AcDBStatement::DefineProcedureInputParameter method
AcDBStatement::DefineProcedureOutputParameter method

AcDBStatement::Delete method

The destructor method.

Syntax Sub Delete( )

AcDBStatement::Execute method

Executes the SQL statement that does not return data. Examples of such SQL statements include CREATE TABLE, INSERT, and UPDATE. When executing statements that do not return data, do not call AllocateCursor( ) and OpenCursor( ) after calling Execute( ). To execute a SQL statement, such as SELECT, that returns data, you must use a cursor instead of Execute( ). For information about creating and using a cursor, see Class AcDBCursor.

Typically, you call Execute( ) for each SQL statement you created and prepared with the connection's Prepare( ) method. You can, however, call Execute( ) any number of times for the same statement if the statement contains parameters. For example, if you want to insert 15 rows into a database, you can use one of two techniques:

- Create 15 INSERT statements using Prepare( ), then call Execute( ) to execute each statement. The following code snippet creates and executes two INSERT statements:

  ```
  Set stmt1 = connection.Prepare( "INSERT INTO MyTable
     (fName, lName) VALUES ("John", "Smith")" )
  stmt1.Execute( )

  Set stmt2 = connection.Prepare( "INSERT INTO MyTable
     (fName, lName) VALUES ("Nancy", "Alvarez")" )
  stmt2.Execute( )
  ```

- Create one INSERT statement with parameters using Prepare( ), bind each parameter to a variable, then call Execute( ) 15 times after assigning different values to the variables. The following code snippet creates one INSERT statement and executes it twice, each time with different parameter values:

```
Dim firstName As String
Dim lastName As String
Set stmt = connection.Prepare( "INSERT INTO MyTable
   (fName, lName) VALUES (:param1, :param2)" )

stmt.BindParameter( 1, firstName )
stmt.BindParameter( 2, lastName )

firstName = "John"
lastName = "Smith"
stmt.Execute( )

firstName = "Nancy"
lastName = "Alvarez"
stmt.Execute( )
```

Both techniques achieve the same results. The second technique, however, executes more efficiently.

Syntax Function Execute( ) As Boolean

Returns True if the statement executed successfully.
False if an error occurred. You can call the error-handling methods on AcDBConnection to return the error that occurred.

See also AcDBConnection::Prepare method
AcDBStatement::BindParameter method

AcDBStatement::GetOutputCount method

Returns the number of columns. Use the GetOutputCount( ) method in conditional code that requires the number of columns in the rows returned by the SQL statement. GetOutputCount( ) is also useful if you prepared a SQL statement that returns all columns in a table, such as SELECT * FROM Customers, and you need to know how many columns will be returned. You can call GetOutputCount( ) any time after the statement is prepared with Prepare( ).

Syntax Function GetOutputCount( ) As Integer

Returns The number of columns in the rows that the SQL statement returns.

AcDBStatement::GetOutputParameter method

Returns the value of a stored procedure's output parameter. You should already have defined each output parameter using DefineProcedureOutputParameter( ).

GetOutputParameter( ) returns a single value. To get rows of data, use the cursor's Fetch method.

Syntaxes Function GetOutputParameter(columnName As String) As Variant

Function GetOutputParameter(columnIndex As Integer) As Variant

Parameters **columnName**

The name of the database column from which the data for the output parameter is fetched. Output parameters with a data type of V_CPOINTER cannot be used.

columnIndex

The position of the column from which the data for the output parameter is fetched. This argument cannot be used for Oracle stored procedures.

Returns Value of the output parameter.

Example The procedure in the following example executes an Oracle stored procedure that returns two output values. DefineProcedureOutputParameter( ) is called to define the name and type of each output parameter. After executing the stored procedure with Execute( ), GetOutputParameter( ) is called to return the value of each output parameter.

```
Sub GetSPValues( connection As AcDBConnection )
   Dim stmt As AcDBStatement
   Dim id As Long
   Dim newId As Long
   Dim name As Variant
   ' Prepare the statement to execute the OracleProc stored
   ' procedure
   Set stmt = connection.Prepare( "BEGIN OracleProc (:id,
      :name); END;" )
   If stmt Is Nothing Then
      Print #1 "Failed to prepare statement"
      Print #1 connection.GetSpecificErrorText( )
      Exit Sub
   End If

   ' Define the first parameter that is both an input and output
   ' parameter
   id = 20
   If stmt.DefineProcedureOutputParameter( "id", V_INTEGER, id )
      = 0 Then
      Print #1 "Failed to define input/output parameter"
      Print #1 connection.GetSpecificErrorText( )
      Exit Sub
   End If
   ' Define the second output parameter
   If stmt.DefineProcedureOutputParameter( "name", V_STRING ) =
      0 Then
      Print #1 "Failed to define output parameter"
      Print #1 connection.GetSpecificErrorText( )
      Exit Sub
   End If

   ' Execute the stored procedure
   If stmt.Execute( ) = 0 Then
      Print #1 "Failed to execute OracleProc stored procedure"
      Print #1 connection.GetSpecificErrorText( )
   Else
      Print #1 "OracleProc executed"
   End If

   ' Get the values of the output parameters and write the
   ' information to a file
   name = stmt.GetOutputParameter( "name" )
   newId = stmt.GetOutputParameter( "id" )

   Print #1, "Output: name = ", name
   Print #1, "Output: id = ", newId
End Sub
```

See also AcDBStatement::DefineProcedureOutputParameter method
AcDBStatement::Execute method

AcDBStatement::GetParameterCount method

Returns the number of parameters used in the SQL statement. Use GetParameterCount( ) in conditional code that requires the number of parameters used in the SQL statement. You can call GetParameterCount( ) any time after the statement is prepared.

Syntax Function GetParameterCount( ) As Integer

Returns The number of parameters in the SQL statement.

See also AcDBStatement::BindParameter method

AcDBStatement::GetProcedureStatus method

Returns the return value or status from a stored procedure. GetProcedureStatus( ) is typically used to monitor the proper execution and termination of the stored procedure.

Syntax Function GetProcedureStatus( ) As Variant

Returns A return or status value from a stored procedure.

AcDBStatement::GetStatementText method

Returns the text of the prepared SQL statement. You can call GetStatementText( ) to check that the prepared statement is in the form you intended. You can call GetStatementText( ) any time after the statement is prepared.

Syntax Function GetStatementText( ) As String

Returns The text of the prepared SQL statement.

See also AcDBConnection::Prepare method
AcDBStatement::Prepare method

AcDBStatement::OpenCursor method

Allocates and opens a database cursor.

Another way to allocate and open a cursor is to use the statement's AllocateCursor method followed by the cursor's OpenCursor method. If you need to reuse the cursor multiple times on the same database, or if you are using Oracle stored procedures, you must call AllocateCursor( ) before calling OpenCursor( ).

Syntax Function OpenCursor( ) As AcDBCursor

Example The following example illustrates two ways to allocate and open a cursor. The example assumes the statement is created and prepared.

```
Dim cursor1 As AcDBCursor
Dim cursor2 As AcDBCursor

' Using the statement's OpenCursor( ) method
Set cursor2 = stmt.OpenCursor( )

' Using the cursor's AllocateCursor( ) and OpenCursor( )
' methods
Set cursor1 = stmt.AllocateCursor( )
cursor.OpenCursor( )
```

Returns The instantiated database cursor.

See also AcDBCursor::OpenCursor method
AcDBStatement::AllocateCursor method

AcDBStatement::Prepare method

Prepares a SQL statement for execution. Call the Prepare( ) method to initialize the statement object to work with the SQL statement you provide. Before calling Prepare( ), you must instantiate the statement with New( ).

Another way to create and prepare a statement is to use the connection's Prepare( ) method. The following example illustrates the two ways to create and prepare a statement:

```
Sub Example( connection As AcDBConnection )
   Dim stmt1 As AcDBStatement
   Dim stmt2 As AcDBStatement
   ' Using the statement’s New( ) and Prepare( ) methods
   Set stmt2 = New AcDBStatement( connection )
   stmt2.Prepare( "DROP TABLE MyTable" )

   ' Using the connection’s Prepare( ) method
   Set stmt1 = connection.Prepare( "DROP TABLE MyTable" )
End Sub
```

Syntax Function Prepare(statement As String) As Boolean

Parameters **statement**
The text of the SQL statement.

Returns True if the statement executed successfully.
False if an error occurred.

See also AcDBConnection::Prepare method

Class AcDoubleControl

Displays a Double value in a report. Figure 7-41 shows the class hierarchy of AcDoubleControl.

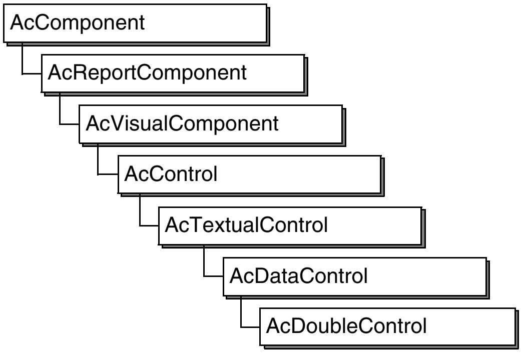


Figure 7-41 AcDoubleControl

Description Use the AcDoubleControl class to display a Double value. You can also use a currency control or integer control to display numeric values.

See also Class AcControl
Class AcCurrencyControl
Class AcDataControl
Class AcIntegerControl
Class AcTextualControl

Variable

Table 7-36 describes the variable for AcDoubleControl.

Table 7-36 AcDoubleControl variable

Variable	Type	Description
DataValue	Double	Stores the value of the control

Methods for Class AcDoubleControl

Methods inherited from Class AcDataControl

Format, GetGroupKey, IsSummary

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcDrawing

A control that displays a drawing. Figure 7-42 shows the class hierarchy of AcDrawing.

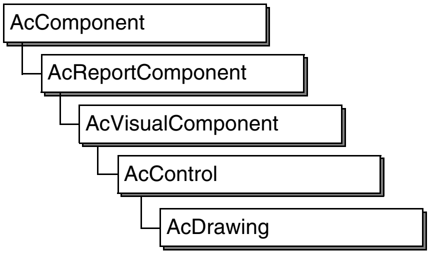


Figure 7-42 AcDrawing

Description Use AcDrawing to display a dynamically created image that scales smoothly on screen and in print.

A drawing contains zero or more drawing planes, represented by AcDrawingPlane objects. To define the contents of a drawing, you must override one or more of AcDrawing's methods to create and populate drawing planes.

AcDrawing is the parent class of AcChart.

Example In the following example, a drawing's Finish( ) method has been overridden to draw a rectangle in an SVG drawing plane:

```
Sub Finish( )
   ' Get the size of the drawing in points
   Dim w As Double
   w = Size.Width / OnePoint
   Dim h As Double
   h = Size.Height / OnePoint
   Dim svg As String
   ' Scale the drawing to use points as the default units
   svg = "<svg viewBox='0 0 " & SVGDbl( w ) & " " & SVGDbl( h )
+     & "'preserveAspectRatio='none'>"

   ' Draw a rectangle with a 2pt border
   svg = svg
+     & "<rect x='10%' y='10%' width='80%' height='80%'"
+     & " fill='red' stroke='black' stroke-width='2'/>"
+     & "</svg>"

   ' Create an SVG drawing plane
   Dim svgPlane As AcDrawingSVGPlane
   Set svgPlane = AddDrawingPlane( DrawingPlaneTypeSVG )
   svgPlane.SetSVG( svg )
   Super::Finish( )
End Sub
```

See also Class AcChart
Class AcDrawingPlane

Variables

Table 7-37 describes the AcDrawing variable.

Table 7-37 AcDrawing variables

Variable	Type	Description
Antialias	Boolean	Antialias property
BackgroundColor	AcColor	BackgroundColor property
RenderIn24BitColor	Boolean	RenderIn24BitColor property

Properties

Table 7-38 lists AcDrawing properties.

Table 7-38 AcDrawing properties

Property	Type	Description
Antialias	Boolean	Specifies whether the drawing will be rendered with antialiasing. Antialiasing improves the appearance of diagonal and curved lines, but increases the cost of rendering a drawing. The default value is False.
BackgroundColor	AcColor	The background color of the drawing. The default value is Transparent.
DesignTimeSVG	String	SVG code used to draw a sample image in e.Report Designer Professional at design time. The default value is "".
RenderIn24BitColor	Boolean	Specifies whether the drawing will be rendered as a 24-bit color bitmap, using 8 bits per color. If this property's value is False, the drawing will be rendered as an indexed color bitmap using a palette of 256 colors. 24-bit color often produces smoother gradient fills. It may also improve performance when using antialiasing. However, 24-bit color images may be considerably larger than indexed color images. The default value is False.

(continues)

Table 7-38 AcDrawing properties (continued)

Property	Type	Description
Volatile	Boolean	Specifies whether the drawing component defines a drawing that is the same in every instance. If False, the drawing component defines a drawing that is always the same in every instance, such as rotated text. If True, the drawing component defines a set of different drawings based on data. The default is True. This property is hidden in charts. AcDrawing::Volatile is used by the PDF Writer only. Volatile has no effect on any other behavior.

Methods for Class AcDrawing

Methods defined in Class AcDrawing

AddDrawingPlane, GetAntialias, GetBackgroundColor, GetDrawingPlane, GetNumberOfDrawingPlanes, GetRenderIn24BitColor, InsertDrawingPlane, RemoveDrawingPlane, RenderToFile, SetAntialias, SetRenderIn24BitColor

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDrawing::AddDrawingPlane method

Call this method to add a drawing plane to the end of a drawing's list of drawing planes. Drawing planes within a drawing are rendered sequentially so that a drawing plane whose index is 2 is rendered in front of a drawing plane whose index is 1.

Syntax Function AddDrawingPlane(drawingPlaneType As AcDrawingPlaneType)
As AcDrawingPlane

Parameters **drawingPlaneType**
The type of drawing plane to create. The valid value is DrawingPlaneTypeSVG.

Returns A handle to the new drawing plane object.

Example In the following example, a chart's DrawOnChart( ) method has been overridden to add some translucent text in front of the chart:

```
Sub DrawOnChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
  ' Get the size of the drawing in points
  Dim w As Double
  w = Size.Width / OnePoint
  Dim h As Double
  h = Size.Height / OnePoint

  ' Create SVG to draw some translucent text
  Dim svg As String
  svg = "<svg version='1.1'"
+ ' Standard SVG 1.1 namespaces
+ & " xmlns='http://www.w3.org/2000/svg'"
+ & " xmlns:xlink='http://www.w3.org/1999/xlink'"
+ ' Do not collapse whitespace in text
+ & " xml:space='preserve'"
+ ' Scale the SVG to use points as the default units
+ & " viewBox='0 0 " & SVGDbl( w ) & " " & SVGDbl( h ) & "'>"

  ' Define the font style
  Dim sampleFont As AcFont
  sampleFont.Bold = True
  sampleFont.Color = Red
  sampleFont.FaceName = "Arial"
  sampleFont.Size = 80
  svg = svg + & "<defs>"
+ & SVGFontStyle( "Sample", sampleFont )
+ & "</defs>"

  ' Draw the text
  svg = svg + & "<text class='Sample'"
+ & " transform='translate(60,250) rotate(-30)'"
+ & " fill-opacity='0.35'>"
+ & SVGStr( "SAMPLE" )
+ & "</text>"
+ & "</svg>"

  ' Add the text in front of the chart
  Dim svgPlane As AcDrawingSVGPlane
  Set svgPlane = AddDrawingPlane( DrawingPlaneTypeSVG )
  svgPlane.SetSVG( svg )
End Sub
```

See also Class AcDrawingPlane
AcChart::DrawOnChart method
AcDrawing::GetDrawingPlane method
AcDrawing::InsertDrawingPlane method
AcDrawing::RemoveDrawingPlane method

AcDrawing::GetAntialias method

Determines whether a drawing will be rendered with antialiasing.

Syntax Function GetAntialias( ) As Boolean

Returns True if the drawing will be rendered with antialiasing.
False if the drawing will be rendered without antialiasing.

See also AcDrawing::SetAntialias method

AcDrawing::GetBackgroundColor method

Returns the background color of a drawing. The background color always fills the entire area of the drawing, regardless of the positions and sizes of individual drawing planes within the drawing.

Syntax Function GetBackgroundColor( ) As AcColor

Returns The background color of the drawing.

AcDrawing::GetDrawingPlane method

Returns a reference to the specified drawing plane within a drawing. To determine the number of drawing planes in a drawing, call the drawing's GetNumberOfDrawingPlanes( ) method.

Syntax Function GetDrawingPlane(index As Integer) As AcDrawingPlane

Parameter **index**
An index into the drawing's list of drawing planes. The first drawing plane is index 1.

Returns A reference to the specified drawing plane within the drawing.

See also Class AcDrawingPlane
AcDrawing::AddDrawingPlane method
AcDrawing::GetNumberOfDrawingPlanes method
AcDrawing::InsertDrawingPlane method
AcDrawing::RemoveDrawingPlane method

AcDrawing::GetNumberOfDrawingPlanes method

Determines the number of drawing planes in a drawing.

Syntax Function GetNumberOfDrawingPlanes( ) As Integer

Returns The number of drawing planes in the drawing.

See also AcDrawing::AddDrawingPlane method
AcDrawing::GetDrawingPlane method
AcDrawing::InsertDrawingPlane method
AcDrawing::RemoveDrawingPlane method

AcDrawing::GetRenderIn24BitColor method

Determines whether a drawing will be rendered in 24-bit color. Note that not all image formats support 24-bit color. If a drawing is rendered to an image format that does not support 24-bit color, this setting will be ignored.

Syntax Function GetRenderIn24BitColor( ) As Boolean

Returns True if the drawing will be rendered in 24-bit color.
False if the drawing will not be rendered in 24-bit color.

See also AcDrawing::SetRenderIn24BitColor method

AcDrawing::InsertDrawingPlane method

Call this method to insert a drawing plane at a specific position within a drawing's list of drawing planes. Drawing planes within a drawing are rendered sequentially so that a drawing plane whose index is 2 is rendered in front of a drawing plane whose index is 1.

When you insert a new drawing plane, the original drawing plane at the insertion point and all the drawing planes above the insertion point move up one place.

Syntax Function InsertDrawingPlane(index As Integer,
  drawingPlaneType As AcDrawingPlaneType) As AcDrawingPlane

Parameters **index**
The position in the drawing's list of drawing planes at which the new drawing plane will be inserted. The first drawing plane is index 1.

Must be greater than or equal to one. Must be less than or equal to the current number of drawing planes in the drawing plus one.

drawingPlaneType
The type of drawing plane to create. The valid value is DrawingPlaneTypeSVG.

Returns A handle to the new drawing plane object.

Example In the following example, a chart's DrawOnChart( ) method has been overridden to draw a filled rectangle with rounded corners behind the chart drawing plane:

```
Sub DrawOnChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
  ' Get the size of the drawing in points
  Dim w As Double
  w = Size.Width / OnePoint
  Dim h As Double
  h = Size.Height / OnePoint

  Dim svg As String
  svg = "<svg version='1.1'"
+ ' Standard SVG 1.1 namespaces
+ & " xmlns='http://www.w3.org/2000/svg'"
+ & " xmlns:xlink='http://www.w3.org/1999/xlink'"
+ ' Do not collapse whitespace in text
+ & " xml:space='preserve'"
+ ' Scale the SVG to use points as the default units
+ & " viewBox='0 0 " & SVGDbl( w ) & " " & SVGDbl( h ) & "'>"

  ' Draw the background rectangle
  svg = svg
+   & "<rect"
+   & SVGColorAttr( "fill", RGB( 255, 255, 204 ) )
+   & SVGColorAttr( "stroke", Black )
+   & SVGAttr( "stroke-width", 3.0 )
+   & SVGAttr( "x", 1.5 )
+   & SVGAttr( "y", 1.5 )
+   & SVGAttr( "width", w - 3.0 )
+   & SVGAttr( "height", h - 3.0 )
+   & SVGAttr( "rx", 9.0 )
+   & "/>"
+   & "</svg>"

  ' Insert the background rectangle behind the chart
  Dim svgPlane As AcDrawingSVGPlane
  Set svgPlane = InsertDrawingPlane( 1, DrawingPlaneTypeSVG )
  svgPlane.SetSVG( svg )
End Sub
```

See also Class AcDrawingPlane
AcChart::DrawOnChart method
AcDrawing::AddDrawingPlane method
AcDrawing::GetDrawingPlane method
AcDrawing::RemoveDrawingPlane method

AcDrawing::RemoveDrawingPlane method

Call this method to remove a drawing plane from a drawing. When you remove a drawing plane from a drawing, all the drawing planes above that move down one place.

You cannot remove a chart's own chart drawing plane. To hide a drawing plane without removing it from a drawing, use the AcDrawingPlane::SetHidden( ) method.

Syntax Sub RemoveDrawingPlane(index As Integer)

Parameters **index**
The position in the drawing's list of drawing planes from the drawing plane will be removed. The first drawing plane is index 1.

Must be greater than or equal to one. Must be less than or equal to the current number of drawing planes in the drawing.

drawingPlaneType
The type of drawing plane to create. The only value allowed is DrawingPlaneTypeSVG.

Returns A handle to the new drawing plane object.

See also Class AcDrawingPlane
AcDrawing::AddDrawingPlane method
AcDrawing::GetDrawingPlane method
AcDrawing::InsertDrawingPlane method
AcDrawingPlane::SetHidden method

AcDrawing::RenderToFile method

Call this method to render a drawing into a file. The recommended place from which to call this method is a drawing's Finish( ) method.

Note that rendering drawings with high dpi or scale values may result in high CPU usage and very long rendering times.

Syntaxes Sub RenderToFile(fileName As String)

Sub RenderToFile(fileName As String, dpi As Double, scale As Double)

Parameters **fileName**
The name of the file into which the drawing will be rendered. The file extension is used to determine the image format. The following image formats are supported:

- BMP
- GIF
- PNG

The recommended image format is PNG. GIF is larger than PNG and cannot support 24-bit color. BMP is much larger than both GIF and PNG.

dpi
The number of dots per inch at which the drawing will be rendered. Must be in the range 72 through 768. If this parameter is omitted, a value of 96 will be used.

The following values are recommended to ensure accurate alignment when drawing onto charts:

- 96
- 192
- 288
- 384
- 576
- 768

scale
The scale at which the drawing will be rendered. Must be in the range 0.25 through 4. If this parameter is omitted, a value of 1.0 will be used.

Example In the following example, a drawing's Finish( ) method has been overridden to render the drawing into a file:

```
Sub Finish( )
   Super::Finish( )
   RenderToFile( "C:\Temp\Drawing Test.png", 96.0, 1.5 )
End Sub
```

AcDrawing::SetAntialias method

Call this method to specify whether a drawing will be rendered with antialiasing.

Syntax Sub SetAntialias(antialias As Boolean)

Parameter **antialias**
True causes the drawing to be rendered with antialiasing. False causes the drawing to be rendered without antialiasing.

See also AcDrawing::GetAntialias method

AcDrawing::SetRenderIn24BitColor method

Call this method to specify whether a drawing will be rendered in 24-bit color. Note that not all image formats support 24-bit color. If a drawing is rendered to an image format that does not support 24-bit color, this setting will be ignored.

Syntax Sub SetRenderIn24BitColor(renderIn24BitColor As Boolean)

Parameter **renderIn24BitColor**
True causes the drawing to be rendered in 24-bit color. False causes the drawing not to be rendered in 24-bit color.

See also AcDrawing::GetRenderIn24BitColor method

Class AcDrawingChartPlane

A chart drawing plane within a drawing. Figure 7-43 shows the class hierarchy of AcDrawingChartPlane.

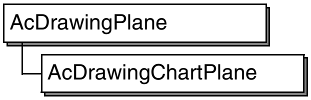


Figure 7-43 AcDrawingChartPlane

Description AcDrawingChartPlane represents a drawing plane whose contents are a chart. You cannot create AcDrawingChartPlane objects. A single AcDrawingPlaneChart objects is created automatically by each chart. To get a handle to a chart's chart drawing plane, use the AcChart::GetChartDrawingPlane( ) method.

Example In the following example, a chart's DrawOnChart( ) method has been overridden to move the chart to the right, reduce its width so that it still fits within the drawing, and add a rotated title at the left side:

```
Sub DrawOnChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
  ' Move the chart drawing plane to the right and resize it
  Dim chartPlane As AcDrawingChartPlane
  Set chartPlane = GetChartDrawingPlane( )
  Dim offset As AcTwips
  offset = 48 * OnePoint
  chartPlane.SetSize( Size.Width - (offset), Size.Height )
  chartPlane.SetPosition( offset, 0 )

  ' Get the size of the drawing in points
  Dim w As Double
  w = Size.Width / OnePoint
  Dim h As Double
  h = Size.Height / OnePoint

  Dim svg As String
  svg = "<svg version='1.1'"
+ ' Standard SVG 1.1 namespaces
+ & " xmlns='http://www.w3.org/2000/svg'"
+ & " xmlns:xlink='http://www.w3.org/1999/xlink'"
+ ' Do not collapse whitespace in text
+ & " xml:space='preserve'"
+ ' Scale the SVG to use points as the default units
+ & " viewBox='0 0 " & SVGDbl( w ) & " " & SVGDbl( h ) & "'>"

  ' Define the font style
  Dim titleFont As AcFont
  titleFont.Bold = True
  titleFont.Color = Red
  titleFont.FaceName = "Arial"
  titleFont.Size = 56
  svg = svg
+ & "<defs>"
+ & SVGFontStyle( "Title", titleFont )
+ & "</defs>"

  ' Draw the text
  svg = svg
+ & "<text class='Title'"
+ & " transform='translate(12,12) rotate(90)'>"
+ & SVGStr( "My Chart" )
+ & "</text>"
+ & "</svg>"

  ' Add the text in front of the chart
  Dim svgPlane As AcDrawingSVGPlane
  Set svgPlane = AddDrawingPlane( DrawingPlaneTypeSVG )
  svgPlane.SetSVG( svg )
End Sub
```

See also Class AcChart
Class AcDrawing
Class AcDrawingPlane
Class AcDrawingSVGPlane
AcChart::GetChartDrawingPlane method

Methods for Class AcDrawingChartPlane

Methods inherited from Class AcDrawingPlane

GetDrawingPlaneType, IsHidden, SetHidden, SetPosition, SetSize

Class AcDrawingPlane

Abstract class that represents a single drawing plane within a drawing. Figure 7-44 shows the class hierarchy of AcDrawingPlane.

Figure 7-44 AcDrawingPlane

Description AcDrawingPlane represents a drawing plane within a drawing. Use drawing planes to define drawing elements such as lines, rectangles, and text.

Drawing planes within a drawing are rendered sequentially so that a drawing plane whose index is 2 is rendered in front of a drawing plane whose index is 1. You can set the size and position of each drawing plane within a drawing independently.

AcDrawingPlane is an abstract class. You cannot create and use objects of this class. Instead, use concrete subclasses of AcDrawingPlane, such as AcDrawingSVGPlane.

Example For an example of how to work with a drawing plane, see the example for the AcDrawing class.

See also Class AcDrawing
Class AcDrawingChartPlane
Class AcDrawingSVGPlane

Methods for Class AcDrawingPlane

Methods defined in Class AcDrawingPlane

GetDrawingPlaneType, IsHidden, SetHidden, SetPosition, SetSize

AcDrawingPlane::GetDrawingPlaneType method

Returns the drawing plane type of a drawing plane. A drawing plane’s drawing plane type is set when the drawing plane is created and cannot be changed.

Syntax Function GetDrawingPlaneType( ) As AcDrawingPlaneType

Returns The drawing plane type of the drawing plane.

See also AcDrawing::AddDrawingPlane method
AcDrawing::InsertDrawingPlane method

AcDrawingPlane::IsHidden method

Determines whether a drawing plane is hidden. A hidden drawing plane is ignored when its parent drawing is rendered.

Syntax Function IsHidden( ) As Boolean

Returns True if the drawing plane is hidden.
False if the drawing plane is not hidden.

See also AcDrawingPlane::SetHidden method

AcDrawingPlane::SetHidden method

Call this method to specify whether a drawing plane is hidden. A hidden drawing plane is ignored when its parent drawing is rendered.

Syntax Sub SetHidden(hidden As Boolean)

Parameter **hidden**
True causes the drawing plane to be ignored when its parent drawing is rendered. False causes the drawing plane to be rendered when its parent drawing is rendered.

Example In the following example, a chart's DrawOnChart( ) method has been overridden to replace the chart with the words "No Data!" if the chart has no data points:

```
Sub DrawOnChart( baseLayer As AcChartLayer,
+ overlayLayer As AcChartLayer, studyLayers() As AcChartLayer )
  ' Check for empty chart
  Dim hasData As Boolean
  Dim numberOfSeries As Integer
  numberOfSeries = baseLayer.GetNumberOfSeries( )
  Dim seriesIndex As Integer
  For seriesIndex = 1 To numberOfSeries
    Dim series As AcChartSeries
    Set series = baseLayer.GetSeries( seriesIndex )
    If Series.GetNumberOfPoints( ) > 0 Then
      hasData = True
      Exit For
    End If
  Next seriesIndex
  If hasData Then
    Exit Sub
  End If

  ' Hide empty chart
  GetChartDrawingPlane( ).SetHidden( True )

  ' Get the size of the drawing in points
  Dim w As Double
  w = Size.Width / OnePoint
  Dim h As Double
  h = Size.Height / OnePoint

  Dim svg As String
  svg = "<svg version='1.1'"
+ ' Standard SVG 1.1 namespaces
+ & " xmlns='http://www.w3.org/2000/svg'"
+ & " xmlns:xlink='http://www.w3.org/1999/xlink'"
+ ' Do not collapse whitespace in text
+ & " xml:space='preserve'"
+ ' Scale the SVG to use points as the default units
+ & " viewBox='0 0 " & SVGDbl( w ) & " " & SVGDbl( h ) & "'>"

  ' Define the font style
  Dim messageFont As AcFont
  messageFont.Bold = True
  messageFont.Color = Red
  messageFont.FaceName = "Arial"
  ' Font size is 25% of chart height
  messageFont.Size = h * 0.25
  svg = svg
+ & "<defs>"
+ & SVGFontStyle( "Message", messageFont, "text-anchor:middle;" )
+ & "</defs>"

  ' Draw the text
  svg = svg
+ & "<text class='Message'"
+ ' Center text horizontally and vertically
+ & SVGAttr( "x", w * 0.5 )
+ & SVGAttr( "y", (h * 0.5) + (messageFont.Size * 0.35) )
+ & ">"
+ & SVGStr( "No Data!" )
+ & "</text>"
+ & "</svg>"

  ' Add the text in front of the chart
  Dim svgPlane As AcDrawingSVGPlane
  Set svgPlane = AddDrawingPlane( DrawingPlaneTypeSVG )
  svgPlane.SetSVG( svg )
End Sub
```

See also AcChart::DrawOnChart method
AcDrawingPlane::IsHidden method

AcDrawingPlane::SetPosition method

Call this method to set the position of a drawing plane within its parent drawing.

Syntax Sub SetPosition(x As AcTwips, y As AcTwips)

Parameters **x**
The *x*-coordinate of the drawing plane, measured from the left edge of the drawing.

y
The *y*-coordinate of the drawing plane, measured from the top of the drawing.

Example For an example of how to use this method, see the example for the AcDrawingChartPlane class.

See also Class AcDrawingChartPlane
AcDrawingPlane::SetSize method

AcDrawingPlane::SetSize method

Call this method to set the size of a drawing plane.

Syntax Sub SetSize(width As AcTwips, height As AcTwips)

Parameters **width**
The width of the drawing plane.

height
The height of the drawing plane.

Example For an example of how to use this method, see the example for the AcDrawingChartPlane class.

See also Class AcDrawingChartPlane
AcDrawingPlane::SetPosition method

Class AcDrawingSVGPlane

An SVG drawing plane within a drawing. Figure 7-45 shows the class hierarchy of AcDrawingSVGPlane.

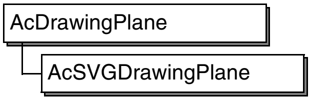


Figure 7-45 AcDrawingSVGPlane

Description AcDrawingSVGPlane represents a drawing plane whose contents are defined using Scalable Vector Graphics (SVG).

To add SVG drawing planes to a drawing, use the AcDrawing::AddDrawingPlane( ) or AcDrawing::InsertDrawingPlane( ) methods. You cannot use the New keyword, or the NewInstance( ) or NewPersistentInstance( ) methods to create AcDrawingSVGPlane objects.

Example For an example of how to work with an SVG drawing plane, see the example for the AcDrawing class.

See also Class AcDrawing
Class AcDrawingChartPlane
Class AcDrawingPlane

Methods for Class AcDrawingSVGPlane

Methods defined in Class AcDrawingSVGPlane

GetSVG, SetSVG

Methods defined in Class AcDrawingPlane

GetDrawingPlaneType, IsHidden, SetHidden, SetPosition, SetSize

AcDrawingSVGPlane::GetSVG method

Returns the SVG code for an SVG drawing plane.

Syntax Function GetSVG( ) As String

Returns The SVG code for the SVG drawing plane.

See also AcDrawingSVGPlane::SetSVG method

AcDrawingSVGPlane::SetSVG method

Call this method to set the SVG code for an SVG drawing plane.

Syntax

```
Sub SetSVG( svg As String )
```

Parameter

svg
The SVG code for the SVG drawing plane.

Example

For an example of how to use this method, see the example for the AcDrawing class.

See also

Class AcDrawing
AcDrawingSVGPlane::GetSVG method

Class AcDynamicTextControl

Displays text that uses more than one format style and varying amounts of data. Figure 7-46 shows the class hierarchy of AcDynamicTextControl.

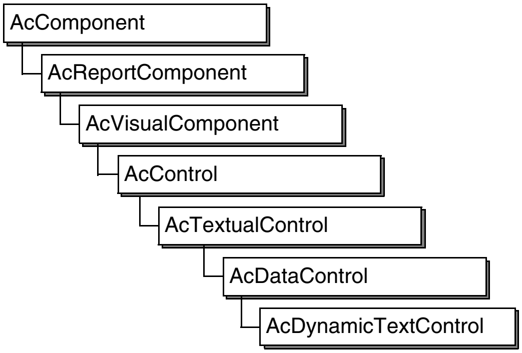


Figure 7-46 AcDynamicTextControl

Description AcDynamicTextControl is the AFC class that provides the ability to display text with multiple style formats and varying amounts of data. A dynamic text control adjusts its size and the size of the frame containing it to accommodate varying amounts of data. If necessary, a dynamic text control may split over multiple flows.

A single AcDynamicTextControl supports one of the following text formats:

- Plaintext

  Plaintext has no format tags but can contain ASCII control codes for specifying carriage returns, line feeds, tabs, and so on. AcDynamicTextControl supports only the CR, LF, and TAB control codes.

- HTML

  AcDynamicTextControl supports a subset of the HTML 4 standard. The Dynamic Text Control ignores HTML tags it does not recognize.

- RTF

  AcDynamicTextControl supports a subset of the RTF 1.6 standard. The Dynamic Text Control ignores RTF tags it does not recognize.

Properties

Table 7-39 lists AcDynamicTextControl properties.

Table 7-39 AcDynamicTextControl properties

Property	Group	Type	Description
AutoSplit Vertical	Pagination	AcAutoSplit	Specifies how the control may be split vertically. The default value is DefaultSplitting.
Justified LineWidth Padding	Text Layout	AcPercentage	The percentage by which the line width used to determine line breaks for fully justified lines is less than the actual width of the line. This padding can prevent clipping when report output is scaled. The default value is 2.5%.
KeepTagged Text	N/A	Boolean	Specifies whether to retain the tagged text once it has been processed. The default value is False.
LineSpacing	Text Layout	Double	The multiplier to be applied to determine the amount of vertical space between lines. The value is multiplied by the line height to calculate line spacing. The default value is 1.
LineWidth Padding	Text Layout	AcPercentage	The percentage by which the line width used to determine line breaks for lines that are not fully justified is less than the actual width of the line. This padding can prevent clipping when report output is scaled. The default value is 7.5%.
Minimum LineHeight	Text Layout	AcTwips	The minimum height of a line in the control. The default value is 0pt.
NoSplit Bottom	Pagination	AcTwips	The height of the area that must not be split at the bottom of the control. The default value is 0pt.
NoSplitTop	Pagination	AcTwips	The height of the area that must not be split at the top of the control. The default value is 0pt.
Space Between Lines	Text Layout	AcTwips	The fixed amount of space to add between lines within a paragraph. The default value is 0pt.
Space Between Paragraphs	Text Layout	AcTwips	The amount of vertical space between paragraphs. The default value is 6pt.

(continues)

Table 7-39 AcDynamicTextControl properties (continued)

Property	Group	Type	Description
SplitMargin Bottom	Pagination	AcTwips	The margin between the bottom edge and the contents of segments of split controls. This setting does not apply to the last segment. The default value is 0pt.
SplitMargin Top	Pagination	AcTwips	The margin between the top edge and the contents of segments of split controls. This setting does not apply to the first segment. The default value is 0pt.
TabPadding	Text Layout	AcPercentage	The percentage by which the width of the text chunk is increased when calculating the text chunk's end position to determine the next tab stop. The default value is 7.5%.
TabSpacing	Text Layout	AcTwips	The default spacing between tab stops, and the default indentation of bulleted and numbered lists. The default value is 36pt.
TextFormat	N/A	AcTextFormat	The tagging format of the text. The default value is TextFormatPlain.
WidowAnd Orphan Control	Pagination	Boolean	True prevents the last line of a paragraph from appearing at the top of a page by itself and prevents the first line of a paragraph from appearing at the bottom of the page by itself. The default value is True.

Methods for Class AcDynamicTextControl

Methods defined in Class AcDynamicTextControl

AutoSplitVertical, BuildText, GetAvailableHeight, GetAvailableWidth, GetFixedWidthFontFaceName, GetPlainText, GetTaggedText, KeepTaggedText, LineSpacing, LineWidthPadding, MinimumLineHeight, NoSplitBottom, NoSplitTop, ProcessText, SetTaggedText, SpaceBetweenLines, SpaceBetweenParagraphs, SplitMarginBottom, SplitMarginTop,TabPadding, TabSpacing, TextFormat, WidowAndOrphanControl

Methods inherited from Class AcDataControl

Format, GetGroupKey, IsSummary

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcDynamicTextControl::AutoSplitVertical method

Returns the value of the AutoSplitVertical property for a dynamic text control.

Syntax Function AutoSplitVertical( ) As AcAutoSplit

Returns The value of the control's AutoSplitVertical property.

AcDynamicTextControl::BuildText method

Parses tagged text and populates the internal data structure of the control. If the operation is unsuccessful, report execution fails.

Syntax Function BuildText( ) As Boolean

Returns True if the operation is successful.
False if the operation is unsuccessful.

AcDynamicTextControl::GetAvailableHeight method

Returns the height of the area in which text can be placed within the control. This value is the height of the control after resizing, less the top and bottom margins.

Syntax Function GetAvailableHeight( ) As AcTwips

Returns The height of the area in which text can be placed within the control in twips.

AcDynamicTextControl::GetAvailableWidth method

Returns the width of the area in which text can be placed within the control. This value is the width of the control, less the left and right margins.

Syntax Function GetAvailableWidth( ) As AcTwips

Returns The width of the area in which text can be placed within the control.

AcDynamicTextControl::GetFixedWidthFontFaceName method

Returns the name of the font to use as the default fixed-width font.

Syntax Function GetFixedWidthFontFaceName( ) As String

Returns The value of the name of the font to use as the default fixed-width font.

AcDynamicTextControl::GetPlainText method

Returns the Plaintext variable value. The Plaintext value is the tagged text without the text formatting tags.

Syntax Function GetPlainText( ) As String

Returns The value of the control's Plaintext variable.

AcDynamicTextControl::GetTaggedText method

Returns the TaggedText value. The TaggedText value is the tagged text including the text formatting tags.

Syntax Function GetTaggedText( ) As String

Returns The value of the control's TaggedText variable.

AcDynamicTextControl::KeepTaggedText method

Returns the value of the KeepTaggedText property.

Syntax Function KeepTaggedText( ) As Boolean

Returns The value of the control's KeepTaggedText property.

AcDynamicTextControl::LineSpacing method

Returns the value of the LineSpacing property.

Syntax Function LineSpacing( ) As Double

Returns The value of the control's LineSpacing property.

AcDynamicTextControl::LineWidthPadding method

Returns the value of the LineWidthPadding property.

Syntax Function LineWidthPadding( ) As AcPercentage

Returns The value of the control's LineWidthPadding property.

AcDynamicTextControl::MinimumLineHeight method

Returns the value of the MinimumLineHeight property.

Syntax Function MinimumLineHeight( ) As AcTwips

Returns The value of the control's MinimumLineHeight property.

AcDynamicTextControl::NoSplitBottom method

Returns the value of the NoSplitBottom property.

Syntax Function NoSplitBottom( ) As AcTwips

Returns The value of the control's NoSplitBottom property.

AcDynamicTextControl::NoSplitTop method

Returns the value of the NoSplitTop property.

Syntax Function NoSplitTop( ) As AcTwips

Returns The value of the control's NoSplitTop property.

AcDynamicTextControl::ProcessText method

Creates the internal data structure for the control and calls the BuildText( ) method.

Syntax Sub ProcessText( )

See also AcDynamicTextControl::BuildText method

AcDynamicTextControl::SetTaggedText method

Sets the TaggedText value.

Syntax Sub SetTaggedText(newText As String)

Parameter **newText**
The value to set.

AcDynamicTextControl::SpaceBetweenLines method

Returns the value of the SpaceBetweenLines property.

Syntax Function SpaceBetweenLines( ) As AcTwips

Returns The value of the control's SpaceBetweenLines property.

AcDynamicTextControl::SpaceBetweenParagraphs method

Returns the value of the SpaceBetweenParagraphs property.

Syntax Function SpaceBetweenParagraphs( ) As AcTwips

Returns The value of the control's SpaceBetweenParagraphs property.

AcDynamicTextControl::SplitMarginBottom method

Returns the value of the SplitMarginBottom property.

Syntax Function SplitMarginBottom( ) As AcTwips

Returns The value of the control's SplitMarginBottom property.

AcDynamicTextControl::SplitMarginTop method

Returns the value of the SplitMarginTop property.

Syntax Function SplitMarginTop( ) As AcTwips

Returns The value of the control's SplitMarginTop property.

AcDynamicTextControl::TabPadding method

Returns the value of the TabPadding property.

Syntax Function TabPadding( ) As AcPercentage

Returns The value of the control's TabPadding property.

AcDynamicTextControl::TabSpacing method

Returns the value of the TabSpacing property.

Syntax Function TabSpacing( ) As AcTwips

Returns The value of the control's TabSpacing property.

AcDynamicTextControl::TextFormat method

Returns the value of the TextFormat property.

Syntax Function TextFormat( ) As AcTextFormat

Returns The value of the x property.

AcDynamicTextControl::WidowAndOrphanControl method

Returns the value of the WidowAndOrphanControl property.

Syntax Function WidowAndOrphanControl( ) As Boolean

Returns The value of the control's WidowAndOrphanControl property.

Class AcExcelApp

Creates an Excel object within a report design. Figure 7-47 shows the class hierarchy of AcExcelApp.

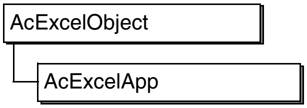


Figure 7-47 AcExcelApp

Description AcExcelApp is the root class that contains all instances of classes you use to generate and work with Excel files. You create other objects using methods in AcExcelApp or methods in other objects created from AcExcelApp.

For information about how the report gets font information for rendering Excel files, see *Developing Reports using e.Report Designer Professional.*

Methods for Class AcExcelApp

Methods defined in Class AcExcelApp

AddWorkbook, DeleteWorkbook, FindWorkbook, New, SetFontScalingFactor

AcExcelApp::AddWorkbook method

Adds a new workbook. The framework assigns a default file name to the workbook, such as Book1.xls. The default path to the workbook is the directory that contains the report object instance (.roi) file. For reports generated on iServer, the path listed in the AC_VIEWSERVER_EXCELOUTPUTDIR environment variable is the default path. To change either the file name or the default path for the file, use the AcExcelWorkbook::SaveAs method. To set a default directory for all Excel output, use the AC_VIEWSERVER_EXCELOUTPUTDIR environment variable.

Syntax AddWorkbook( ) As AcExcelWorkbook

Returns The handle to the workbook if the workbook is added.
An empty handle if an error occurred.

Example

```
Dim excelWorkbook As AcExcelWorkbook
Set excelWorkbook = excelApp.AddWorkbook( )
```

AcExcelApp::DeleteWorkbook method

Deletes a workbook, using either the workbook name or a reference to the workbook.

Syntaxes Function DeleteWorkbook(wbName As String) As Integer

Function DeleteWorkbook(workbook As AcExcelWorkbook) As Integer

Parameters **wbName**
The fully qualified name of the workbook to delete.

workbook
The handle to the workbook to delete.

Example

```
excelApp.DeleteWorkbook(workbookName);
excelApp.DeleteWorkbook (excelWorkbook)
```

AcExcelApp::FindWorkbook method

Finds the specified workbook that was created using AddWorkbook( ). You can not use FindWorkbook( ) to find an existing Excel file on disk.

Syntaxes Function FindWorkbook(wbName As String) As AcExcelWorkbook

Function FindWorkbook(index As Integer) As AcExcelWorkbook

Parameters **wbName**
The fully qualified name of the workbook to find.

index
The index of the workbook to find.

Returns The handle to the workbook if found.
An empty handle if the workbook is not found or if an error occurred.

Example The following example shows how to find the workbook myWorkbook:

```
Dim excelWorkbook As AcExcelWorkbook
excelWorkbook = excelApp.FindWorkbook( "myWorkbook" )
```

See also AcExcelApp::AddWorkbook method

AcExcelApp::New method

Creates a new Excel application instance. Call New( ) before calling any other method.

Syntax Sub New( )

Returns The handle to the class.

Example The following example creates a new Excel application instance:

```
Dim excelApp As AcExcelApp
Set excelApp = New AcExcelApp
```

AcExcelApp::SetFontScalingFactor method

Specifies a scaling factor to apply to the specified font.

SetFontScalingFactor( ) applies the scaling factor to font sizes between startSize to endSize. If either startSize or endSize is zero, the scaling factor is applied to every font size. If either startSize or endSize is less than zero, the scaling factor is not applied.

Syntax

```
Sub SetFontScalingFactor( face As String, scalingFactor As Double, startSize As
    Integer, endSize As Integer )
```

Parameters

face
The font to which to apply the scaling factor.

scalingFactor
The scaling factor. Column width is adjusted if scalingFactor is greater than one.

startSize
The smallest font size for which to apply the font scaling factor.

endSize
The largest font size for which to apply the font scaling factor.

Class AcExcelCell

Represents a cell in a worksheet. Figure 7-48 shows the class hierarchy of AcExcelCell.

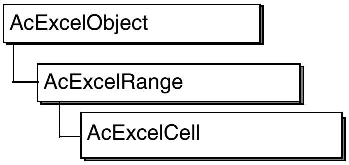


Figure 7-48 AcExcelCell

Description The AcExcelCell class represents a cell in a worksheet.

An AcExcelCell object is an AcExcelRange object in which rowNumTop=rowNumBottom, columnNumLeft=columnNumRight. The cell can contain up to 4,000 characters.

AcExcelCell does not override any methods of the AcExcelRange class.

Methods for Class AcExcelCell

Methods inherited from Class AcExcelRange

AddImage, DrawLine, GetBackgroundColor, GetBorder, GetFont, GetHorizontalAlignment, GetIndent, GetMergeCells, GetNumberFormat, GetValue, GetValueAsDate, GetVerticalAlignment, GetWrapText, SetBackgroundColor, SetBorder, SetBorderAround, SetFont, SetHorizontalAlignment, SetIndent, SetMergeCells, SetNumberFormat, SetValue, SetVerticalAlignment, SetWrapText

Class AcExcelColumn

Represents a column in a worksheet. Figure 7-49 shows the class hierarchy of AcExcelColumn.

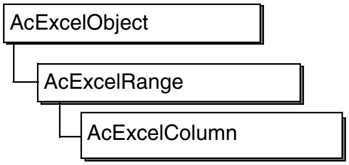


Figure 7-49 AcExcelColumn

Methods inherited from Class AcExcelRange

AddImage, DrawLine, GetBackgroundColor, GetBorder, GetFont, GetHorizontalAlignment, GetIndent, GetMergeCells, GetNumberFormat, GetValue, GetValueAsDate, GetVerticalAlignment, GetWrapText, SetBackgroundColor, SetBorder, SetBorderAround, SetFont, SetHorizontalAlignment, SetIndent, SetMergeCells, SetNumberFormat, SetValue, SetVerticalAlignment, SetWrapText

Description The AcExcelColumn class represents a column in a worksheet.

An AcExcelColumn object is an AcExcelRange object in which columnNumLeft is equal to columnNumRight.

Methods for Class AcExcelColumn

Methods defined in Class AcExcelColumn

Autofit, GetColumnWidth, SetAutofitFont, SetAutofitString, SetColumnWidth

AcExcelColumn::Autofit method

Adjusts the column width to fit the contents of the tallest cell in the column.

Syntax Function Autofit( ) As Integer

Returns The column width expressed as an integer.

AcExcelColumn::GetColumnWidth method

Returns the column width, in number of characters that can be displayed in a column. The default value is 8.43 characters.

Syntax Function GetColumnWidth( ) As Double

Returns The column width if it was explicitly set.
The default column width if the column width was not explicitly set.

AcExcelColumn::SetAutofitFont method

Sets the font to use to calculate column width.

Syntax Sub SetAutofitFont(font As AcFont)

Parameter **font**
The font to use.

AcExcelColumn::SetAutofitString method

Sets the string to use to calculate column width.

Syntax Sub SetAutofitString(val As String)

Parameter **val**
The string to use.

AcExcelColumn::SetColumnWidth method

Sets the number of characters that can appear in a column when you use a standard font. The default column width is 8.43 characters. The standard font is Arial size 10. The default value is assigned when the column is created.

Syntax Sub SetColumnWidth(width As Double)

Parameter **width**
The number of characters that can be appear in the column. Must be 0-255.

Class AcExcelObject

The abstract base class for creating Excel objects in a report. Figure 7-50 shows the class hierarchy of AcExcelObject.

Figure 7-50 AcExcelObject

Description Classes derived from AcExcelObject create and manage the Excel workbooks, worksheets, ranges, rows, columns, and cells you use in an Actuate report.

Methods for Class AcExcelObject

There are no public methods for this class.

Class AcExcelRange

The abstract base class for the AcExcelCell, AcExcelColumn, and AcExcelRow classes. Figure 7-51 shows the class hierarchy of AcExcelRange.

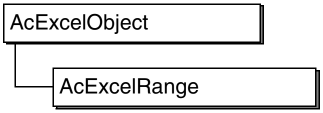


Figure 7-51 AcExcelRange

Description AcExcelRange class is the base class for the AcExcelCell, AcExcelColumn, and AcExcelRow classes.

Unless otherwise stated, all Set( ) methods for this class set the properties for every cell in the range. If all cells contain the same property values, Get( ) methods for this class return the property value. If any cell contains different property values, Get( ) methods for this class return null.

See also Class AcExcelCell
Class AcExcelColumn
Class AcExcelRow

Methods for Class AcExcelRange

Methods defined in Class AcExcelRange

AddImage, DrawLine, GetBackgroundColor, GetBorder, GetFont, GetHorizontalAlignment, GetIndent, GetMergeCells, GetNumberFormat, GetValue, GetValueAsDate, GetVerticalAlignment, GetWrapText, SetBackgroundColor, SetBorder, SetBorderAround, SetFont, SetHorizontalAlignment, SetIndent, SetMergeCells, SetNumberFormat, SetValue, SetVerticalAlignment, SetWrapText

AcExcelRange::AddImage method

Adds an image to the range.

Syntax Sub AddImage(fName As String)

Parameter **fName**
The file name of the image to add.

AcExcelRange::DrawLine method

Sets properties of a line in the range.

Syntax Sub DrawLine(which As Integer, style As Integer, weight As Integer, color As AcColor)

Parameters **which**
The line to draw.

style
The style of the line.

weight
The weight of the line.

color
The color of the line.

AcExcelRange::GetBackgroundColor method

Returns the background color of the range.

Syntax Function GetBackgroundColor( ) As AcColor

Returns The background color if it is the same for all cells in the range.
Null if the background color is different for any cells in the range.

AcExcelRange::GetBorder method

Returns the border of the range.

Syntax Function GetBorder(which As Integer) As AcExcelBorder

Parameter **which**
The side of the border to access. Values are:

- ExcelBorderTop
- ExcelBorderBottom
- ExcelBorderLeft
- ExcelBorderRight

Returns The border if the border attribute for the specified border is the same for all cells in the range.
Null if the border attribute for the specified border is different for any cells in the range.

AcExcelRange::GetFont method

Returns the font used for the range.

Syntax Function GetFont( ) As AcFont

Returns The font used for the range if all cells in the range use the same font.
Null if cells in the range use different fonts.

AcExcelRange::GetHorizontalAlignment method

Returns the setting of the horizontal alignment option.

Syntax Function GetHorizontalAlignment( ) As AcExcelHorizontalAlignment

Returns The horizontal alignment setting if it is the same for all cells in the range.
Null if the horizontal alignment setting of any cell in the range is different.

AcExcelRange::GetIndent method

Returns the number of characters that the text in the range is indented from the left edge of the cell.

Syntax Function GetIndent( ) As Integer

Returns The number of indent characters if the indent is set and is the same for all cells in the range.
0 if the indent for any cell is not set.
Null if the indent is different for any cells in the range.

AcExcelRange::GetMergeCells method

Returns the setting of the merge cells option.

Syntax Function GetMergeCells( ) As Boolean

Returns True if cells in the range are merged.
False if the merge cells option is set to False for all cells in the range or if the merge cells option is set to True for all cells but the cells are not merged.
Null if the merge cells setting of any cell in the range is different.

AcExcelRange::GetNumberFormat method

Returns the string used for formatting numbers in the range. The string can be different from the string passed in SetNumberFormat( ) method.

Syntax Function GetNumberFormat( ) As String

Returns The string used for formatting the range if all cells in the range use the same format string.
Null if cells in the range use different format strings.

AcExcelRange::GetValue method

Returns the contents of the range. If the return value is not a String or Date, Actuate Basic converts return values into the appropriate format. If the return value is a Date, use the GetValueAsDate( ) method instead of the GetValue( ) method.

Syntax Function GetValue( ) As Variant

Returns The contents of the range if all cells in the range contain the same value.
Null if calls in the range contain different values.

See also AcExcelRange::GetValueAsDate method

AcExcelRange::GetValueAsDate method

Converts the contents of the range into date format.

Syntax Function GetValueAsDate( ) As Date

Returns The date value of the range if all cells in the range contain the same value.
Null if cells in the range contain different values.

AcExcelRange::GetVerticalAlignment method

Returns the vertical alignment setting.

Syntax Function GetVerticalAlignment( ) As AcExcelVerticalAlignment

Returns The vertical alignment setting if it is the same for all cells in the range.
Null if the vertical alignment setting of any cell in the range is different.

AcExcelRange::GetWrapText method

Returns the setting of the wrap text option.

Syntax Function GetWrapText( ) As Boolean

Returns The wrap text setting if it is the same for all cells in the range.
Null if the wrap text setting of any cell in the range is different.

AcExcelRange::SetBackgroundColor method

Sets the background color for the range. You can use up to 48 custom colors in a workbook. This includes background, font, and border colors. If the color is invalid, SetBackgroundColor( ) sets the background color to white.

Syntax Sub SetBackgroundColor(color As AcColor)

Parameter **color**
The color to set.

AcExcelRange::SetBorder method

Sets the border for one or more sides of the range.

If the border color is invalid, SetBorder( ) sets the border color to black.

If the border style value is greater than zero, SetBorder( ) sets the border style to zero, ExcelBorderNone. If the border style value is greater than 13, SetBorder( ) sets the border style to 13, ExcelBorderSlantedDashDot.

Syntax Sub SetBorder(border As AcExcelBorder, which As Integer)

Parameters **border**

The side of the border to set. Values are:

- ExcelBorderTop
- ExcelBorderBottom
- ExcelBorderLeft
- ExcelBorderRight

The default setting is no border.

which

The border to set.

AcExcelRange::SetBorderAround method

Sets a border around the range.

Syntax Sub SetBorderAround(border As AcExcelBorder)

Parameter **border**

The border to set.

AcExcelRange::SetFont method

Sets the font properties for the range. You can use up to 512 fonts in a single file, including default Excel fonts. You can use up to 48 custom colors. These limits include background, font, and border colors.

If the font color is invalid, SetFont( ) sets the font color to black.

If the font size is less than one, SetFont( ) sets the font size to one. If the font size is greater than 409, SetFont( ) sets the font size to 409.

If the font name is an empty string, SetFont( ) sets the font to Arial.

Syntax Sub SetFont(font As AcFont)

Parameters **font**

The font to set.

AcExcelRange::SetHorizontalAlignment method

Sets horizontal alignment option for the range.

If the value of the alignment option is less than zero, SetHorizontalAlignment( ) sets horizontal alignment to ExcelHAlignmentGeneral. If the value of the alignment option is greater than six, SetHorizontalAlignment( ) sets horizontal alignment to ExcelHAlignmentCenterAcrossSelection.

Syntax Sub SetHorizontalAlignment(h As AcExcelHorizontalAlignment)

Parameter **h**
The horizontal alignment option to set. Values are:

- ExcelHAlignGeneral
- ExcelHAlignLeft
- ExcelHAlignCenter
- ExcelHAlignRight
- ExcelHAlignFill
- ExcelHAlignJustify
- ExcelHAlignCenterAcrossSelection

AcExcelRange::SetIndent method

Sets the indent property for the range. Specify the indent as an integer denoting the number of characters to indent. If indent is set, the horizontal alignment type automatically changes to left indent.

If indent is less than 0, SetIndent( ) sets the indent to 0. If indent is greater than 15, SetIndent( ) sets the indent to 15.

Syntax Sub SetIndent(indent As Integer)

Parameter **indent**
The number of characters to indent. Must be 0-15.

AcExcelRange::SetMergeCells method

Turns the merge cells option on and off. When the merge cells option is on, the top leftmost cell contains the option settings and all other cells contain the default settings. These settings include value, font, and alignment of the first cell in the range with a set value.

Setting <mergeCells> to True on both a column and a row in a single worksheet causes all cells in the worksheet to be created. This setting can use all virtual memory in a system.

Syntax Sub SetMergeCells(mergeCells As Boolean)

Parameter **mergeCells**
True turns on the merge cells option.
False turns off the merge cells option.

AcExcelRange::SetNumberFormat method

Sets the number format properties for the range.

To display numbers in a specific format, use SetNumberFormat( ) in conjunction with SetValue( ). When you use SetNumberFormat( ) with SetValue( ), you must call SetValue( ) before SetNumberFormat( ).

You specify the format using the constants in Table 7-40 or an explicit format string.

Table 7-40 Number formats

Constant	Format
ExcelCurrencyFloat	$0.00
ExcelCurrencyFloatWithSeparator	$#,##0.00
ExcelCurrencyInt	$0
ExcelCurrencyIntWithSeparator	$#,##0
ExcelExp	0.00E+00
ExcelFixed	0.00
ExcelFloat	0.00
ExcelFloatWithSeparator	#,##0.00
ExcelGeneralDate	mm/dd/yyyy hh:mm:ss AM/PM
ExcelGeneralNumber	General
ExcelInt	0
ExcelIntWithSeparator	#,##0
ExcelLongDate	dddd, mmmm dd, yyyy
ExcelLongTime	hh:mm:ss AM/PM
ExcelMediumDate	dd-mmm-yy
ExcelMediumTime	h:mm AM/PM
ExcelPercent	0.00%
ExcelShortDate	mm/dd/yyyy
ExcelShortTime	hh:mm
ExcelStandard	0.00

Syntax Sub SetNumberFormat(numFormat As String)

Parameter **numFormat**
The format to set.

Example The following example shows how to use SetNumberFormat( ) and SetValue( ) to display a date in the mm-yyyy format:

```
Dim date As Date
date=DateValue ("11-1982")
Cell.SetValue (date)
Cell.SetNumberFormat ("mm-yyyy")
```

See also AcExcelRange::SetValue method

AcExcelRange::SetValue method

Sets the contents for every cell in the range. If the numeric value is of Date or Currency type, SetValue( ) displays the number in the default Date or Currency Format. To display numbers in a different format, use SetValue( ) in conjunction with SetNumberFormat( ). When using SetNumberFormat( ) with SetValue( ), you must call SetValue( ) before SetNumberFormat( ).

The maximum number of characters for a cell string value is 255.

Syntax Sub SetValue(val As Variant)

Parameter **val**
The value to set.

See also AcExcelRange::SetNumberFormat method

AcExcelRange::SetVerticalAlignment method

Sets the vertical alignment option for the range.

If the value of the alignment option is less than zero, SetHorizontalAlignment( ) sets vertical alignment to ExcelVAlignmentTop. If the value of the alignment option is greater than three, SetVerticalAlignment( ) sets vertical alignment to three, which equals ExcelVAlignJustify.

Syntax Sub SetVerticalAlignment(v As AcExcelVerticalAlignment)

Parameter **v**
The vertical alignment option to set. Values are:

- ExcelVAlignTop
- ExcelVAlignCenter
- ExcelVAlignBottom
- ExcelVAlignJustify

AcExcelRange::SetWrapText method

Turns the wrap text option on and off.

Syntax Sub SetWrapText(wrapText As Boolean)

Parameter **wrapText**

True turns on the wrap text option.
False turns off the wrap text option.

Class AcExcelRow

Represents a row in a workbook. Figure 7-52 shows the class hierarchy of AcExcelRow.

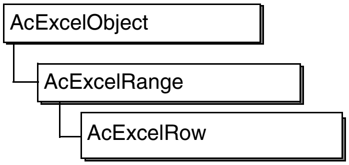


Figure 7-52 AcExcelRow

Description The AcExcelRow class represents a row in a workbook. An AcExcelRow object is an AcExcelRange object in which rowNumTop is equal to rowNumBottom.

See also Class AcExcelCell
Class AcExcelColumn
Class AcExcelRange

Methods for Class AcExcelRow

Methods defined in Class AcExcelRow

GetRowHeight, SetRowHeight

Methods inherited from Class AcExcelRange

AddImage, DrawLine, GetBackgroundColor, GetBorder, GetFont, GetHorizontalAlignment, GetIndent, GetMergeCells, GetNumberFormat, GetValue, GetValueAsDate, GetVerticalAlignment, GetWrapText, SetBackgroundColor, SetBorder, SetBorderAround, SetFont, SetHorizontalAlignment, SetIndent, SetMergeCells, SetNumberFormat, SetValue, SetVerticalAlignment, SetWrapText

AcExcelRow::GetRowHeight method

Returns the row height, in points.

Syntax Function GetRowHeight( ) As Double

Returns The row height if explicitly set.
The default row height, 12.75 points, if not explicitly set.

AcExcelRow::SetRowHeight method

Sets the row height. The default row height is 12.75 points. The default value is assigned when the row is created. Excel adjusts row height to fit row contents. As such, you typically do not need to call SetRowHeight( ).

Syntax

```
Sub SetRowHeight( height As Double )
```

Parameter

height
The row height in points. Must be 0-409.

Class AcExcelWorkbook

Provides the logic for adding and removing worksheets and getting information about a specific workbook. Figure 7-53 shows the class hierarchy of AcExcelWorkbook.

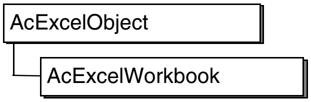


Figure 7-53 AcExcelWorkbook

Description Use AcExcelWorkbook to add, find, save, and delete worksheets in a workbook in Microsoft Excel 97-2003 format.

See also AcExcelApp::AddWorkbook method

Methods for Class AcExcelWorkbook

Methods defined in Class AcExcelWorkbook

AddWorksheet, DeleteWorksheet, FindWorksheet, GetFullName, Save, SaveAs

AcExcelWorkbook::AddWorksheet method

Adds a worksheet to the workbook. The new worksheet is added after the most recently created worksheet. AddWorksheet( ) assigns a unique label to the worksheet, such as Sheet1.

Syntax Function AddWorksheet( ) As AcExcelWorksheet

Returns The handle to the added worksheet.
An empty handle if an error occurred.

Example The following code shows how to add a worksheet to an existing workbook:

```
Dim excelWorksheet As AcExcelWorksheet
Set excelWorksheet = excelWorkbook.AddWorksheet( )
```

AcExcelWorkbook::DeleteWorksheet method

Deletes a worksheet from the workbook if the worksheet exists.

Syntaxes Function DeleteWorksheet(wsName As String)

Function DeleteWorksheet(worksheet As AcExcelWorksheet)

Parameters **wsName**
The fully qualified name of the worksheet to delete.

worksheet
The handle to the worksheet to delete.

Example The following code shows how to delete a worksheet by using its name or a handle to a worksheet object:

```
excelWorkbook.DeleteWorksheet( worksheetName );
excelApp.DeleteWorksheet( excelWorksheet )
```

AcExcelWorkbook::FindWorksheet method

Finds the specified worksheet in the workbook.

Syntaxes Function FindWorksheet(wsName As String) As AcExcelWorksheet

Function FindWorksheet(index As Integer) As AcExcelWorksheet

Parameters **wsName**
The unique name of the worksheet to find.

index
The index of the worksheet to find.

Returns The handle to the worksheet if found.
An empty handle if the worksheet is not found or if an error occurred.

Example The following example shows how to find the worksheet Sheet1:

```
Dim excelWorksheet As AcExcelWorksheet
Set excelWorksheet = excelWorkbook.FindWorksheet( "Sheet1" )
```

AcExcelWorkbook::GetFullName method

Returns the fully qualified name of the workbook.

Syntax Function GetFullName( ) As String

Returns The name of the workbook.

AcExcelWorkbook::Save method

Saves the workbook. When testing a report design in e.Report Designer Professional, the workbook saves in the directory that contains the report object instance (.roi) file.

For reports that run on iServer, the workbook saves in the directory specified in the AC_VIEWSERVER_EXCELOUTPUTDIR environment variable. If the AC_VIEWSERVER_EXCELOUTPUTDIR environment variable is not set, reports that run on iServer save in the $AC_SERVER_HOME/Excel directory.

If the workbook is successfully saved, the report runs without an error. If an error occurs during this operation, an error message to signal an I/O error is logged and the Excel file is not saved.

Syntax Function Save( )

AcExcelWorkbook::SaveAs method

Saves the workbook with a specified name. The specified name overwrites the default name assigned when the workbook is created. For reports generated on the local machine, the workbook saves in the directory that contains the .roi file.

For reports generated on iServer, the workbook saves in directory specified in the AC_VIEWSERVER_EXCELOUTPUTDIR environment variable. If the AC_VIEWSERVER_EXCELOUTPUTDIR environment variable is not set, reports generated on iServer save in the $AC_SERVER_HOME/Excel directory.

If the workbook is successfully saved, the report generates without an error. If an error occurs during this operation, an error message to signal an I/O error is logged and the Excel file is not saved.

Syntax Function SaveAs(wbName As String) As Integer

Parameter **wbName**
The file name. Can be fully qualified or only the file name.
If only the file name is specified, the workbook is saved in the default directory.
Can not be an empty string or a Null value.

Class AcExcelWorksheet

Contains information about a specific worksheet. Figure 7-54 shows the class hierarchy of AcExcelWorksheet.

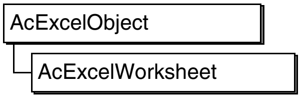


Figure 7-54 AcExcelWorksheet

Description A worksheet contains cells that can be identified by the cell's unique coordinate composed of a row number and a column number. Use AcExcelWorksheet to manipulate cells in a worksheet. Use the AcExcelWorkbook::AddWorksheet method to obtain the handle to the worksheet.

The worksheet can contain up to 65,536 rows and up to 256 columns.

See also Class AcExcelWorkbook

Methods for Class AcExcelWorksheet

AutoFit, GetCell, GetColumn, GetDisplayGridlines, GetName, GetRange, GetRow, SetDisplayGridlines, SetName

AcExcelWorksheet::AutoFit method

Adjusts the column width of all cells in the worksheet to fit the contents of the tallest cell.

Use AcExcelColumn::SetAutofitString( ) and AcExcelColumn::SetAutofitFont( ) to set the string and font to use to calculate column width.

Syntax Function AutoFit( ) As Integer

See also AcExcelColumn::SetAutofitFont method
AcExcelColumn::SetAutofitString method

AcExcelWorksheet::GetCell method

Returns the handle to the cell to access. GetCell( ) creates the cell if the cell does not exist.

Syntax Function GetCell(row As Integer, col As Integer) As AcExcelCell

Parameters **row**
The row number of the cell to access. Must be 1-65,536.

col
The column number of the cell to access. Must be 1-256.

Returns The handle to the cell if successful.
An empty handle if the row or column is out of range.

Example The following example returns a handle to cell 2 in row 1:

```
Dim cell As AcExcelCell
Set cell = excelWorksheet.GetCell(1, 2)
```

AcExcelWorksheet::GetColumn method

Returns the handle to the column to access.

Syntax Function GetColumn(col As Integer) As AcExcelColumn

Parameter **col**
The column number to access. Must be 1-256.

Returns The handle to the column if successful.
An empty handle if the column is out of range.

Example The following example returns a handle to column 1:

```
Dim range As AcExcelColumn
Set range = excelWorksheet.GetColumn(1)
```

AcExcelWorksheet::GetDisplayGridlines method

Determines whether the gridline setting is turned on.

Syntax Function GetDisplayGridlines( ) As Boolean

Returns True if the gridlines are turned on.
False if the gridlines are turned off.

AcExcelWorksheet::GetName method

Returns the unique name of the worksheet.

Syntax Function GetName( ) As String

Returns The name of the worksheet.

AcExcelWorksheet::GetRange method

Returns the handle to the cells to access. GetRange( ) creates the range if the range does not exist.

Syntax Function GetRange(cell1 As AcExcelCell, cell2 As AcExcelCell) As
AcExcelRange

Parameters **cell1**
The handle to the first cell to access.

cell2
The handle to the last cell to access.

Returns The handle to the range if successful.
An empty handle if an error occurred.

Example The following example returns a range of cells A1 through C3:

```
Dim cell1 As AcExcelCell
Dim cell2 As AcExcelCell
Set cell1 = excelWorksheet.GetCell ( 1,1 )
Set cell2 = excelWorksheet.GetCell ( 3,3 )
range = excelWorksheet.GetRange ( cell1, cell2 )
```

AcExcelWorksheet::GetRow method

Returns a handle to the row to access. GetRow( ) creates the row if the row does not exist.

Syntax Function GetRow(row As Integer) As AcExcelRow

Parameter **row**
The row number to access. Must be 1-65,536.

Returns A handle to the row if execution is successful.
An empty handle if the row is out of range.

Example The following example returns a handle to column 1:

```
Dim row As AcExcelRow
Set row = excelWorksheet.GetRow(1)
```

AcExcelWorksheet::SetDisplayGridlines method

Turns the gridlines on and off. The default setting is True.

Syntax Sub SetDisplayGridlines(bGrid As Boolean)

Parameter **bGrid**
True turns on the gridlines. False turns off the gridlines.

AcExcelWorksheet::SetName method

Sets the name of the worksheet. The name of the worksheet must be unique in a workbook. If the worksheet with the specified name exists in the workbook, the name remains unchanged.

Syntax Function SetName(wsName As String)

Parameter **wsName**
The name to set.

Class AcExternalDataSource

An abstract base class for generic data source objects that use a command to retrieve a single result set through an associated connection. Figure 7-55 shows the class hierarchy of AcExternalDataSource.

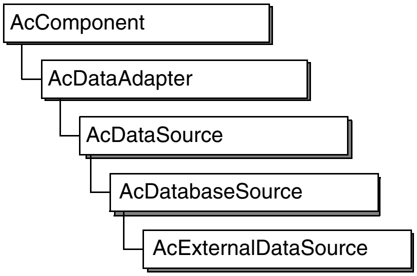


Figure 7-55 AcExternalDataSource

Description AcExternalDataSource is an abstract base class for generic data source objects that use a command to retrieve a single result set through an associated connection. Derived classes define how to specify the command. The data source then allocates a cursor to read rows from the command's result data.

See also Class AcDataAdapter
Class AcDatabaseSource
Class AcDataSource

Methods for Class AcExternalDataSource

Methods defined in Class AcExternalDataSource

ObtainCommand

Methods inherited from Class AcDatabaseSource

BindDataRow, BindStaticParameters, GetCursor, GetDBConnection, GetPreparedStatement, OpenCursor, SetStatementProperty

Methods inherited from Class AcDataSource

HasFetchedLast

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcExternalDataSource::ObtainCommand method

Obtains the command that retrieves the result set from the database. Derived classes must override ObtainCommand( ) to provide a custom command.

Syntax Function ObtainCommand( ) As String

Class AcFlow

An abstract base class that defines the logic for placing frames in a flow, the printable area of a page. Figure 7-56 shows the class hierarchy of AcFlow.

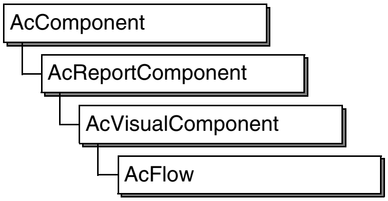


Figure 7-56 AcFlow

Description AcFlow is the abstract base class for AcTopDownFlow, the default flow in a report design. AcFlow defines the protocol that specifies how to add subpages, content, and header and footer frames to a flow. This class also specifies how to allocate flow space. Its derived class, AcTopDownFlow, implements the details for executing those tasks.

Class protocol

Table 7-41 lists the order of execution of AcFlow protocol methods.

Table 7-41 Class protocol for AcFlow

Method	Task
Start( )	Prepares the flow for receiving frames
AddFrame( )	Adds each frame to the flow
Finish( )	Identifies that the flow has received all the frames that it will get

Subclassing AcFlow

Create a subclass directly from AcFlow if your report requires a flow other than a top-down flow. For example, you might need a grid layout that flows from left to right. AcFlow defines several methods that you must override to specify implementation details. These methods, called pure virtual methods, are empty in AcFlow.

Variables

Table 7-42 lists AcFlow variables.

Table 7-42 AcFlow variables

Variable	Type	Description
BackgroundColor	AcColor	Specifies the background color of the flow. The default value is Transparent.
Border	AcBorder	Specifies the border, if any, to draw around the flow.

Properties

Table 7-43 lists AcFlow properties.

Table 7-43 AcFlow properties

Property	Type	Description
BackgroundColor	AcColor	Specifies the background color of the flow. The default value is Transparent.
Border	AcBorder	Specifies the border, if any, to draw around the flow.

See also Class AcBasePage
Class AcTopDownFlow

Methods for Class AcFlow

Methods defined in Class AcFlow

AddFooter, AddFrame, AddHeader, AddSubpage, AdjustFooter, CanFitFrame, CanFitHeight, GetFirstDataFrame, GetFreeSpace, GetInsideSize, GetLastDataFrame, IsEmpty, ReleaseSpace, ReserveSpace, ResetSpace, ResizeByConstrainedByContents, ShiftFooterUp

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcFlow::AddFooter method

Adds a frame to the flow as a footer.

Syntax Function AddFooter(footer As AcFrame) As Boolean

Parameter **footer**
The footer frame to add to the flow.

AcFlow::AddFrame method

Adds the frame to the flow. Derived classes override the AddFrame( ) method, a pure virtual method in AcFlow, to specify how frames are placed in the flow. If you create a subclass from AcFlow, you must override AddFrame( ) to specify the implementation details.

Syntax Sub AddFrame(frame As AcFrame)

Parameter **frame**
The frame to add to the flow.

See also AcPageList::EjectPage method

AcFlow::AddHeader method

Adds a page header frame to the flow. Derived classes override the AddHeader( ) method, a pure virtual method in AcFlow, to specify how header frames are placed at the top of the flow. If you create a subclass from AcFlow, you must override AddHeader( ) to specify the implementation details.

Syntax Function AddHeader(header As AcFrame) As Boolean

Parameter **header**
The page header frame to add.

Returns True if the page header was added to the flow.
False if the page header was not added to the flow.

AcFlow::AddSubpage method

Adds a subpage to the flow. Derived classes override the AddSubPage( ) method, a pure virtual method in AcFlow, to specify how subpages are placed in the flow. If you create a subclass from AcFlow, you must override AddSubPage( ) to specify the implementation details.

Syntax Function AddSubpage(subpage As AcSubpage) As Boolean

Parameter **subpage**
The subpage to add to the flow.

Returns True if the subpage was added to the flow.
False if the subpage was not added to the flow.

AcFlow::AdjustFooter method

Changes the available space in the flow to the correct amount of space for the page footer. Derived classes override AdjustFooter( ) to modify the amount of space in the flow reserved for the page footer. When the AFC starts the flow, it reserves space for the page footer in the flow. Later, the size of the page footer can change based on the data row processing. AdjustFooter can modify the space reserved for the page footer to account for size differences between the estimate made by the AFC at flow start time and the final size of the page footer.

Syntax Sub AdjustFooter(footer As AcFrame)

Parameter **footer**
The page footer to add.

AcFlow::CanFitFrame method

Checks if the flow contains enough space to contain a frame. Derived classes override the CanFitFrame( ) method, a pure virtual method in AcFlow, to specify how to determine if a frame fits in a flow. If you create a subclass from AcFlow, you must override CanFitFrame( ) to specify the implementation details.

Syntax Function CanFitFrame(frame As AcBaseFrame) As Boolean

Parameter **frame**
The frame to check for fit.

Returns True if the frame will fit into the flow.
False if the frame will not fit into the flow.

AcFlow::CanFitHeight method

Checks if the flow contains enough vertical space to contain a frame. Derived classes override the CanFitHeight( ) method, a pure virtual method in AcFlow, to specify how to determine if a frame fits in a flow. If you create a subclass from AcFlow, you must override CanFitHeight( ) to specify the implementation details.

Syntax Function CanFitHeight (height As Integer) As Boolean

Parameter **height**
The height of the component.

Returns True if the frame will fit into the flow.
False if the frame will not fit into the flow.

AcFlow::GetFirstDataFrame method

Returns the first data frame associated with the current flow.

Syntax Function GetFirstDataFrame( ) As AcFrame

Returns A handle to the first frame in the flow.

AcFlow::GetFreeSpace method

Returns the free space in the flow. Free space is the area of the flow minus the space occupied by frames in the flow. Derived classes override the GetFreeSpace( ) method, a pure virtual method in AcFlow, to specify how to return the available space. If you create a subclass from AcFlow, you must override GetFreeSpace( ) to specify the implementation details.

Syntax Function GetFreeSpace( ) As AcSize

Returns The amount of unused space, in twips, available in the flow.

AcFlow::GetInsideSize method

Returns the size, in twips, of the content rectangle of the flow.

Syntax Function GetInsideSize( ) As AcSize

Returns The size of the content rectangle of the flow.

AcFlow::GetLastDataFrame method

Returns the last data frame associated with the current flow.

Syntax Function GetLastDataFrame( ) As AcFrame

Returns A handle to the last frame in the flow.

AcFlow::IsEmpty method

Indicates whether the flow contains a data frame, such as a Content, Before, or After frame.

Syntax Function IsEmpty( ) As Boolean

Returns True if the flow is empty.
False if the flow contains a data frame.

AcFlow::ReleaseSpace method

In derived classes, ReleaseSpace( ) releases back to the flow all or part of the space reserved using ReserveSpace( ). Derived classes override ReleaseSpace( ), a pure virtual method in AcFlow, to specify how to release space. If you create a subclass from AcFlow, you must override ReleaseSpace( ) to specify the implementation details.

Be careful when using ReleaseSpace( ). If you release too much space, the next frame added to the flow might overlap with the contents of the existing flow.

Syntax Sub ReleaseSpace(width As Integer, height As Integer)

Parameters **width**
The width of the space to release.

height
The height of the space to release.

See also AcFlow::ReserveSpace method

AcFlow::ReserveSpace method

Reserves a part of the available space within the flow. Derived classes override ReserveSpace( ), a pure virtual method in AcFlow, to specify how to reserve space. If you create a subclass from AcFlow, you must override ReserveSpace( ) to specify the implementation details.

In derived classes, you can call ReserveSpace( ) to expand a frame or subpage, or to leave a gap between frames.

Syntax Sub ReserveSpace(width As Integer, height As Integer)

Parameters **width**
The width of the space to reserve.

height
The height of the space to reserve.

See also AcFlow::ReleaseSpace method

AcFlow::ResetSpace method

Sets the amount of space in the flow to zero. Derived classes can override ResetSpace( ), a pure virtual method in AcFlow, to reset the amount of space available for frames remaining in the flow to zero. If you create a subclass from AcFlow, you must override ResetSpace( ) only if you want the AFC to perform automatic balancing of the contents between multiple flows. You request automatic balancing by setting the BalanceFlows property on the page or subpage.

Syntax

```
Sub ResetSpace( )
```

See also AcFlow::ReleaseSpace method
AcFlow::ReserveSpace method

AcFlow::ResizeByConstrainedByContents method

Calls the ResizeByConstrained( ) method of AcVisualComponent. Subclasses of AcFlow must implement ResizeByConstrainedByContents( ) to ensure that the flow's contents constrain the amount by which the flow is resized.

Syntax

```
ResizeByConstrainedByContents( deltaWidth As AcTwips, deltaHeight As
    AcTwips )
```

Parameters **deltaHeight**
The amount, in twips, by which to resize the height of the flow.

deltaWidth
The amount, in twips, by which to resize the width of the flow.

See also AcVisualComponent::ResizeByConstrained method

AcFlow::ShiftFooterUp method

Moves the footer so it appears immediately after the last content frame in the flow. Derived classes override ShiftFooterUp( ), a pure virtual method in AcFlow, to specify where to place a footer frame in the flow. If you create a subclass of AcFlow, you must override ShiftFooterUp( ) to specify the implementation details.

Syntax

```
Sub ShiftFooterUp( footer As AcFrame )
```

Parameter **footer**
The page footer frame.

Class AcFrame

The base class for frames in a report design. Figure 7-57 shows the class hierarchy of AcFrame.

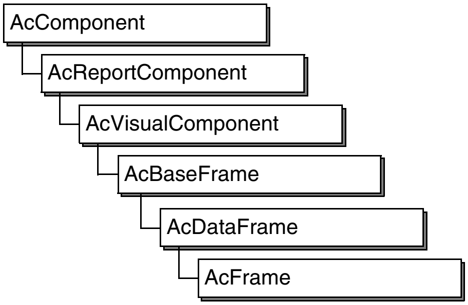


Figure 7-57 AcFrame

Description A frame is a container for visual components such as controls, charts, and other nested frames. Data controls must be placed in a frame. Constant controls can be placed in a frame or on a page.

In a report design, a frame and its contents are typically associated with a data row. For example, if a data row contains name, address, and telephone number, you can place in your report design a frame that contains three data controls for the data.

The framework uses a standard protocol for creating frames. For example, you can easily move a frame from the Content slot to the PageHeader slot of a report section.

Class protocol

A frame is contained by another component, such as a section or another frame. The frame's container calls the methods in AcFrame as shown in Table 7-44.

Table 7-44 Class protocol for AcFrame

Method	Task
New( )	Container instantiates the frame.
Start( )	Container starts the frame.
N/A	Frame instantiates all of its controls and nested frames.
Build( ) or BuildFromRow( )	Container builds the frame.

Table 7-44 Class protocol for AcFrame

Method	Task
Finish( )	Container finishes the frame and does any cleanup work.
N/A	Container places the frame on the page.

Subclassing AcFrame

Each time you drag a frame from a toolbox and drop it into the report design, e.Report Designer Professional creates a subclass of AcFrame. You can override methods in the subclass to do special processing or change default properties of the frame.

Properties

Table 7-45 lists AcFrame properties.

Table 7-45 AcFrame properties

Property	Type	Description
AutoSplit Vertical	AcAutoSplit	Specifies how the frame or control is split vertically over multiple pages: ■ DefaultSplitting. The default setting. If the frame contains a dynamic text control, splits the frame to maximize use of space within a flow. Otherwise, the frame is not split. ■ DoNotSplit. Does not split. Text that does not fit within a flow is truncated. ■ SplitIfNecessary. Splits the frame if it cannot fit as the first non-decoration frame in a flow. ■ SplitIfPossible. Splits to maximize use of space within a flow.
CanIncrease Height	Not applicable	Determines whether the height of the frame can increase as the height of its contents increases.
CanIncrease Width	Not applicable	Determines whether the width of the frame can increase as the width of its contents increases.
CanMoveLeft	Not applicable	Determines whether the frame can move left.
CanMoveUp	Not applicable	Determines whether the frame can move up.

(continues)

Table 7-45 AcFrame properties (continued)

Property	Type	Description
CanReduce Height	Not applicable	Determines whether the height of the frame can decrease as the height of its contents decreases.
CanReduce Width	Not applicable	Determines whether the width of the frame can decrease as the width of its contents decreases.
Custom DHTMLFooter	String	Enables use of custom browser scripting control in an HTML form. The PDF Converter ignores CustomDHTMLFooter.
Custom DHTML Header	String	Enables use of custom browser scripting control in an HTML form. The PDF Converter ignores CustomDHTMLHeader.
NoSplitBottom	AcTwips	The height of the area that must not be split at the bottom of the frame. The default value is 1".
NoSplitTop	AcTwips	The height of the area that must not be split at the top of the frame. The default value is 1".
SplitMargin Bottom	AcTwips	The margin between the bottom edge and the contents of segments of split frames. The margin is not applied to the last segment. The default value is 0".
SplitMargin Top	AcTwips	The margin between the top edge and the contents of segments of split frames. The margin is not applied to the first segment. The default value is 0".

Methods for Class AcFrame

Methods defined in Class AcFrame

AdjustContentVerticalGeometry, AutoSplitVertical, CustomDTMLFooter, CustomDHTMLHeader, GetBorderOrigin, GetBorderRect, GetBorderSize, NoSplitBottom, NoSplitTop, PageBreakAfter, PageBreakBefore, SplitMarginBottom, SplitMarginTop

Methods inherited from ClassAcBaseFrame

AddToAdjustSizeList, BindToFlow, FindContentByClass, FindContentByClassID, GetControl, GetControlValue, GetPageNumber, GetSearchValue, IsDataFrame, IsFooter, IsHeader, MakeContents, RebindToFlow, SearchAttributeName

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcFrame::AdjustContentVerticalGeometry method

Adjusts the height of the frame and the vertical position and height of the frame's contents.

Syntax Sub AdjustContentVerticalGeometry( )

AcFrame::AutoSplitVertical method

Returns the value of the AutoSplitVertical property, with which you can modify how the factory splits a frame or a dynamic text control. Table 7-46 lists the valid values for the AutoSplitVertical property.

Table 7-46 Valid values for the AutoSplitVertical property

Property	Description
DefaultSplitting	The factory's default behavior is to split a frame and its contents only if the frame contains at least one dynamic text control. This setting splits the frame contents and places the segments in the most space-efficient manner, ensuring that no segments are too small at the top and bottom of pages when the frame splits over multiple pages. For frames that contain dynamic text controls, this setting yields the same results as SplitIfPossible.
DoNotSplit	Only data that fits in the flow appears. The remaining data does not appear in the report. Use this setting to limit the frame to one page.
SplitIfNecessary	Splits the frame, excluding header and footer, only if the frame is the first one in the flow. Subsequent frames are placed on the next page, where again only the first frame is split, if necessary. Use this setting to minimize the number of split frames. This setting increases the amount of empty space on pages and, therefore, the number of pages.
SplitIfPossible	Splits the frame and its contents to maximize the use of space in the flow. Use this setting to minimize the number of pages. Using this setting, more frames split across pages.

Syntax Function AutoSplitVertical( ) As AcAutoSplit

AcFrame::CustomDHTMLFooter method

Enables use of custom browser scripting control in an HTML form. CustomDHTMLFooter( ) is called for report viewing using the DHTML viewer.

Syntax Function CustomDHTMLFooter( ) As String

See also AcFrame::CustomDHTMLHeader method

AcFrame::CustomDHTMLHeader method

Enables use of custom browser scripting control in an HTML form. CustomDHTMLHeader( ) is called for report viewing using the DHTML viewer.

Syntax Function CustomDHTMLHeader( ) As String

See also AcFrame::CustomDHTMLFooter method

AcFrame::GetBorderOrigin method

Returns the origin coordinates of the border. The origin coordinates define the upperleft position of the border.

Syntax Function GetBorderOrigin( ) As AcPoint

See also AcFrame::GetBorderRect method
AcFrame::GetBorderSize method

AcFrame::GetBorderRect method

Returns the origin, or upperleft, coordinates and size of the border. In the user interface, a rectangle defines the size and position of the border.

Syntax Function GetBorderRect( ) As AcRectangle

Returns A rectangle that defines the border of the frame.

See also AcFrame::GetBorderOrigin method
AcFrame::GetBorderSize method

AcFrame::GetBorderSize method

Returns the size of the content area of the frame.

Syntax Function GetBorderSize ( ) As AcSize

Returns The size, in twips, of the frame.

See also AcFrame::GetBorderOrigin method
AcFrame::GetBorderRect method

AcFrame::NoSplitBottom method

Returns the value of the NoSplitBottom property. NoSplitBottom specifies the height of the area that must not be split at the bottom of the frame, or the minimum height of the last segment.

Syntax Function NoSplitBottom( ) As AcTwips

Returns The value of the frame's NoSplitBottom property.

AcFrame::NoSplitTop method

Returns the value of the frame's NoSplitTop property. NoSplitTop specifies the height of the area that must not be split at the top of the frame, or the minimum height of the first segment.

Syntax Function NoSplitTop( ) As AcTwips

Returns The value of the frame’s NoSplitTop property.

AcFrame::PageBreakAfter method

Determines whether the frame is the last one on the current page.

Syntax Function PageBreakAfter( ) As Boolean

Returns True if the frame is the last one on the current page.
Otherwise, False.

AcFrame::PageBreakBefore method

Determines whether the frame is the first one on the current page.

Syntax Function PageBreakBefore( ) As Boolean

Returns True if the frame is the first one on the current page.
Otherwise, False.

AcFrame::SplitMarginBottom method

Returns the value of the frame’s SplitMarginBottom property. When a dynamic text control is split to fit on multiple pages, SplitMarginBottom specifies a blank area between the bottom edge of each segment, except the last, and its contents.

Syntax Function SplitMarginBottom( ) As AcTwips

Returns The value of the frame’s SplitMarginBottom property.

AcFrame::SplitMarginTop method

Returns the value of the frame’s SplitMarginTop property. When a dynamic text control is split to fit on multiple pages, SplitMarginTop specifies a blank area between the top edge of each segment, except the first, and its contents.

Syntax Function SplitMarginTop( ) As AcTwips

Returns The value of the frame’s SplitMarginTop property.

Class AcGroupSection

Groups related rows into a section based on a key column. Figure 7-58 shows the class hierarchy of AcGroupSection.

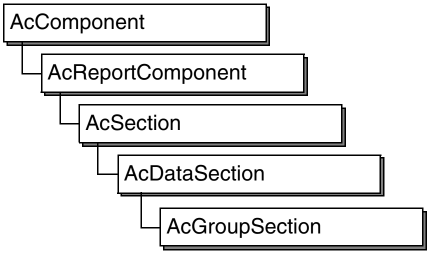


Figure 7-58 AcGroupSection

Description Reports often contain grouped data and can display subtotals for each group. For example, a report of customers can group the customers by state and provide a year-to-date summary of customer expenditures. AcGroupSection is the report component that creates groups in a report.

Grouping is based on a key column in your data row, which you identify by setting the Key property, and optionally, the GroupOn and GroupInterval properties. You use the GroupOn and GroupInterval properties to convert a key column in a data row into a value that is more suitable for your report. For example, if a key column contains the full date of sale for an item and you need to produce a quarterly sales report, use the GroupOn property to group the data by quarter. The framework extracts the date information from the key column, converts it to a calendar quarter, and creates groups based on calendar quarter. If you need to produce a bimonthly report, set GroupOn to GroupOnMonth and set GroupInteval to 2.

Because a group section is a data section, the group section inherits the Before, After, Page Header, Page Footer, and Content slots. Use the After slot to create subtotals over your group. Use the Content slot to process each row in the group. The Content can be a frame or another group section to create a nested group.

You always use a group section with a report section. If the report section uses a query data stream, the report section ensures that the query sorts the data rows as needed for the group sections. Specifically, the report section tells the query data stream to sort the data based on the columns identified in the Key properties of group sections associated with the report. If you had specified an Order By clause in your query, then the columns you specified appear after the key columns in the modified Order By clause.

To specify a group to use with a query data source, set the Key property to the name of a column in your data row. You must use the database column name, not the Basic variable name in the data row. The framework ensures the rows are sorted correctly.

If the data stream is other than a query, you must ensure that the data stream returns the rows sorted in the correct order as needed by the group sections. The framework cannot automatically sort the rows for you unless you specify the Key property for each group section. The Key property must be the name of an Actuate Basic variable in the data row.

To do a level break on a computed value, such as a range of account numbers, or a substring within a field, such as the area code portion of a phone number, you must create a column that represents the computed value in the data stream. For example, if you use a query data stream, you must create a computed column to hold the computed value that you want to be the key.

You can nest group sections. Each group is identified not only by its own key, but by the entire set of keys of any enclosing group sections. As such, when an outer group ends, all nested groups end also. For example, suppose you want a list of customers and their orders by state. You create the query, then create an outer group for states and an inner group for customers. Assume that the database contains both Alabama and Alaska, that both states have only one customer, each having the name “Smith.” The framework treats the customers as belonging to different groups. That is, the full key of the first customer is (“Alabama,” “Smith”) and the full key of the second customer is (“Alaska,” “Smith”).

Building a group section

Group sections nest within a report section that reads rows from a data stream. The report section passes each group section its contents by calling the content’s BuildFromRow( ) method. The group section uses this method to build up the group, one row at a time. When the enclosing section calls BuildFromRow( ) on a group section for the first time, the group section creates the Before and, optionally, the Page Header components. On the first row, BuildFromRow( ) calls the generated GetKey( ) method to identify and store the key value for this group.

For each row after the first, BuildFromRow( ) again calls GetKey( ) to determine the key value for that row. If the key value differs from the value that identifies the group, then the row is the first row of the next group. In this case, BuildFromRow( ) produces its After component, and returns False without processing the data row. When the enclosing section sees the return value False, the section starts a new group section to represent the new data group and passes the data row to the new group.

In summary, BuildFromRow( ) uses the following sequence of events:

- If the row is the first row, BuildFromRow( ) produces the Page Header and Before components as described in Class AcDataSection. BuildFromRow( ) records the row’s key value.
- If the row is not the first row, BuildFromRow( ) compares the row’s key value with the recorded key value. If the key values differ, BuildFromRow( ) produces the After and Page Footer components and returns False.

- If there is no current content component, BuildFromRow( ) calls NewContent( ) to create a content component.
- BuildFromRow( ) passes the row to the content's BuildFromRow( ) method. If the content accepts the row, then BuildFromRow( ) returns True.
- BuildFromRow( ) finishes the current content and instantiates a new one by calling NewContent( ).
- BuildFromRow( ) calls the content's BuildFromRow( ) method. This time, the content must accept the row. Then, BuildFromRow( ) returns True.

Variables

Table 7-47 lists AcGroupSection variables.

Table 7-47 AcGroupSection variables

Variable	Type	Description
KeyValue	Variant	The value of the key column for the group
KeyColumnName	String	The name of the key column for the group

Properties

Table 7-48 lists AcGroupSection properties.

Table 7-48 AcGroupSection properties

Property	Type	Description
GroupInterval	Function	GroupInterval works with GroupOn to control how data is grouped in the report. GroupInterval contains the number of values to group together. If you specify GroupOnPrefix, set GroupInterval to the number of characters in the prefix. The default value is 1.
GroupOn	AcGroupOn Type	GroupOn controls how data is grouped in the report. Valid values are: ■ GroupOnCustom. The developer builds a custom group key in the GetKeyValue( ) method. ■ GroupOnDay. Group key is the date excluding the time. Valid for key columns of Date data type only. ■ GroupOnHour. Group key is the full date and time excluding minutes and seconds. Valid for key columns of Date data type only.

(continues)

Table 7-48 AcGroupSection properties (continued)

Property	Type	Description
GroupOn *(continued)*	AcGroupOn Type *(continued)*	■ GroupOnInterval. Provides grouping for keys with data types other than Currency, Date, Double, Integer, Single, or String. ■ GroupOnMinute. Group key is the full date and time excluding seconds. Valid for key columns of the Date data type only. ■ GroupOnMonth. Group key is year and month. Valid for key columns of the Date data type only. ■ GroupOnPrefix. Group key is the first n characters of a key. Valid for columns of the String data type only. ■ GroupOnQuarter. Group key is a calendar quarter. Valid for key columns of the Date data type only. ■ GroupOnWeek. Group key is the full date converted to a week within a year. Valid for key columns of the Date data type only. ■ GroupOnYear. Group key is the year. Valid for key columns of the Date data type only. The default value is GroupOnEveryValue.
Key	Expression	The database column that identifies the group.

Methods for Class AcGroupSection

Methods defined in Class AcGroupSection

GetKeyString, IsSameKey

Methods inherited from Class AcDataSection

GetAfter, GetBefore, GetFirstPageFooter, GetFirstPageHeader, GetPageFooter, GetPageHeader, NewAfter, NewBefore, NewContent, NewPageFooter, NewPageHeader, OnEmptyGroup

Methods inherited from Class AcSection

CommittedToFlow, DeletePageFrame, FinishConnection, FinishFlow, FinishPage, GetComponentACL, GetCurrentRow, GetSearchValue, NewPage, ObtainConnection, PageBreakAfter, PageBreakBefore, SetSearchValue, SetSecurity, StartFlow, StartPage, StopAfterCurrentFrame, StopAfterCurrentRow, StopNow, TocAddComponent, TocAddContents

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcGroupSection::GetKeyString method

Returns the key value as a string.

Syntax Function GetKeyString( ) As String

Returns The string containing the group's key value.

Example In the following example, the user-defined method, GetKeyValue, on a control, returns the value of the group section key. This method calls GetKeyString( ) to return the key as a string value after the group section is located.

```
Function GetKeyValue( ) As String
' Get the key value of the Group Section containing this control.
   Dim myGroupSection As AcGroupSection
   Dim component As AcReportComponent
   Set component = GetContainer()
   Do While Not component Is Nothing
      If IsKindOf( component, "AcGroupSection") Then
         Exit Do
      End If
      Set component = component.GetContainer()
   Loop
   If Not component Is Nothing Then
      Set myGroupSection = component
      GetKeyValue = myGroupSection.GetKeyString()
   End If
End Function
```

AcGroupSection::IsSameKey method

Checks whether the group section key has changed by comparing the value of the current group section key and the prior group section key. Call IsSameKey( ) to use the results of a computation to determine whether to do a level break.

Syntax Function IsSameKey(curKey As Variant, prevKey As Variant) As Boolean

Parameters **curKey**
The current group section key.

prevKey
The prior group section key.

Returns True if the keys are equal.
False if the keys are not equal.

Class AcImageControl

Displays external images in a report. Figure 7-59 shows the class hierarchy of AcImageControl.

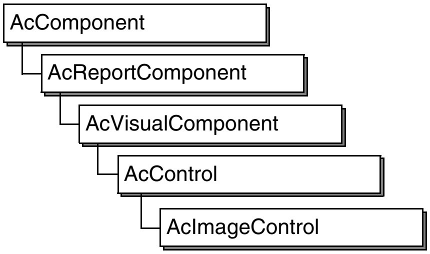


Figure 7-59 AcImageControl

Description Use AcImageControl to display an image in a report. You can display a static image file. Alternatively, if the image file name is in a data column, you can direct Actuate software to present different images for each data row, based on the contents of the data column. In this case, the size of the images must be the same.

If you distribute a report to users using e-mail or an Encyclopedia volume, use the Embedded property to include the image in the report at image design time or image run time. If you do not embed the image with the report you distribute, an X appears in place of the image. Table 7-49 lists the types of images that are supported.

Table 7-49 Supported image types

File type	Supported formats
BMP	1 bit per pixel.
	4, 8, or 24 bits per pixel (RLE encoding).
GIF	4 or 8 bits per pixel. Appears in DHTML reports only.
JPG	24 bits per pixel.
PCX	1, 4, 8, or 24 bits per pixel.
TGA	8, 16, or 24 bits per pixel (RLE encoding).
	32 bits per pixel with alpha channel.
TIFF	1 bit per pixel (uncompressed). Appears on Windows only.
	8, 16, or 24 bits per pixel (LZW compression).
	CCITT Fax Groups 3 and 4 compression.

To use an image in your report, complete the following tasks:

- Describe how to get the file name for the image.

  - If the image file name or URL is accessed from a data column, set the FileNameExp property to the name of the data column containing the image file name or URL or use the column in an expression.
  - If the name or URL of the image file is a static value, set the FileName property to the file name or URL of the image to display.
- Set the Embedded property to direct the framework when to include the image in the report. Valid values are:
  - ImageDesignTime. Include the image when the report object design (.rod) file compiles.
  - ImageFactoryTime. Include the image when the report object instance (.roi) file builds
  - ImageFactoryTimeSingle. Include the image only once when the report object instance (.roi) file builds, resulting in faster report generation and smaller ROI and PDF files.
  - ImageViewTime. Include the image when the ROI appears in the report viewer.
  - ImageViewTimeSingle. Include the image only once when the ROI appears in the report viewer, making PDF generation faster, and PDF files smaller.

  If you distribute a report to users through e-mail or through an Encyclopedia volume, set the image control's Embedded property to ImageDesignTime or ImageFactoryTime. If you do not set one of these values, an X appears in place of the image.

Properties

Table 7-50 lists AcImageControl properties.

Table 7-50 AcImageControl properties

Property	Type	Description
Embedded	AcImage EmbedType	The point at which to embed the image. Valid values are: ■ ImageDesignTime. e.Report Designer Professional retrieves the image and embeds it in the report object executable (.rox) file. ■ ImageFactoryTime. The Factory retrieves the image and embeds it in the report object instance (.roi) file. Use the FileName property to specify the name of the image file. ■ ImageFactoryTimeSingle. Embeds images that are used multiple times only once in an ROI file.

Table 7-50 AcImageControl properties

Property	Type	Description
Embedded *(continued)*	AcImage EmbedType *(continued)*	■ ImageViewTime. The view and print process retrieves the image when the user views or prints the report. ■ ImageViewTimeSingle. Images that are used more than once are retrieved only one time instead of multiple times. The default value is ImageDesignTime.
FileName	String	The name or URL of an image file to display.
FileName Exp	Expression	If the image is derived from a data column type the name of the data column enclosed in brackets. For example, type `"http://" + [catalog.itempic]` to use the itempic data column from the catalog table as the source for a URL to the image.

Methods for Class AcImageControl

Methods defined in Class AcImageControl

GetFileName

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcImageControl::GetFileName method

Returns the file name for the image to be displayed when the report is viewed or printed. Override GetFileName( ) to specify a custom file name for the image to be displayed. The default return value for GetFileName( ) is the value of the FileName member variable.

If the report runs locally, the file name can be relative or absolute. A relative file name must be relative to the directory that contains the report object instance (.roi) file. If the report runs on iServer, the file name must be absolute.

Syntax Function GetFileName( ) As String

Returns Name of the file containing the image.

Class AcIntegerControl

Displays an Integer value in a report. Figure 7-60 shows the class hierarchy of AcIntegerControl.

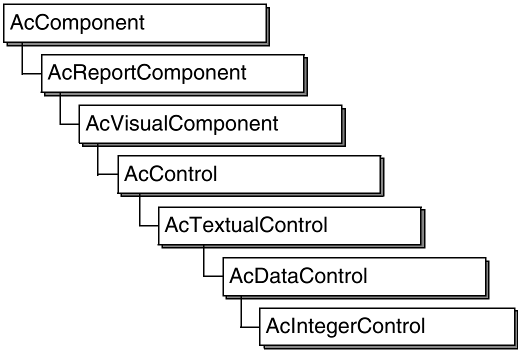


Figure 7-60 AcIntegerControl

Description Use AcIntegerControl to display an Integer value. You can also use a currency control or double control to display numeric values.

See also Class AcControl
Class AcCurrencyControl
Class AcDataControl
Class AcDoubleControl
Class AcTextualControl

Variable

Table 7-51 describes the AcIntegerControl variable.

Table 7-51 AcIntegerControl variable

Variable	Type	Description
DataValue	Integer	Stores the value of the control

Methods for Class AcIntegerControl

Methods inherited from Class AcDataControl

Format, GetGroupKey, IsSummary

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcIterator

The base class for all iterators. Figure 7-61 shows the class hierarchy of AcIterator.

Figure 7-61 AcIterator

Description

AcIterator provides the methods needed to work with iterators. Typically, you create an iterator in one of the following two ways:

- Call the NewIterator( ) method on your collection.
- Call the GetContentIterator( ) method on your report component.

These methods can return specialized iterators derived from AcIterator. Always handle the returned iterator as an instance of AcIterator.

Traversing a list

AcIterator provides two techniques for traversing a list of objects. You can use the GetNext( ) method to advance the iterator and return objects. You call the methods HasMore( ) and IsDone( ) to test whether more objects exist or you have reached the end of the list. You can also use MoveNext( ) to move the iterator and GetItem( ) to retrieve the item. The return value from MoveNext( ) tells you when you have reached the end of the list. The latter technique improves list processing because positioning is independent of retrieval. You can also combine MoveNext( ) with the SkipTo( ) method to position the iterator anywhere in the list.

You can position the iterator at a specific item in a list by specifying the item's position number. Items in the list are numbered sequentially starting with 1.

Updating items in a list

Iterators are valid only when iterating over a list that does not change. If you create an iterator over a list, then change the list, the operation of the iterator is unpredictable. If you must iterate over a list as it changes, you can create a copy of the list and iterate over the copy until the changes are complete. Alternatively, you can call the Restart( ) method on the iterator after you finish updating the list.

Examples

The following example shows how to create a list and traverse it in sequential order:

```
Dim iter As AcIterator
Dim obj As MyClass
Set iter = aCollection.NewIterator( )
Do While iter.HasMore( )
   Set obj = iter.GetNext( )
Loop
```

The following example shows how to create a list and traverse it using methods that help you process the list sequentially or randomly. The code calls the NewIterator( ) method on the collection to create the iterator.

```
Dim iter As AcIterator
Dim obj As MyClass
Set iter = aCollection.NewIterator( )
Do While iter.MoveNext( )
   Set obj = iter.GetItem( )
Loop
```

See also Class AcCollection
Class AcReportComponent

Methods for Class AcIterator

Methods defined in Class AcIterator

Copy GetItem, GetNext, GetPosition, HasMore, IsDone, MoveNext, Restart, SkipForwardTo, SkipTo, SkipToItem

AcIterator::Copy method

Copies this iterator. Use Copy( ) when the report must retain the state of the iterator. The copy has the same state as the original iterator.

Syntax Function Copy( ) As AcIterator

AcIterator::GetItem method

Returns the item to which the iterator points. GetItem( ) does not change the position of the iterator. This method can be called multiple times to retrieve the item at a given position.

You must not process lists using both GetNext( ) and GetItem( ) because the iterator positioning logic is different for the two methods.

Syntax Function GetItem( ) As AnyClass

Returns The object to which the iterator points.
Nothing if the iterator is not pointing to an object.

See also AcIterator::GetNext method
AcIterator::MoveNext method

AcIterator::GetNext method

Returns the next item in the list. After GetNext( ) returns the object, it advances the iterator to point to the next item in the list.

You must not process lists using both GetNext( ) and GetItem( ) because the iterator positioning logic is different for the two methods.

Syntax Function GetNext( ) As AnyClass

Returns The next item in the list.

See also AcIterator::GetItem method

AcIterator::GetPosition method

Returns the current position of the iterator. The items in the list have position number 1, 2, and so on. The framework positions new or restarted iterators before the start of the list, at position 0. GetPosition( ) returns the number of items in the list plus one if the iterator is positioned past the end of the list.

Syntax Function GetPosition( ) As Integer

Returns An integer indicating the item number in the list.

AcIterator::HasMore method

Determines whether there are more items in the list. This method is the inverse of IsDone( ). You can use HasMore( ) to detect if there are more items when you use GetNext( ) to retrieve items.

Syntax Function HasMore( ) As Boolean

Returns True if there are other items in the list.
False if there are no more items in the list.

See also AcIterator::GetNext method
AcIterator::IsDone method

AcIterator::IsDone method

Determines if there are more items in the list. This method is the inverse of HasMore( ). You can use IsDone( ) to detect if there are more items when you use GetNext( ) to retrieve items.

Syntax Function IsDone( ) As Boolean

Returns True if there are no more items in the list.
False if there are more items in the list.

See also AcIterator::HasMore method
AcIterator::GetNext method

AcIterator::MoveNext method

Moves the iterator to the next position in the list.

After you create or restart the iterator, you can call MoveNext( ) to position the iterator at the first item in the list. Then, you can call GetItem( ) to retrieve the item from the list.

Syntax Function MoveNext( ) As Boolean

Returns True if the next position is in the list.
False if the next position is past the end of the list.

See also AcIterator::GetItem method

AcIterator::Restart method

Positions the iterator before the first item in the list.

Syntax Sub Restart( )

AcIterator::SkipForwardTo method

Skips forward from the current node to the node that contains the specified object. Searches from the current node to the end of the list. The next call to GetNext( ) or GetItem( ) returns either the object or Nothing if the object is not in the list.

The preferred method is SkipToItem( ), which searches the entire list.

Syntax Sub SkipForwardTo(obj As AnyClass)

Parameter **obj**
The object to locate.

See also AcIterator::SkipTo method
AcIterator::SkipToItem method

AcIterator::SkipTo method

Places the iterator at the position of a specified object. The next call to GetItem( ) returns the object. If the object is not in the list, the position of the iterator does not change.

Syntax Sub SkipTo(obj As AnyClass)

Parameter **obj**
The object to locate.

Returns True if the object is in the list.
False if the object is in the list.

See also AcIterator::GetItem method

AcIterator::SkipToItem method

Searches the entire index to locate the object at which to reposition the iterator. If the object is not in the list, returns False and the position of the iterator does not change.

To search only from the current iterator position forward, use SkipForwardTo( ).

Syntax Function SkipToItem(obj As AnyClass) As Boolean

Parameter **obj**
The object to locate.

Returns True if the object is in the list.
False if the object is not in the list.

See also AcIterator::SkipForwardTo method
AcIterator::SkipTo method

Class AcLabelControl

Displays static, non-searchable text labels in a report. Figure 7-62 shows the class hierarchy of AcLabelControl.

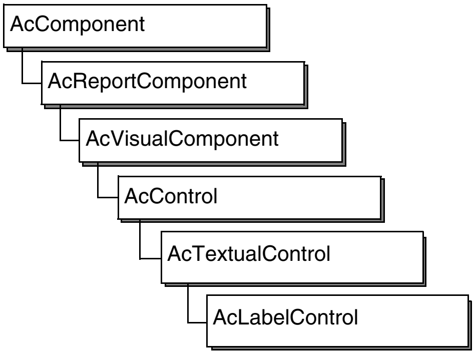


Figure 7-62 AcLabelControl

Description A label control is a constant control that is fully defined at design time. Use it to display a title or textual information that does not come from a data row. You specify the text to display using the Text property. To display string data from a data row, use AcTextControl instead.

There are no public methods defined specifically for AcLabelControl.

Variable

Table 7-52 describes the AcLabelControl variable.

Table 7-52 AcLabelControl variable

Variable	Type	Description
Text	String	Stores the text of the control

Property

Table 7-53 describes the AcLabelControl property.

Table 7-53 AcLabelControl property

Property	Type	Description
Text	String	Stores the text of the control

See also Class AcBaseFrame
Class AcDataFrame

Methods for Class AcLabelControl

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcLeftRightPageList

Builds a page list with alternating left-right pages. Figure 7-63 shows the class hierarchy of AcLeftRightPageList.

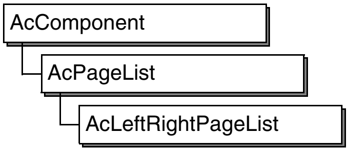


Figure 7-63 AcLeftRightPageList

Description The AcLeftRightPageList class provides a report format that has alternating left and right pages. AcLeftRightPageList has two component relationships, LeftPage and RightPage.

This class also has a FirstIsRight property that you can set in the Properties window. When you set FirstIsRight to True, AcLeftRightPageList starts the first page as a right-hand page. When you set FirstIsRight to False, AcLeftRightPageList starts the first page as a left-hand page.

There are no public methods defined specifically for AcLeftRightPageList.

Properties

Table 7-54 lists AcLeftRightPageList properties.

Table 7-54 AcLeftRightPageList properties

Property	Type	Description
FirstIsRight	Boolean	True if the first page in the page list is a right-hand page
LeftPage	AcPage	The page style to use for left-hand pages
RightPage	AcPage	The page style to use for right-hand pages

See also AcPageList
AcTitleBodyPageList

Methods for Class AcLeftRightPageList

Methods inherited from Class AcPageList

AddFrame, EjectPage, Finish, GetContentIterator, GetContents, GetCurrentFlow, GetCurrentPage, GetCurrentPageACL, GetEstimatedPageCount, GetFirstPage, GetLastPage, GetPage, GetPageCount, GetPageList, GetReport, HasPageSecurity, NeedCheckpoint, NeedHeight, NewPage, Start, UseAcceleratedCheckpoints

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcLinearFlow

Provides logic for adding frames to a flow that fills in one direction, either top-down or left-to-right. Figure 7-64 shows the class hierarchy of AcLinearFlow.

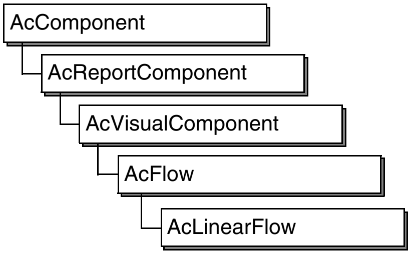


Figure 7-64 AcLinearFlow

Description AcLinearFlow is the abstract base class for flows that fill in one direction, either top-down or left-to-right. Top-down flows fill with frames in the standard top-down order, as described in AcTopDownFlow. The left-to-right flow fills with frames starting on the left side of the flow, with each subsequent frame placed to the right of the previous frame. When the report receives a frame that does not fit in the remaining space in the flow, the report advances to the next flow or page.

Variable

Table 7-55 describes the AcLinearFlow variable.

Table 7-55 AcLinearFlow variable

Variable	Type	Description
Alignment	AcFlowPlacement	Specifies how to align a frame within a flow. Values are: ■ FlowAlignLeftOrTop. Causes the frame to appear left-justified within the flow. ■ FlowAlignRightOrBottom. Causes the frame to appear right-justified. ■ FlowAlignCenter. Centers the frame in the flow. ■ FlowAlignCustom. Supports custom alignment. The framework uses the value of x in the Position property to align the frame in the flow.

Property

Table 7-56 describes the AcLinearFlow property.

Table 7-56 AcLinearFlow property

Property	Type	Description
Alignment	AcFlowPlacement	Specifies how to align frames. Valid values are: ■ FlowAlignLeftOrTop. Use this setting to cause the frame to appear left-justified within the flow. ■ FlowAlignRightOrBottom. Use this setting to cause the frame to appear right-justified in the flow. ■ FlowAlignCenter. Use this setting to center the frame in the flow. ■ FlowAlignCustom. Use this setting to do custom alignment. If you use custom alignment, the framework uses the value of x in the Position property to align the frame in the flow.

See also Class AcTopDownFlow

Methods for Class AcLinearFlow

Methods defined in Class AcLinearFlow

GetFreeSpace, GetInsideOrigin, GetInsideRect, GetInsideSize

Methods inherited from Class AcFlow

AddFooter, AddFrame, AddHeader, AddSubpage, AdjustFooter, CanFitFrame, CanFitHeight, GetFirstDataFrame, GetLastDataFrame, GetFreeSpace, GetInsideSize, IsEmpty, ReleaseSpace, ReserveSpace, ResetSpace, ResizeByConstrainedByContents, ShiftFooterUp

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcLinearFlow::GetFreeSpace method

Returns the free space in the flow. Free space is the area of the flow minus the space occupied by frames in the flow. Derived classes override the GetFreeSpace( ) method, a pure virtual method in AcLinearFlow, to specify how to return the available space. If you create a subclass from AcLinearFlow, you must override GetFreeSpace( ) to specify the implementation details.

Syntax Function GetFreeSpace( ) As AcSize

Returns The amount of unused space, in twips, available in the flow.

AcLinearFlow::GetInsideOrigin method

Gets the position of the inside area of the flow, relative to the upper left corner, or origin, of the flow.

Syntax Function GetInsideOrigin( ) As AcPoint

Returns The origin coordinates, in twips.

AcLinearFlow::GetInsideRect method

Gets the rectangle that defines the inside space of the flow, relative to the upper left corner, or origin, of the flow.

Syntax Function GetInsideRect( ) As AcRectangle

Returns A rectangle that defines the size of the flow's content.

AcLinearFlow::GetInsideSize method

Returns the size, in twips, of the content rectangle of the flow.

Syntax Function GetInsideSize( ) As AcSize

Returns The size of the flow's content.

Class AcLineControl

Displays a line in a report design. Figure 7-65 shows the class hierarchy of AcLineControl.

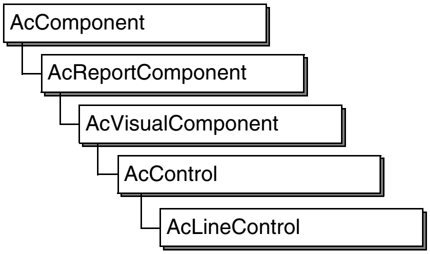


Figure 7-65 AcLineControl

Description A line control is a constant control that is fully defined at design time. The properties you are likely to set are Position, EndPosition, and LineStyle. Position specifies the starting coordinates of the line. EndPosition specifies the end coordinates. The *x*-coordinate is measured from the top left corner of the frame that contains the line control. The *y*-coordinate is measured from the top of the enclosing frame.

The Line Style properties support specifying the color, style, and width of the line. The length of the line is determined by the Position and EndPosition properties and the width by the width property under LineStyle.

There are no public methods defined specifically for AcLineControl.

Variables

Table 7-57 lists AcLineControl variables.

Table 7-57 AcLineControl variables

Variable	Type	Description
EndPosition	AcPoint	The end point of the line
LineStyle	AcLineStyle	The style of the line

Properties

Table 7-58 lists AcLineControl properties.

See also Class AcControl

Table 7-58 AcLineControl properties

Property	Type	Description
EndPosition	AcPoint	The end point of the line
LineStyle	AcLineStyle	The style of the line

Methods for Class AcLineControl

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcList

A base class that defines the list interface. Figure 7-66 shows the class hierarchy of AcList.

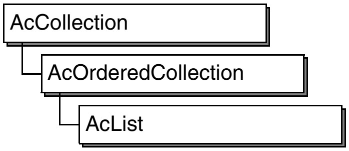


Figure 7-66 AcList

Description The AcList class is an abstract class that defines the list interface. Using inherited methods, this class provides a complete set of list functions, including the ability to add an item anywhere in the list, remove an item anywhere in the list, or obtain statistics about the contents of the list.

The framework defines one concrete subclass of AcList, AcSingleList, which implements a singly-linked list.

There are no public methods defined specifically for AcList.

Example The following example shows how to create a list and traverse it with a list iterator:

```
Class MyClass
   Dim counter As Integer
End Class

Sub Main ( )
   Dim list As AcList
   Dim obj As MyClass
   Dim count As Integer
   Dim iterator As AcIterator

   'Create a list
   Set list = New AcSingleList
   'Create an object and add it to the list
   Set obj = New MyClass
   obj.counter = 1
   list.AddToTail( obj )
   'Create a second object and append it to the list
   Set obj = New MyClass
   obj.counter = 2
   list.AddToTail( obj )
   'Count the number of objects in the list
   count = list.GetCount( )
   MsgBox "Number of objects in the list: " & count
   'Create a list iterator and get each object in the list
   Set iterator = New AcSingleListIterator( list )
   Do While iterator.HasMore( )
      Set obj = iterator.GetNext( )
      MsgBox "The position of this object in the list:" &
         obj.counter
   Loop

   'Delete the objects from the list
   list.RemoveHead( )
   list.RemoveHead( )
End Sub
```

See also Class AcCollection
Class AcOrderedCollection
Class AcSingleList

Methods for Class AcList

Methods inherited from Class AcOrderedCollection

AddToHead, AddToTail, Copy, GetAt, GetHead, GetIndex, GetTail, InsertAfter, InsertAt, InsertBefore, RemoveHead, RemoveTail, SetAt

Methods inherited from Class AcCollection

Compare, Contains, Copy, FindByValue, GetCount, IsEmpty, NewIterator, Remove, RemoveAll

Class AcMSSQLConnection

Establishes a connection to a DB2 database. Figure 7-67 shows the class hierarchy of AcMSSQLConnection.

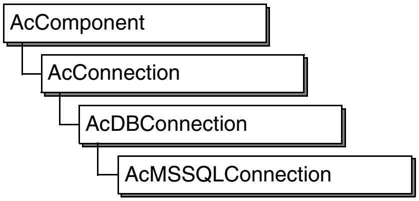


Figure 7-67 AcMSSQLConnection

Description Use the AcMSSQLConnection class to establish a connection to a Microsoft SQL database. The report must set the server name, user name, and password prior to connecting. After connecting, the report must not change these values.

There are no public methods defined specifically for AcMSSQLConnection.

Variables

Table 7-59 lists AcMSSQLConnection variables.

Table 7-59 AcMSSQLConnection variables

Variable	Type	Description
DllPath	String	The name of the DLL providing the client database
Password	String	The client password for the connection
ServerName	String	The client server name for the connection
UserName	String	The client user name for the connection

Properties

Table 7-60 lists AcMSSQLConnection properties.

Table 7-60 AcMSSQLConnection properties

Property	Type	Description
DllPath	String	The name of the DLL providing the client database
Password	String	The client password for the connection
ServerName	String	The client server name for the connection
UserName	String	The client user name for the connection

About MS-SQL data types

Table 7-61 describes the default conversion between MS-SQL and Actuate data types.

Table 7-61 Default mapping of MS-SQL to Actuate data types

DB2 data type	Maps to
Binary	Actuate String.
Bit	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.
Char	Actuate String.
DateTime	Actuate Date. Can also map to Actuate String.
Decimal	Actuate Double. Can also map to Actuate Integer, Long, Single, or String.
Int	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.
Moneyc	Actuate Currency. Can also map to Actuate Double, Integer, Long, Single, or String.
Numeric	Actuate Double. Can also map to Actuate Integer, Long, Single, or String.
Smallint	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.
SmallDateTime	Actuate Date. Can also map to Actuate String.
SmallMoney	Actuate Currency. Can also map to Actuate Double, Integer, Long, Single, or String.
Tinyint	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.
Varbinary	Actuate String.
Varchar	Actuate String.

Actuate software accesses MS-SQL databases using the DB-library API. The DB-library API returns 255 characters of data. If the data column size exceeds 255 characters, the remainder of the database column is truncated.

About queries

MS-SQL uses identifiers to name SQL server objects, such as servers and databases, and database objects, such as tables, views, columns, and procedures. Identifiers can include special characters and reserved words. You must enclose any identifier that contains a special character or reserved word in quotation marks. For example, if your table name contains blanks, enclose the table name identifier with quotation marks in the SQL query, as shown in the following example:

```
SELECT * FROM "New York office"
```

To enable the use of quoted identifiers by MS-SQL DB-library API, Actuate software issues the following command at the start of each session:

```
Set QUOTED_IDENTIFIER ON
```

Methods for Class AcMSSQLConnection

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Methods inherited from Class AcConnection

Connect, Disconnect, IsConnected, RaiseError

Methods inherited from Class AcDBConnection

GetGeneralError, GetGeneralErrorText, GetSpecificError, GetSpecificErrorText, Prepare

Class AcMultipleInputFilter

A base class for data filters, one type of data adapter. AcMultipleInputFilter accepts input from any number of data adapters, processes the data, then passes the data to the next data adapter or to the report. Figure 7-68 shows the class hierarchy of AcMultipleInputFilter.

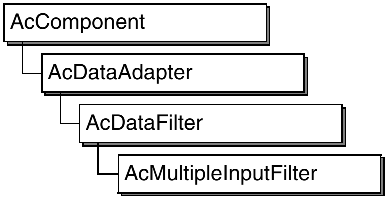


Figure 7-68 AcMultipleInputFilter

Description AcMultipleInputFilter defines the mechanism for filtering and sorting data from multiple data sources. Multi-input data filters work with data sources to produce and deliver data rows to the report.

A multi-input data filter receives input from other data adapters, either data sources or other data filters. You can use the design perspective to specify the input adapters that provide input rows to the filter.

As Figure 7-69 shows, the data sources can retrieve data from multiple input sources. For example, one data source can receive data from an SQL database while another data source receives data from an ODBC database.

To implement the filter algorithm, you must override the Fetch( ) method.

Working with the input adapters

The multiple input filter creates, opens, reads, and closes a set of one or more input adapters. You specify these adapters by dropping them into the Input slot in Report Structure. The multiple input filter instantiates, opens, and closes the adapters. You must write code in the Fetch( ) method to work with the input adapters.

The Start( ) method of the multiple input filter instantiates each of the input adapters you specify in the Input slot in Report Structure. These adapters are in an AcList called InputAdapters. You can also add adapters programmatically by using the methods on AcList. Start( ) then iterates over these adapters to give each adapter its connection, then starts the adapter. Similarly, Finish( ) iterates over each adapter to close the input adapter.

You must implement the Fetch( ) method to work with the adapters. When you implement Fetch( ), you can use the methods on AcList to work with the list of adapters. Also, you can use an iterator to loop over the adapters. The adapters appear in the list in the same order that they appeared in Report Structure.

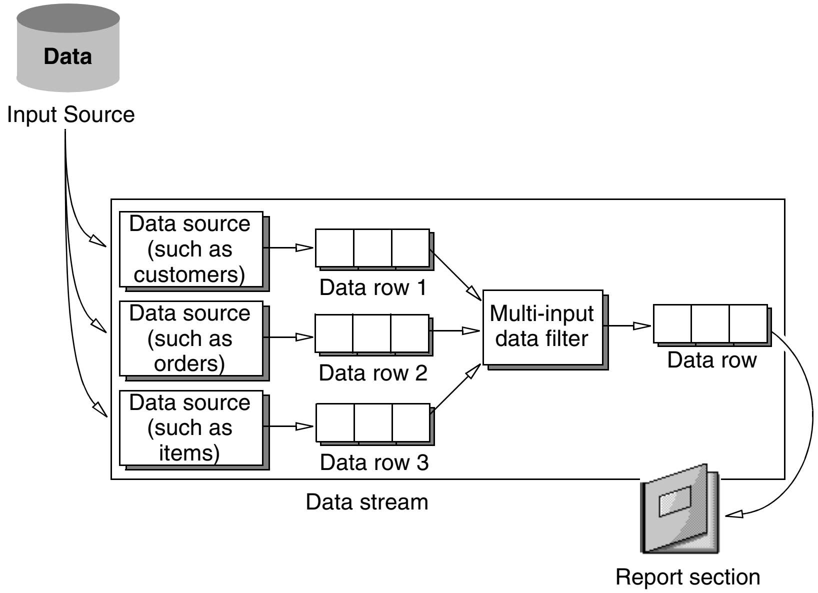


Figure 7-69 Retrieving data from multiple data sources

You can build many kinds of filters using this class. For example, you can create a subclass to:

- Concatenate the rows from each of the input filters. This type of filter is a union filter.
- Join or merge rows from one input adapter with those from a second input adapter. This type of filter is a merge filter.
- Return all rows from one adapter, except those that appear in another adapter. This type of filter is a subtraction filter.

Subclassing AcMultipleInputFilter

Typically, you subclass AcMultipleInputFilter and take the following steps to create a custom filter:

- Override Fetch( ) to specify how to process the data.
- Optionally, override Start( ) to specify a different way to create the input adapters.

Variable

Table 7-62 describes the AcMultipleInputFilter variable.

Table 7-62 AcMultipleInputFilter variable

Variable	Type	Description
InputAdapters	AcList	A list of the input adapters

Example The following example overrides Start( ), Fetch( ), and Finish( ) to create a union filter. It also defines two variables. The first variable is UnionIter of type AcIterator. The second variable is CurrentInput of type AcDataAdapter.

```
Function Start( ) As Boolean
   ' Start the multiple input filter
   Start = Super::Start( )
   If Not Start Then
      Exit Function
   End If

   ' Keep track of the input adapter from which to read.
   Set UnionIter = InputAdapters.NewIterator( )
   Set CurrentInput = UnionIter.GetNext( )
End Function

Function Fetch( ) As AcDataRow
   Do While True

      ' If all the adapters have been read, then just return
      ' Nothing.
      If CurrentInput Is Nothing Then
         Exit Function
      End If

      ' Try to read the next row from the current input adapter.
      Set Fetch = CurrentInput.Fetch( )

      ' If there is a row available, then return it.
      If Not Fetch Is Nothing Then
         Exit Function
      End If

      ' Move to the next input adapter.
      Set CurrentInput = UnionIter.GetNext( )
   Loop
End Function

Sub Finish( )
   ' Delete the iterator created earlier
   Set UnionIter = Nothing
   ' Finish the multiple input filter.
   Super::Finish( )
End Sub
```

See also Class AcDataAdapter
Class AcSingleInputFilter

Methods for Class AcMultipleInputFilter

Methods defined in Class AcMultipleInputFilter

GetInputCount, NewInputAdapter

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcMultipleInputFilter::GetInputCount method

Counts the number of adapters that provide input.

Syntax Function GetInputCount( ) As Integer

Returns The number as an integer.

AcMultipleInputFilter::NewInputAdapter method

Creates an input adapter. The Start( ) method calls NewInputAdapter( ) to create the input adapters. Override NewInputAdapter( ) to specify a different way to create an input adapter. The default behavior for this method is to instantiate the adapters you place in the Input slot in Report Structure.

Syntax Sub NewInputAdapter( )

Class AcObjectArray

Creates a dynamic array of objects. Figure 7-70 shows the class hierarchy of AcObjectArray.

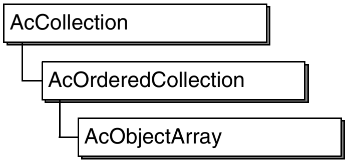


Figure 7-70 AcObjectArray

Description An object array is an ordered collection implemented as an array. AcObjectArray creates a resizable array of objects. The object array can be more efficient than an AcSingleList collection if a report traverses the objects in the collection frequently. Random access, additions to the end of the array and removals from the end of the array are efficient. Additions or removals at the head of the array are not efficient. This collection is not efficient for use as a queue.

Array indexes start at 1 and end at the GetCount( ) value. If you try to retrieve a value beyond the bounds of an array using GetAt( ), the framework returns a run-time error. You can, however, use SetAt( ) to set the value of an array beyond its current upper bound. SetAt( ) resizes the array as needed.

To reduce memory allocations, the array expands its internal storage in predefined increments. The default increment value is 10. Whenever the object array needs to expand, the framework allocates 10 slots. If you expect an array to grow in larger amounts, you can use SetGrowthIncrement( ) to increase the increment value.

See also Class AcCollection
Class AcIterator
Class AcOrderedCollection

Methods for Class AcObjectArray

Methods defined in Class AcObjectArray

RemoveAt, RemoveEmptyEntries, ResizeBy, ResizeTo, SetGrowthIncrementMethod

Methods inherited from Class AcOrderedCollection

AddToHead, AddToTail, Copy, GetAt, GetHead, GetIndex, GetTail, InsertAfter, InsertAt, InsertBefore, RemoveHead, RemoveTail, SetAt

Methods inherited from Class AcCollection

Compare, Contains, Copy, FindByValue, GetCount, IsEmpty, NewIterator, Remove, RemoveAll

AcObjectArray::RemoveAt method

Removes the object at a specific location in the array.

Syntax Function RemoveAt(posn As Integer) As AnyClass

Parameter **posn**
The location in the array of the object to remove.

See also AcObjectArray::RemoveEmptyEntries method

AcObjectArray::RemoveEmptyEntries method

Scans through the array to remove slots that contain Nothing. Reduces the count by the number of empty slots removed. This method expedites removing a group of items from the array. Iterate through the array and replace the items to remove with Nothing. Then call RemoveEmptyEntries( ) to remove those entries. This technique is much faster than calling Remove( ) or RemoveAt( ) for each item.

Syntax Sub RemoveEmptyEntries( )

AcObjectArray::ResizeBy method

Resets the size of the array by a specific number of slots.

Syntax Sub ResizeBy(delta As Integer)

Parameter **delta**
The amount by which to resize the array.

AcObjectArray::ResizeTo method

Resets the size of the array to a specific number of slots. You typically do not need to explicitly resize the array because the array methods do so automatically. For example, when you use SetAt( ) to place an object in an array location and you specify a location beyond the current size, SetAt( ) automatically resizes the array.

To increase or decrease the array size by a specific amount, use ResizeBy( ).

Syntax Sub ResizeTo(newSize As Integer)

Parameter **newSize**
The new size of the array.

See also AcObjectArray::ResizeBy method
AcOrderedCollection::SetAt method

AcObjectArray::SetGrowthIncrement method

Sets the number of slots to add to the array each time the array expands. By default, the array expands its internal storage by 10 when the array grows. Use the SetGrowthIncrement( ) method to increase the default increment value if you expect your array to expand in larger amounts.

Syntax Sub SetGrowthIncrement(incr As Integer)

Parameter **incr**
The number of slots to add to the array whenever the array expands.

Class AcOdaConnection

Establishes a connection to an Open Data Access (ODA) driver. Figure 7-71 shows the class hierarchy of AcOdaConnection.

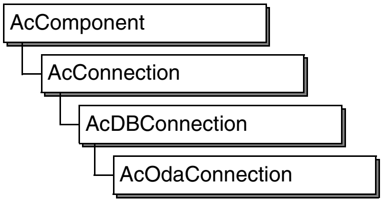


Figure 7-71 AcOdaConnection

Description Use the AcOdaConnection class to establish a connection to an open data access driver.

For information about converting from a native ODA data type to an Actuate ODA data type, see "About ODA data types," later in this chapter.

Properties

Table 7-63 lists AcOdaConnection properties.

Table 7-63 AcOdaConnection properties

Property	Type	Description
DriverName	String	The name of the ODA driver for the connection. This name is specified in the ODA configuration file.
OdaInterfaceName	String	The run-time interface name for the connection type as defined in the ODA driver. The ODA driver uses this value during report generation to create an instance of the connection. The property is optional. If the ODA driver supports only one type of connection, the value is an empty string.

Methods for Class AcOdaConnection

Methods defined in Class AcOdaConnection

SetProperties, SetRuntimeProperties

Methods inherited from Class AcDBConnection

GetGeneralError, GetGeneralErrorText, GetSpecificError, GetSpecificErrorText, Prepare

Methods inherited from Class AcConnection

Connect, Disconnect, IsConnected, RaiseError

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcOdaConnection::SetProperties method

Sets the value of a property variable to the value the user sets.

Syntax Sub SetProperties( )

AcOdaConnection::SetRuntimeProperties method

Calls the SetConnectionProperty( ) method to assign a value to each run-time property of the connection.

You typically call this method before opening a connection.

If you override SetRuntimeProperties( ), you must specify all properties required for report generation before the connection is established.

Syntax Sub SetRuntimeProperties( )

Class AcOdaSource

Creates an Open Data Access (ODA) source object. Figure 7-72 shows the class hierarchy of AcOdaSource.

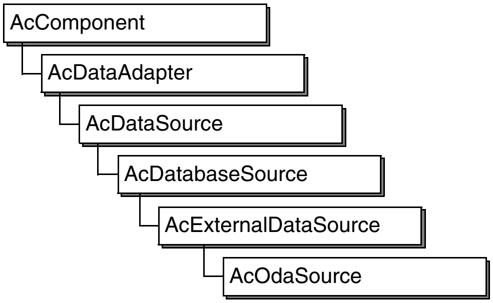


Figure 7-72 AcOdaSource

Description Use the AcOdaSource class to create an object for an ODA data source.

Variables

Table 7-64 lists AcOdaSource variables.

Table 7-64 AcOdaSource variables

Variable	Type	Description
CommandText	String	The text of the data stream's command or query to execute.
FetchLimit	Integer	The maximum number of rows to retrieve from a data source. To retrieve all rows, specify 0. The default value is 0 rows. Applies only if the ODA driver supports a fetch limit.
OdaSourceType	String	The type of ODA data source as defined in the ODA driver. The ODA driver uses this value to create an instance of a statement for report generation.
ResultSetName	String	The name of the primary result set of the data source. Applies only if the ODA driver supports referencing a result set by name. If the ODA driver does not support referencing a result set by name, the value is an empty string.

About ODA data types

The ODA driver must specify the conversion between its native data types and ODA data types. Actuate software converts each ODA data type to an Actuate data type. Table 7-65 describes the default conversion between ODA and Actuate data types.

Table 7-65 Default mapping of ODA data types to Actuate data types

ODA data type	Maps to
Char	Actuate String.
Date	Actuate Date. Also maps to Actuate String.
Decimal	Actuate Currency. Also maps to the Actuate Double and String data types.
Double	Actuate Double. Also maps to the Actuate Currency and Integer data types.
Integer	Actuate Integer. Also maps to the Actuate Double, Currency, and String data types.
Time	Actuate String.

Methods for Class AcOdaSource

Methods defined in Class AcOdaSource

ClearSortKeys, Commit, GetOutputParameter, GetOutputParameterAsType, GetOutputParameters, Rollback, SetInputParameter, SetInputParameters, SetRuntimeProperties, SetStatementAttributes, StartNextSet

Method inherited from Class AcExternalDataSource

ObtainCommand

Methods inherited from Class AcDatabaseSource

BindDataRow, BindStaticParameters, GetCursor, GetDBConnection, GetPreparedStatement, OpenCursor, SetStatementProperty

Methods inherited from Class AcDataSource

HasFetchedLast

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcOdaSource::ClearSortKeys

Removes all previously assigned dynamic sort keys.

Syntax Sub ClearSortKeys( )

AcOdaSource::Commit method

Commits all outstanding transactions on the specified ODA connection. This method applies only if the ODA driver supports this feature.

Syntax Sub Commit(connection As AcDBConnection)

Parameter **connection**
The ODA connection object on which to commit the transactions.

AcOdaSource::GetOutputParameter method

Retrieves the output value of a scalar, structure, or table output parameter in its default Actuate Basic data type.

For a table parameter that returns multiple rows, allocate a cursor to the parameter to retrieve the data.

For a structure or table parameter, you must use the same type of reference for the parameter and the field. For example, you cannot refer to the parameter by name and the field by position.

If you call GetOutputParameter( ) multiple times for a structure or table parameter of a command, you must use the same type of reference in every call for all structure or table parameters. For example, you cannot call GetOutputParameter( ) referring to a parameter and field by name, then call GetOutputParameter( ) again referring to the parameter or field by position.

You must use a type of reference the ODA driver supports.

Syntaxes For a scalar parameter:

Function GetOutputParameter(parameterName As String) As Variant

Function GetOutputParameter(parameterId As Integer) As Variant

For a structure or table parameter:

Function GetOutputParameter(groupParamName As String, fieldName
    As String) As Variant

Function GetOutputParameter(groupParamId As Integer, fieldId As Integer)
    As Variant

Parameters

parameterName
The name of the scalar parameter whose value to retrieve.

parameterId
The position of the scalar parameter whose value to retrieve. The position is 1-based.

groupParamName
The name of the structure or table parameter.

fieldName
The name of the field of a structure parameter or the column of a table parameter whose value to retrieve.

groupParamId
The position of the structure or table parameter. The position is 1-based.

fieldId
The position of the field of a structure parameter or the column of a table parameter whose value to retrieve. The position is 1-based.

AcOdaSource::GetOutputParameterAsType method

Retrieves the output value of a scalar, structure, or table output parameter and converts that value to the specified Actuate data type. For a structure or table parameter, you must use the same type of reference for the parameter and the field. For example, you cannot refer to the parameter by name and the field by position.

If you call GetOutputParameterAsType( ) multiple times for a structure or table parameter of a command, you must use the same type of reference in every call for all structure or table parameters. For example, you cannot call GetOutputParameterAsType( ) referring to the parameter and field by name, then callGetOutputParameterAsType( ) again referring to the parameter or field by position.

You must use a type of reference that the ODA driver supports.

Syntaxes

For a scalar parameter:

Function GetOutputParameterAsType(parameterName As String, vbType
    As Integer) As Variant

Function GetOutputParameterAsType(parameterId As Integer, vbType
    As Integer) As Variant

For a structure or table parameter:

Function GetOutputParameterAsType(groupParamName As String, fieldName As String, vbType As Integer) As Variant

Function GetOutputParameterAsType(groupParamId As Integer, fieldId As Integer, vbType As Integer) As Variant

Parameters **parameterName**
The name of the scalar parameter whose value to retrieve.

parameterId
The position of the scalar parameter whose value to retrieve. The position is 1-based.

groupParamName
The name of the structure or table parameter.

fieldName
The name of the field of a structure parameter or the column of a table parameter whose value to retrieve.

groupParamId
The position of the structure or table parameter. The position is 1-based.

fieldId
The position of the field of a structure parameter or the column of a table parameter whose value to retrieve. The position is 1-based.

vbType
The Actuate type to which to convert the output value. Valid values are:

- V_CURRENCY
- V_DATE
- V_DOUBLE
- V_INTEGER
- V_SINGLE
- V_STRING

For information about converting ODA data types to Actuate data types, see “About ODA data types,” earlier in this chapter.

AcOdaSource::GetOutputParameters method

Calls GetOutputParameter( ) to retrieve the output value of each scalar, structure, and table output parameter of the data stream’s statement. These parameters are defined in the last ODA design session response.

If you override GetOutputParameters( ), you must specify the processing of each applicable output parameter. The output value of each parameter should be available after the Start( ) method executes, which makes the values available in the Fetch( ) method.

Syntax Sub GetOutputParameters( )

AcOdaSource::Rollback method

Rolls back all outstanding transactions on the specified ODA connection. This method applies only if the ODA driver supports rollbacks.

Syntax Sub Rollback(connection As AcDBConnection)

Parameter **connection**
The ODA connection object.

AcOdaSource::SetInputParameter method

Assigns an input value to a specified scalar input parameter, a field of a structure input parameter, or a column of a table input parameter. For a structure or table parameter, you must use the same type of reference for the parameter and the field. For example, you cannot refer to the parameter by name and the field by position.

If you call SetInputParameter( ) multiple times for a structure or table parameter of a command, you must use the same type of reference in every call for every structure or table parameter. For example, you cannot call SetInputParameter( ) referring to the parameter and field by name, then call SetInputParameter( ) again referring to the same or a different parameter and field by position.

You must use a type of reference that the ODA driver supports.

Syntaxes For a scalar parameter:

Sub SetInputParameter(parameterName As String, value As Any)

Sub SetInputParameter(parameterId As Integer, value As Any)

For a structure or table parameter:

Sub SetInputParameter(groupParamName As String, fieldName As String, value As Any)

Sub SetInputParameter(groupParamId As Integer, fieldId As Integer, value As Any)

Parameters **parameterName**
The name of the scalar parameter to which to assign a value.

parameterId
The position of the scalar parameter to which to assign a value. The position is 1-based.

groupParamName
The name of the structure or table parameter.

fieldName
The name of the field of a structure parameter or the column of a table parameter to which to assign a value.

groupParamId
The position of the structure or table parameter. The position is 1-based.

fieldId
The position of the field of a structure parameter or the column of a table parameter to which to assign a value. The position is 1-based.

value
The value to assign.

AcOdaSource::SetInputParameters method

Calls SetInputParameter( ) to assign an input value to each scalar, structure, and table input parameter. These parameters are defined in the last ODA design session response. You typically call this method before executing the command or allocating a cursor.

If you override this method, you must specify all input parameter values before the command executes.

Syntax Sub SetInputParameters( )

AcOdaSource::SetRuntimeProperties method

Calls the SetStatementProperty( ) method to assign a value to run-time properties. You typically call this method before executing the command or allocating a cursor.

If you override SetRuntimeProperties( ), you must specify all properties and their values required at run time.

Syntax Sub SetRuntimeProperties( )

AcOdaSource::SetStatementAttributes method

Sets attributes on the prepared statement before executing the statement or allocating a cursor.

Syntax Sub SetStatementAttributes( )

AcODASource::StartNextSet method

Starts the next result set on the allocated cursor if the result set is not referenced by name. This method implicitly closes the current result set and removes any cursor bindings that the BindColumn( ) method set. You must bind columns again and fetch from the cursor after starting the next result set.

Syntax Function StartNextSet(aCursor As AcDBCursor) As Boolean

Parameter **aCursor**
The cursor for which to start the result set.

Class AcODBCConnection

Establishes a connection to an Open Database Connectivity (ODBC) database. Figure 7-73 shows the class hierarchy of AcODBCConnection.

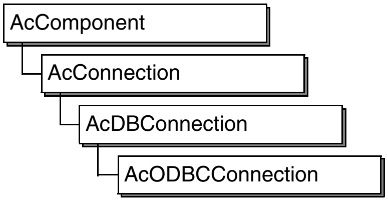


Figure 7-73 AcODBCConnection

Description Use the AcODBCConnection class to establish a connection to an ODBC database. The report must set the connection string, data source, user name, and password prior to connecting. After connecting, the report must not change these values.

There are no public methods defined specifically for AcODBCConnection.

Variables

Table 7-66 lists AcODBCConnection variables.

Table 7-66 AcODBCConnection variables

Variable	Type	Description
Connection String	String	Any additional text that ODBC needs to establish its connection
DataSource	String	The ODBC data source
DllPath	String	The name of the DLL providing the client database
Password	String	The client password for the connection
UserName	String	The client user name for the connection

Properties

Table 7-67 lists AcODBCConnection properties.

Table 7-67 AcODBCConnection variables

Property	Type	Description
Connection String	String Variable	Any additional text that ODBC needs to establish its connection.

Table 7-67 AcODBCConnection variables

Property	Type	Description
DataSource	String Variable	The ODBC data source.
DllPath	String	The name of the .dll providing client database. Default is the most common name used for the connectivity .dll provided by ODBC.
Password	String Variable	The client password for the connection.
UserName	String Variable	The client user name for the connection.

About ODBC data types

Table 7-68 describes the default conversion between ODBC and Actuate data types.

Table 7-68 Default mapping between ODBC and Actuate data types

ODBC data type	Maps to
Bigint	Actuate Double. Can also map to Actuate Currency, Integer, Long, Single, or String.
Binary	Actuate String.
Bit	Actuate Integer. Can also map to Actuate Double, Long, Single, or String.
Char	Actuate String.
Date	Actuate Date. Can also map to Actuate String.
Decimal	Actuate Double. Can also map to Actuate Currency, Integer, Long, Single, or String.
Double	Actuate Double. Can also map to Actuate Currency, Single, or String.
Float	Actuate Double. Can also map to Actuate Currency, Single, or String.
Guid	Actuate String.
Integer	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.
Interval_Day	Actuate Integer. Can also map to Actuate Double, Long, Single, or String.

(continues)

Table 7-68 Default mapping between ODBC and Actuate data types (continued)

ODBC data type	Maps to
Interval_Day_ To_Hour	Actuate String. Can also map to Actuate Double, Integer, Long, or Single.
Interval_Day_ To_Minute	Actuate String. Can also map to Actuate Double, Integer, Long, or Single.
Interval_Day_ To_Second	Actuate String. Can also map to Actuate Double, Integer, Long, or Single.
Interval_Hour	Actuate Integer. Can also map to Actuate Double, Long, Single, or String.
Interval_Hour_ To_Minute	Actuate String. Can also map to Actuate Double, Integer, Long, or Single.
Interval_Hour_ To_Second	Actuate String. Can also map to Actuate Double, Integer, Long, or Single.
Interval_Minute	Actuate Integer. Can also map to Actuate Double, Long, Single, or String.
Interval_Minute_ To_Second	Actuate String. Can also map to Actuate Double, Integer, Long, or Single.
Interval_Month	Actuate Integer. Can also map to Actuate Double, Long, Single, or String.
Interval_Second	Actuate Integer. Can also map to Actuate Double, Long, Single, or String.
Interval_Year	Actuate Integer. Can also map to Actuate Double, Long, Single, or String.
Interval_Year_ To_Month	Actuate String. Can also map to Actuate Double, Integer, Long, or Single.
Longvarbinary	Actuate String.
Longvarchar	Actuate String.
Numeric	Actuate Double. Can also map to Actuate Currency, Integer, Long, Single, or String.
Real	Actuate Single. Can also map to Actuate Currency, Double, or String.
Small_Int	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.
Time	Actuate String.
Timestamp	Actuate Date. Can also map to Actuate String.
Tinyint	Actuate Integer. Can also map to Actuate Currency, Double, Long, Single, or String.

Table 7-68 Default mapping between ODBC and Actuate data types (continued)

ODBC data type	Maps to
Type_Date	Actuate Date. Can also map to Actuate String.
Type_Time	Actuate String.
Type_Timestamp	Actuate Date. Can also map to Actuate String.
Varbinary	Actuate String.
Varchar	Actuate String.
Wchar	Actuate String.
Wlongvarchar	Actuate String.
Wvarchar	Actuate String.

Methods for Class AcODBCConnection

Methods inherited from Class AcDBConnection

GetGeneralError, GetGeneralErrorText, GetSpecificError, GetSpecificErrorText, Prepare

Methods inherited from Class AcConnection

Connect, Disconnect, IsConnected, RaiseError

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcOracleConnection

Establishes a connection to an Oracle database. Figure 7-74 shows the class hierarchy of AcOracleConnection.

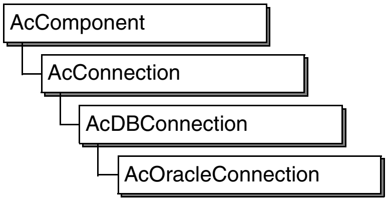


Figure 7-74 AcOracleConnection

Description Use the AcOracleConnection class to establish a connection to an Oracle database. The report must set the server name, user name, and password prior to connecting. After connecting, the report must not change these values.

There are no public methods defined specifically for AcOracleConnection.

Variables

Table 7-69 lists AcOracleConnection variables.

Table 7-69 AcOracleConnection variables

Variable	Type	Description
DllPath	String	The name of the DLL providing the client database
Password	String	The client password for the connection
HostString	String	The Oracle server name for the connection
UserName	String	The client user name for the connection

Properties

Table 7-70 lists AcOracleConnection properties.

Table 7-70 AcOracleConnection properties

Property	Type	Description
DbInterface	String	The name of the DLL providing the client database, acorcl90.dll
Password	String Variable	The client password for the connection

Table 7-70 AcOracleConnection properties

Property	Type	Description
HostString	String Variable	The client server name for the connection
UserName	String Variable	The client user name for the connection

About Oracle data types

Table 7-71 describes the conversion between Oracle and Actuate data types.

Table 7-71 Default mapping between Oracle and Actuate data types

Oracle data type	Maps to
Char	Actuate String.
Date	Actuate Date. Can also map to Actuate String.
Float	Actuate Double. Can also map to Actuate Currency, Integer, Long, Single, or String.
Integer	Actuate Integer. Can also map to Actuate Double, Currency, Long, Single, or String.
Interval Day To Second	Actuate String.
Interval Year To Month	Actuate String.
Long	Actuate String.
Number	Actuate Double. Can also map to Actuate Currency, Integer, Long, Single, and String.
Rowid	Actuate String.
String	Actuate String.
Timestamp	Actuate Date. Can also map to Actuate String.
Timestamp With Local Time Zone	Actuate Date. Can also map to Actuate String.
Timestamp With Time Zone	Actuate Date. Can also map to Actuate String.
Urowid	Actuate String.

Methods for Class AcOracleConnection

Methods inherited from Class AcDBConnection

GetGeneralError, GetGeneralErrorText, GetSpecificError, GetSpecificErrorText, Prepare

Methods inherited from Class AcConnection

Connect, Disconnect, IsConnected, RaiseError

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcOrderedCollection

The abstract base class for the Actuate ordered collection classes. Figure 7-75 shows the class hierarchy of AcOrderedCollection.

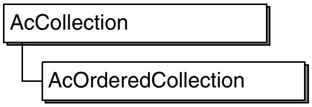


Figure 7-75 AcOrderedCollection

Description A collection contains objects of any type. An ordered collection enables you to control the order in which objects appear in a collection.

See also Class AcCollection
Class AcIterator

Methods for Class AcOrderedCollection

Methods defined in Class AcOrderedCollection

AddToHead, AddToTail, GetAt, GetHead, GetIndex, GetTail, InsertAfter, InsertAt, InsertBefore, RemoveHead, RemoveTail, SetAt

Methods inherited from Class AcCollection

Compare, Contains, Copy, FindByValue, GetCount, IsEmpty, NewIterator, Remove, RemoveAll

AcOrderedCollection::AddToHead method

Adds an item to the beginning of the collection. If you create an iterator over the collection immediately after calling AddToHead( ), the object you just added is the first object the iterator returns.

Syntax Sub AddToHead(item As Anyclass)

Parameter **item**
The object to add to the collection.

See also Class AcIterator

AcOrderedCollection::AddToTail method

Adds an item to the end of the collection. If you create an iterator over the collection immediately after calling AddToTail( ), the object you just added is the last object the iterator returns.

Syntax Sub AddToTail(item As Anyclass)

Parameter **item**
The object to add to the collection.

See also Class AcIterator

AcOrderedCollection::GetAt method

Returns a reference to the item at the specified location in the collection. If you specify an invalid index, the framework returns a run-time error.

Syntax

```
Function GetAt( index As Integer ) As AnyClass
```

Parameter **index**
The position of the object to retrieve.

Returns A reference to the object at the specified location in the collection.

See also AcOrderedCollection::GetHead method
AcOrderedCollection::GetTail method

AcOrderedCollection::GetHead method

Returns a reference to the first object in the collection. If the collection is empty, Actuate returns a run-time error. Therefore, if you do not know if the collection contains any objects, call IsEmpty( ) first.

Syntax

```
Function GetHead( ) As AnyClass
```

Returns A reference to the first object in the collection.

See also AcCollection::IsEmpty method
AcOrderedCollection::GetTail method

AcOrderedCollection::GetIndex method

Returns the position of an object in the collection.

Syntax

```
Function GetIndex( item As AnyClass ) As Integer
```

Returns The object's position expressed as an integer.

AcOrderedCollection::GetTail method

Returns the last object in the collection. If the collection is empty, the framework returns a run-time error. If you do not know whether the collection contains any objects, call IsEmpty( ) first.

Syntax

```
Function GetTail( ) As AnyClass
```

Returns A reference to the last object in the collection.

See also AcCollection::IsEmpty method

AcOrderedCollection::GetHead method

AcOrderedCollection::InsertAfter method

Inserts an object after another object in the collection. Both objects remain in the collection. To replace an object with another object, use the SetAt( ) method.

Syntax Function InsertAfter(item As AnyClass, after As AnyClass)

Parameters **item**
The object to insert.

after
The object after which to insert a new object.

See also AcOrderedCollection::SetAt method

AcOrderedCollection::InsertAt method

Inserts an object at a specific location in the collection. The object currently at that location and all objects above it move up one position in the collection. To replace an object with the object you are inserting, use the SetAt( ) method.

Syntax Sub InsertAt(index As Integer, newItem As AnyClass)

Parameters **index**
The location at which to insert the object.

newItem
The object to add.

See also AcOrderedCollection::SetAt method

AcOrderedCollection::InsertBefore method

Inserts an object before another object in the collection. Both objects remain in the collection. To replace an object with another object, use the SetAt( ) method.

Syntax Function InsertBefore(item As AnyClass, after As AnyClass)

Parameters **item**
The object to insert.

after
The object before which to insert a new object.

See also AcOrderedCollection::SetAt method

AcOrderedCollection::RemoveHead method

Removes the first item in the collection. If the collection is empty, the framework returns a run-time error. If you do not know if the collection contains any objects, call IsEmpty( ) first.

Syntax Function RemoveHead( ) As AnyClass

Returns A reference to the deleted object.

See also AcCollection::IsEmpty method
AcOrderedCollection::AddToHead method

AcOrderedCollection::RemoveTail method

Removes the last item in the collection. If the collection is empty, the framework returns a run-time error. If you do not know if the collection contains any objects, call IsEmpty( ) first.

Syntax Function RemoveTail( ) As AnyClass

Returns A reference to the deleted object.

See also AcCollection::IsEmpty method
AcOrderedCollection::AddToTail method

AcOrderedCollection::SetAt method

Sets an object at a specified position, replacing the object at that position. To insert an object into a collection without replacing the current object, see the InsertAfter( ), InsertAt( ), and InsertBefore( ) methods.

Use SetAt( ) to store an object in a particular location in the connection. If the index you specify is beyond the current collection size, SetAt( ) resizes the collection accordingly. If the index you specify is within the current collection size, SetAt( ) places the specified object in the existing location, replacing any object that might be stored there.

To store an object to the end of the collection, use AddToTail( ).

Syntax Sub SetAt(index As Integer, obj As AnyClass)

Parameters **index**
The position at which to set the object.

obj
The object to set at the specified position.

See also AcOrderedCollection::AddToTail method
AcOrderedCollection::InsertAfter method
AcOrderedCollection::InsertAt method
AcOrderedCollection::InsertBefore method

Class AcPage

The base class for all pages. Figure 7-76 shows the class hierarchy of AcPage.

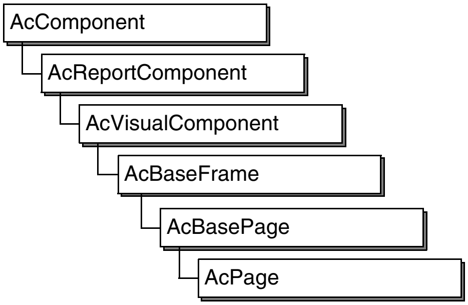


Figure 7-76 AcPage

Description The AcPage class represents pages in a report. When you instantiate a page, you set the size and orientation of the page. The Factory creates pages one at a time as needed. Pages are persistent in the report object instance (.roi) file. Pages can contain flows and other page decoration controls. Pages do not work with data rows.

Two numbers identify each page. The first number is the page index, which identifies the position of the page within the report, starting with 1. The second number is the page number, which you can display on the page in a variety of formats, including Page 2 of 25 and Page 3:42.

You can number and display pages in your report by using the functionality of the AcPageNumberControl class.

Resizing the page

Pages provide the ability to grow or shrink depending on the number of frames that appear within them. Pages can always grow as large as the flow in which they appear. To shrink a page, set the CanShrink property to True. Then, set the MinimumHeight property to specify how small the subpage can get.

A page provides a simple model for moving and resizing its contents based on where contents appear relative to the flows in that page. Consider Figure 7-77.

Flow 1 and Flow 2 are flows. Lettered items indicate various controls. Controls belong to one of three groups depending on how changes in a page size affect the controls.

The controls in the first group remain constant in size and position. A control remains constant if its top is above the flow midpoint and its bottom is above the flow bottom. In the diagram, controls X, C, D, and F remain unchanged as the page resizes.

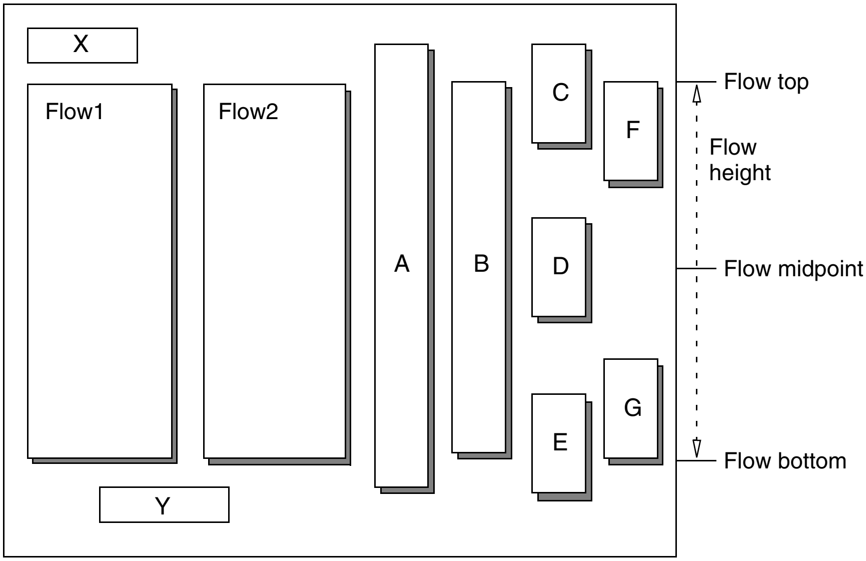


Figure 7-77 Overview of page flow features

The controls in the second group maintain the same size but their top position changes to maintain a constant distance from the bottom of the page. A control belongs to this group if its top is below the flow midpoint. In the diagram, controls E, G, and Y belong to this group.

The controls in the third group maintain a constant position but change in size by the same amount as the flow. A control belongs to this group if its top is at or above the flow top and its bottom is at or below the flow bottom. Using controls in this group allows you to create lines alongside the flows. In the preceding diagram, controls A and B belong to this group.

To make a line grow or shrink with the flow but remain somewhat shorter than the flow, set the ResizeRegion property to indicate the tolerance for resizing. If you set this property, the page adds its value to the flow top and subtracts it from the flow bottom to shrink the region that a control must span to be resized.

Class protocol

Table 7-72 describes the class protocol for AcPage.

Table 7-72 Class protocol for AcPage

Method	Task
Start( )	Instantiates and start the contents of the frame
FormatPageNumber( )	Creates the formatted page number
AddFrame( )	Adds each frame to the page

Table 7-72 Class protocol for AcPage

Method	Task
Finish( )	Finishes each of the content objects

Variables

Table 7-73 lists AcPage variables.

Table 7-73 AcPage variables

Variable	Type	Description
PageIndex	Integer	The number of the page within the report, starting at 1
PageNumber	String	The formatted page number

Properties

Table 7-74 lists AcPage properties.

Table 7-74 AcPage properties

Property	Type	Description
HorizontalOverlap	AcTwips	The amount of horizontal overlap between adjacent page fragments. Set this property to select the appropriate place to join multiple pages. The setting of this property applies only if SmartSplitHorizontally is set to False. The default value is 1 inch.
MaximumHeight	Integer	The maximum page height before a page break is forced. If CanShrink and CanExpand are both False, MaximumHeight is ignored. If MaximumHeight is smaller than the page height, the page height is used as the maximum height. The default value is 200 inches.
PrintSize	AcSize	The default page size to use for printing the page or exporting the page to PDF. If a source external to the report, such as a printer driver, specifies a different value, the value of PrintSize is ignored.

(continues)

Table 7-74 AcPage properties (continued)

Property	Type	Description
PrintSize *(continued)*	AcSize *(continued)*	If the height or width is set to 0, the Factory service substitutes this value with the initial height or width of the page during report generation. The default value is 0, 0.
SmartSplitHorizontally	Boolean	Applies only to a cross-tab report. If True, the Factory service splits a page horizontally at a boundary of an element in the report. The Factory service splits the page at the left edge of a column with a horizontal overlap of 0.5 point. The default value is True.
SmartSplitVertically	Boolean	Applies only to a cross-tab report and a dynamic text control. If True, the page splits vertically according to the following rules: ■ A cross-tab report splits at the top edge of a row with a vertical overlap of 0.5 point. ■ A dynamic text control splits between lines with no vertical overlap. ■ A page splits between top-level frames within a flow. ■ If a frame overruns the bottom of a page fragment and fits completely on the next page fragment, the page splits at the top of the frame with no overlap. ■ Splitting between frames takes priority over splitting within a frame. The default value is True.
SplitMarginBottom	AcTwips	The margin between the bottom edge of a page fragment and its contents. This setting does not apply to the last page fragment. The default value is 0.75 inch.
SplitMarginLeft	AcTwips	The margin between the left edge of a page fragment and its contents. This setting does not apply to the rightmost page fragment. The default value is 0.75 inch.

Table 7-74 AcPage properties (continued)

Property	Type	Description
SplitMarginRight	AcTwips	The margin between the right edge of a page fragment and its contents. This setting does not apply to the leftmost page fragment. The default value is 0.75 inch.
SplitMarginTop	AcTwips	The margin between the top edge of a page fragment and its contents. This setting does not apply to the first page fragment. The default value is 0.75 inch.
VerticalOverlap	AcTwips	The amount of vertical overlap between adjacent page fragments. Set this property to select the appropriate place to join multiple pages. The setting of this property applies only if SmartSplitVertically is set to False. The default value is 0.5 inch.

See also Class AcPageNumberControl

Methods for Class AcPage

Methods defined in Class AcPage

FormatPageNumber, GetVisiblePageIndex, SplitMarginBottom, SplitMarginLeft, SplitMarginRight, SplitMarginTop

Methods inherited from Class AcBasePage

BalanceFlows, GetFirstDataFrame, GetLastDataFrame

Methods inherited from Class AcBaseFrame

AddToAdjustSizeList, BindToFlow, FindContentByClass, FindContentByClassID, GetControl, GetControlValue, GetPageNumber, GetSearchValue, IsDataFrame, IsFooter, IsHeader, MakeContents, RebindToFlow, SearchAttributeName

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcPage::FormatPageNumber method

Returns the formatted page number when the report uses custom page number formatting. Override FormatPageNumber( ) to perform custom page number formatting.

Syntax Function FormatPageNumber(pageIndex As Integer) As String

Parameter **pageIndex**
The index of the current page.

Returns The formatted page number.

AcPage::GetVisiblePageIndex method

Returns the index for a visible page. A visible page is one that the current user is authorized to view.

Syntax Function GetVisiblePageIndex( ) As Integer

Returns The index for a visible page.

AcPage::SplitMarginBottom method

Implements the SplitMarginBottom property. The SplitMarginBottom property specifies the margin between the bottom edge of a page fragment and its contents.

Syntax Function SplitMarginBottom( ) As AcTwips

AcPage::SplitMarginLeft method

Implements the SplitMarginLeft property. The SplitMarginLeft property specifies the margin between the left edge of a page fragment and its contents.

Syntax Function SplitMarginLeft( ) As AcTwips

AcPage::SplitMarginRight method

Implements the SplitMarginRight property. The SplitMarginRight property specifies the margin between the right edge of a page fragment and its contents.

Syntax Function SplitMarginRight( ) As AcTwips

AcPage::SplitMarginTop method

Implements the SplitMarginTop property. The SplitMarginTop property specifies the margin between the top edge of a page fragment and its contents.

Syntax Function SplitMarginTop( ) As AcTwips

Class AcPageList

Instantiates and holds the report pages. Figure 7-78 shows the class hierarchy of AcPageList.

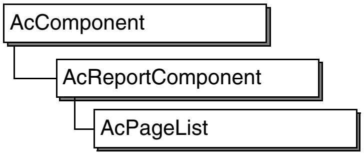


Figure 7-78 AcPageList

Description AcPageList is an abstract class whose methods apply to all types of page lists. Derived classes provide the organization of the pages within the page list.

About page structure

A report's page structure consists of a page list, pages, flows, and frames. A page list can contain multiple pages, a page can contain multiple flows, and a flow can contain multiple frames. Figure 7-79 shows the page structure of page list, page, flow, and frame.

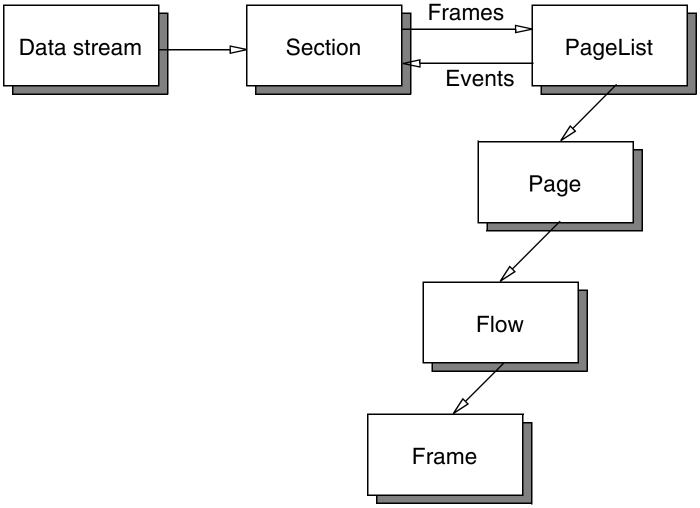


Figure 7-79 Overview of page creation

The page structure creation process is:

- A page list receives a frame from a section.
- The page list then attempts to place that frame on the current page.
- The page attempts to place the frame in the current flow:

- If there is room for the frame on the current page, the flow places the frame on the page.
- If there is no space on the page for the frame, the page list builds another page. The framework then places the frame on the new page.

Figure 7-79 also shows the roles of a data stream and section in building a report. The data stream supplies data rows to the section. The data stream does not control how the section places those data rows in a frame.

The section builds frames from the data rows that the data stream supplies and passes them to the page list. The section does not control how the page list places those frames on the page. In the simplest case, the section does not respond to events. The section only feeds frames to the page list. In a more advanced case, the section can respond to events such as a page break and send special frames such as headers and footers to the page list.

A section can contain logic that causes the page list to eject a page. For example, the section can generate a blank even-numbered page at the end of the section if one is needed for double-sided printing.

Adding frames to the page

The AddFrame( ) method adds a frame to the page list. If there is a current page and the frame fits on that page, AddFrame( ) places the frame on the page. If the frame does not fit, the page list ejects the current page, instantiates a new page, and places the frame on the new page.

The page list must also determine whether the frame requires a page break. If the frame's PageBreakAfter property is set to True, the page list ejects the page after adding the frame. If PageBreakAfter is set to False, the process of adding frames to the page continues.

When the page list does not have a page, AddFrame( ) instantiates a new page. AddFrame( ) sends a request to each active component in the structure hierarchy, starting with the frame passed to AddFrame( ). AddFrame( ) instantiates a page by calling each component's NewPage( ) method.

A frame can contain information about the page on which the frame appears. For example, if the frame's PageBreakBefore property is set to True, AddFrame( ) ejects the current page and instantiates a new page before placing the frame on the new page. When the frame does not contain information about where to place the page, the page list traverses up the content structure searching for page placement information from successively higher nodes. If the search reaches the top, the page list instantiates the default page defined in the page list subclass.

Page and structure hierarchies

The page hierarchy is connected to the structure hierarchy at both the top and bottom ends. Figure 7-80 illustrates how the page hierarchy and content hierarchy relate to each other. The hierarchies are connected at the top by a report object and at the bottom by frames.

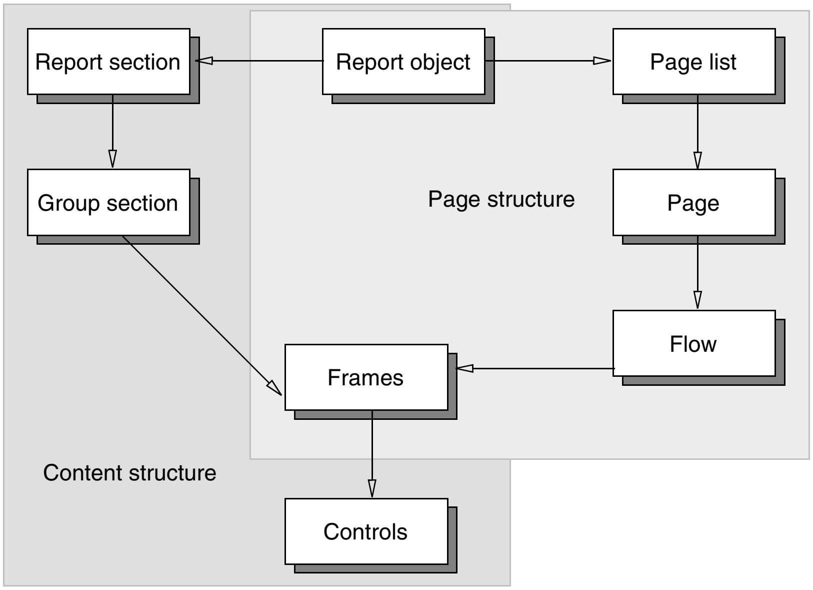


Figure 7-80 Relationship of page hierarchy and content hierarchy

About the current page

The page list always has references to the first and last pages. Typically, the page list also has a reference to the current page. The current page is the page on which the Factory is currently placing and filling flows. At some times, the page list does not have a current page. For example, when a page is ejected and before another page builds, the page list does not have a current page. The page list delays building another page as long as possible. Specifically, if a frame or section has its PageBreakAfter property set to True, the page list does not have a current page until it receives another frame. Figure 7-81 illustrates the state of the page list after page 4 is ejected. The page list that no longer refers to a current page appears broken.

Subclassing AcPageList

AcPageList is an abstract class for all types of page lists. The derived classes define the organization of the pages in the page list. The Actuate framework provides the following derived classes from AcPageList:

- AcSimplePageList builds a page list of body pages all of the same style.
- AcLeftRightPageList builds a page list of alternating left and right pages.

- AcTitleBodyPageList builds a page list of a title page followed by a simple page list.

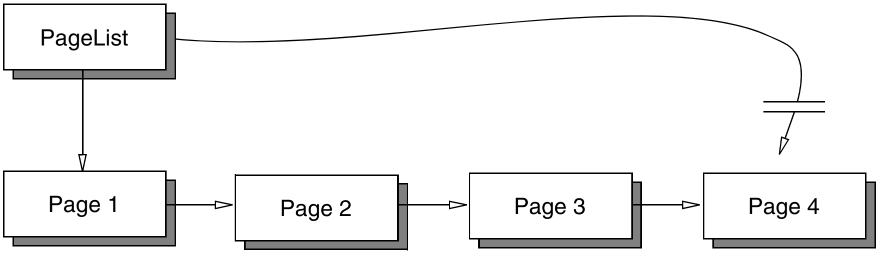


Figure 7-81 The page list

Variable

Table 7-75 describes the AcPageList variable.

Table 7-75 AcPageList variable

Variable	Type	Description
Pages	AcList	The list of all pages in the page list

Property

Table 7-76 describes the AcPageList property.

Table 7-76 AcPageList property

Property	Type	Description
SplitOversizePages WhenPrinting	Boolean	Specifies whether all pages in the page list split to print to an output format that is smaller than the page. If True, the page splits. The default value is True.

Methods for Class AcPageList

Methods defined in Class AcPageList

AddFrame, EjectPage, GetCurrentFlow, GetCurrentPage, GetCurrentPageACL, GetEstimatedPageCount, GetFirstPage, GetLastPage, GetPageCount, HasPageSecurity, NeedCheckpoint, NeedHeight, NewPage, UseAcceleratedCheckpoints

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcPageList::AddFrame method

Adds a frame to the page list and places the frame in a flow. This method creates new pages as needed to accommodate the frame. This method also splits frames over multiple flows if necessary. For additional information about AddFrame( ), see "Adding frames to the page," earlier in this section.

Syntax Sub AddFrame(frame As AcFrame)

Parameter **frame**
The frame to add to the page list.

AcPageList::EjectPage method

Finishes the currently active page. If no page is active, EjectPage( ) does nothing.

Syntax Sub EjectPage( )

AcPageList::GetCurrentFlow method

Returns the active flow on the current page. The active flow is the flow in which the Factory is currently placing components.

Syntax Function GetCurrentFlow( ) As AcFlow

Returns The currently active flow, if any, on the current page, if any.
Nothing if no flow is active.

AcPageList::GetCurrentPage method

Returns the current page in the page list. The current page is the page on which the Factory is currently placing and filling flows.

Syntax Function GetCurrentPage( ) As AcPage

Returns The current page.

AcPageList::GetCurrentPageACL method

Call GetCurrentPageACL( ) to retrieve the access control list (ACL) for the current page. Developers define ACLs to restrict access to pages. For information about ACLs, see "Customizing page-level security." To help debug reports that use page-level security, define a text control that has its ValueExp property set to GetPageList( ).

Syntax Function GetCurrentPageACL( ) As String

Returns A comma-separated list of security IDs that comprise the ACL.
An empty string if page security is not defined for this page.

AcPageList::GetEstimatedPageCount method

Provides an estimate of the number of pages needed for this report. GetEstimatedPageCount( ) supports optimizing the layout of data in the report. Override this method to provide an estimate of the number of pages that the report will contain. The Factory pre-allocates parts of the report. Pre-allocating parts of the report reduces the number of reads required to view the report.

The estimate is accurate to a power of 50. For example, provide a value of 1 for reports up to 50 pages long. Provide a value of 51 for reports up to 2500 pages long. Try to be as accurate as possible when you provide the estimate.

Syntax Function GetEstimatedPageCount( ) As Integer

Example

```
Function GetEstimatedPageCount( )
   GetEstimatedPageCount = 100
End Function
```

AcPageList::GetFirstPage method

The page list holds all the pages in the report. Returns the first page in the page list.

Syntax Function GetFirstPage( ) As AcPage

Returns The first page in the page list.

AcPageList::GetLastPage method

Returns the last page in the page list.

Syntax Function GetLastPage( ) As AcPage

Returns If you call GetLastPage( ) when viewing a complete report, GetLastPage( ) returns the last page of the page list.

If you call GetLastPage( ) during report generation, GetLastPage( ) returns the last page that currently exists. For example, if you call GetLastPage( ) when the report is partially built, it returns the current page.

AcPageList::GetPageCount method

Returns the number of pages in the page list. If called before the page list is complete, GetPageCount( ) returns the number of pages processed so far.

Syntax Function GetPageCount( ) As Integer

Returns The number of pages in the page list.

AcPageList::HasPageSecurity method

Indicates whether a page in the page list uses page-level security.

Syntax Function HasPageSecurity( ) As Boolean

Returns True if the page uses page-level security.
False if the page does not use page-level security.

AcPageList::NeedCheckpoint method

Override this method to control how frequently to flush persistent objects to the report object instance (.roi) file. Flushing too frequently can cause significant performance degradation and can increase the size of the ROI.

Syntax Function NeedCheckpoint(pageCount As Integer) As Boolean

Parameter **pageCount**
The page number to act on, usually the page whose generation has just been completed.

Returns True if a checkpoint is required. The Factory then flushes the page to persistent storage and the viewer can render the page.
False if no checkpoint is required.

AcPageList::NeedHeight method

Ensures that a specified amount of vertical space is available in the current flow, and if not, starts a new flow. NeedHeight( ) requests a certain amount of vertical space in the current flow. NeedHeight( ) requests a new flow if the requested vertical space cannot fit in the current flow. Space is measured in twips.

Syntax Sub NeedHeight (Height As Integer)

Parameter **Height**
The amount of required vertical space in twips.

AcPageList::NewPage method

The page list calls NewPage( ) to instantiate each new page. If you use one of the predefined page classes, the framework creates the page for you based on the components you add to the page list in Report Structure. If you create a custom page list class, you must override this method.

Syntax Function NewPage( ) As AcPage

Example The following example shows how to create different page designs for two reports that run in sequence: a customer list and an order list.

In the following code example, the custom variable TestIndex has a value of 1 when the first of the two sequential reports runs. When the second report runs, TestIndex has a value of 2.

```
Function NewContent( index As Integer ) As AcReportComponent
   Set NewContent = Super::NewContent( index )
   TestIndex = index
End Function
```

The following code example passes the value of TestIndex to the NewPage( ) method, so that the NewPage( ) method knows whether to instantiate either the CustomerPage or the OrderPage component.

The call to Super::NewPage( ) is commented out:

```
Function NewPage( ) As AcPage
' Set NewPage = Super::NewPage( )
   Select Case TestIndex
      Case 1
         Set NewPage = New Persistent CustomerPage
      Case 2
         Set NewPage =  New Persistent OrderPage
   End Select
End Function
```

Returns The new page instance.

AcPageList::UseAcceleratedCheckpoints method

Creates additional page checkpoints in the report. Override UseAcceleratedCheckpoints( ) to return True if you want the Factory to increase the number of page checkpoints written to the report. Additional checkpoints improve report viewing performance.

Syntax Function UseAcceleratedCheckpoints( ) As Boolean

Returns True if more checkpoints are to be created.
False if no additional checkpoints are to be created.

Class AcPageNumberControl

Calculates and displays page numbers. Figure 7-82 shows the class hierarchy of AcPageNumberControl.

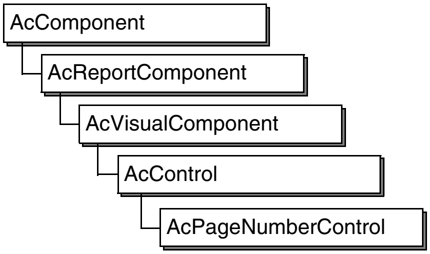


Figure 7-82 AcPageNumberControl

Description Use the page number control to calculate and display the current page number or the total number of pages in the report. You can use the page number control to display relative page numbers in the form, 1 of n. The page number control can number pages for secure reports considering the visibility of pages to the user. Pages in secure reports are visible to the user only if the user is granted access to the page. For information about granting access to pages, see "Using page-level security," later in this chapter.

About page number types

The PageNumberType property of AcPageNumberControl determines how to calculate and display the page number value. The available page number types are:

- ActualPageCount
- ActualPageNofM
- ActualPageNumber
- FormattedPageNumber
- VisiblePageCount
- VisiblePageNofM
- VisiblePageNumber

ActualPageCount and ActualPageNumber show the total and current page numbers, respectively, without considering page security. ActualPageNofM shows the actual page number relative to the actual page count in the report. VisiblePageCount and VisiblePageNumber show the total and current page numbers, respectively, considering page security. VisiblePageNofM shows the visible page number relative to the visible page count in the report. For nonsecure reports, the values for visible and actual page number types are the same.

The FormattedPageNumber displays page numbers using the format specified in the PageNumberFormat property on the component, AcPage. Page numbers displayed using this type do not consider page security.

Selecting the page numbering type

Visible page numbers and counts can result in different page numbers on the same report for users with different access to pages. The total number of pages in a secure report can be different for users with different access to pages. Use actual page numbers and counts when users with different access to pages need to refer to the report by page number.

Property

Table 7-77 describes the AcPageNumberControl property.

Table 7-77 AcPageNumberControl property

Property	Type	Description
PageNumber Type	AcPage Number Style	PageNumberType can be set to the following values: ■ ActualPageCount. The total number of pages in the report, including those not visible to the user. ■ ActualPageNofM. The current page number, N, relative to the total page count, M, displayed in the following form: `Page N of M` The page number and count include pages that are not visible to the user. ■ ActualPageNumber. The current page number considering all pages, including those not visible to the user. ■ FormattedPageNumber. Page number is presented using the format string specified in the PageNumberFormat property. The value presented here does not consider page security. ■ VisiblePageCount. The total number of pages in the report that the user can see considering page security. ■ VisiblePageNofM. The current page number, N, relative to the total page count, M, displayed in the form: Page N of M. The page number and count considers page security. ■ VisiblePageNumber. The current page number considering only the pages that the user can see considering page security. The default value is VisiblePageNumber.

See also Class AcPage
Class AcPageList

Methods for Class AcPageNumberControl

Methods defined in Class AcPageNumberControl

GetActualPageCount, GetActualPageNumber, GetFormattedPageNumber, GetVisiblePageCount, GetVisiblePageNumber, PageN, PageNOfM, PageNumberType

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcPageNumberControl::GetActualPageCount method

Returns the total page count for the report without considering page security. If the report is secure, the total page count considers pages that are not visible to the user because the user has not been granted access to the page.

Syntax Function GetActualPageCount( ) As Integer

Returns The page number.

See also AcPageNumberControl::GetVisiblePageCount method

AcPageNumberControl::GetActualPageNumber method

Returns the current page number for the report without considering page security. If the report is secure, the page number considers pages that are not visible to the user because the user has not been granted access to the page.

Syntax Function GetActualPageNumber( ) As Integer

Returns The page number.

See also AcPageNumberControl::GetVisiblePageNumber method

AcPageNumberControl::GetFormattedPageNumber method

Returns the current page number without considering page security, using the format specified in the PageNumberFormat property for the page. Call GetFormattedPageNumber( ) to retrieve the page number as a string formatted according to the information in the PageNumberFormat property for the page. The string returned is the same string as the one returned by PageNo$( ).

Syntax Function GetFormattedPageNumber( ) As String

Returns The current page number.

AcPageNumberControl::GetVisiblePageCount method

Returns the total page count for the report considering page security. If the report is secure, the total page count excludes any pages that are not visible to the user because the user has not been granted access to the page or pages.

Syntax Function GetVisiblePageCount( ) As Integer

Returns The page count of visible pages.

See also AcPageNumberControl::GetActualPageCount method

AcPageNumberControl::GetVisiblePageNumber method

Returns the current page number for the report considering page security. The page number excludes any pages that are not visible because the user cannot access the pages.

Syntax Function GetVisiblePageNumber( ) As Integer

Returns The page number of a visible page.

See also AcPageNumberControl::GetActualPageNumber method

AcPageNumberControl::PageN method

Formats controls that have the page number types ActualPageN or VisiblePageN.

Syntax Function PageN(pageNo As Integer) As String

Returns The value of the page number format.

Parameter **pageNo**
The page number.

AcPageNumberControl::PageNOfM method

Formats page number controls that have the type ActualPageNofM or VisiblePageNofM.

Syntax Function PageNOfM(pageNo As Integer, pageCount As Integer) As String

Returns The value of the page number format.

Parameters **pageNo**
The page number.

pageCount
The page count.

AcPageNumberControl::PageNumberType method

Returns the value of the PageNumberType property for a page number control. The enumerated type AcPageNumberStyle defines the available values for this property.

Syntax Function PageNumberType( ) As AcPageNumberStyle

Returns The value of the PageNumberType property.

Class AcParallelSection

A class that fills two or more flows on the page. Figure 7-83 shows the class hierarchy of AcParallelSection.

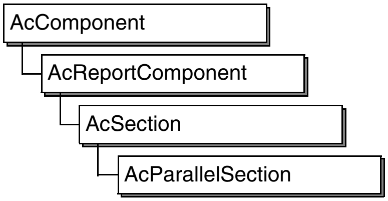


Figure 7-83 AcParallelSection

Description A parallel section contains two or more report sections, group sections, or sequential sections that appear in different flows on the same page. A parallel section can present two different reports side by side.

On a page that has multiple flows, AcParallelSection fills each flow using a different data stream. When each flow on a page is full, a new page builds and the process of filling each flow continues until all data streams are processed.

When designing a parallel section, you must establish the relationship between reports in the parallel section and flows on the page. You do so by setting the report section's FlowName property to the name of the flow in which the report should appear. For example, if you create a page with two flows, Flow1 and Flow2, set the FlowName property of the first report to Flow1 and the FlowName property of the second report to Flow2.

The parallel section class is complex. It is recommended not to override its methods.

Setting up pages for a parallel section

The parallel section must have access to the page that contains the necessary flows. You can provide this page in one of three ways:

- If the only item in your report is the parallel section, you can use a simple page list. Ensure that the page in that page list has the correct number of flows with the correct tags.
- You can override NewPage( ) to instantiate the page you want to use within the parallel section. This page takes precedence over any page the page list provides. If you override NewPage( ), be sure to set the PageBreakBefore and PageBreakAfter properties to True so that the parallel section starts on the correct page and no other components use the special page.
- You can provide a subpage that fits inside the flow on your standard page.

Building the nested reports

The nested reports are built in the parallel section's Build( ) method. The Factory performs the following tasks in sequence to build the section:

- Instantiate and start all of the nested reports.
- Locate the first page.
- For each report that still has rows to produce, locate the corresponding flow using the report and flow's Tag property.
- Run the report until the flows fill.
- Eject the page.
- If any reports have more rows to produce, loop back to locate the next page and repeat the process.

Property

Table 7-78 describes the AcParallelSection property.

Table 7-78 AcParallelSection property

Property	Type	Description
Reports	AcReport Section	The list of nested reports to appear in the parallel report

Methods for AcParallelSection

Methods defined in Class AcParallelSection

AddReport

Methods inherited from Class AcSection

CommittedToFlow, DeletePageFrame, FinishConnection, FinishFlow, FinishPage, GetComponentACL, GetCurrentRow, GetSearchValue, NewPage, ObtainConnection, PageBreakAfter, PageBreakBefore, SetSearchValue, SetSecurity, StartFlow, StartPage, StopAfterCurrentFrame, StopAfterCurrentRow, StopNow, TocAddComponent, TocAddContents

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcParallelSection::AddReport method

Adds a subreport to the Reports slot of a parallel section.

Syntax Sub AddReport(report As AcReportSection)

Class AcQuerySource

Processes a SQL query that you build using Query Editor or Textual Query Editor. Figure 7-84 shows the class hierarchy of AcQuerySource.

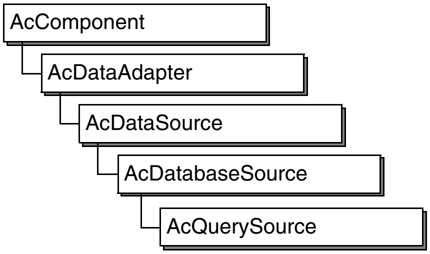


Figure 7-84 AcQuerySource

Description AcQuerySource is an abstract base class that defines the methods for executing a SQL SELECT statement including:

- Processing ad hoc parameters
- Handling dynamic ordering established by group sections
- Preparing the SQL statement and opening a cursor
- Binding static parameters
- Binding the data row to the database cursor

Figure 7-85 illustrates the operation of an AcQuerySource object.

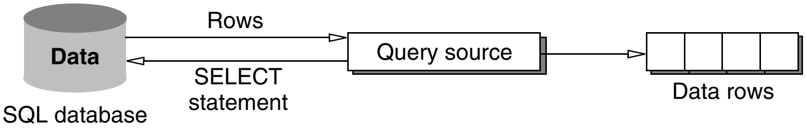


Figure 7-85 A query data source

Typically, you create a query data source using Query Editor or Textual Query Editor. You also can create the query data source programmatically. If you use programming, you must override ObtainSelectStatement( ) to return the complete statement. You also must override BindStaticParameters( ) to bind static parameters and BindDataRow( ) to bind the data row to the cursor.

See also Class AcDataAdapter
Class AcDatabaseSource
Class AcDataRow
Class AcDataSource
Class AcDBConnection
Class AcSqlQuerySource
Class AcTextQuerySource

Methods for Class AcQuerySource

Methods defined in Class AcQuerySource

GetStatementText, ObtainSelectStatement, SetupAdHocParameters

Methods inherited from Class AcDatabaseSource

BindDataRow, BindStaticParameters, GetCursor, GetDBConnection, GetPreparedStatement, OpenCursor, SetStatementProperty

Methods inherited from Class AcDataSource

HasFetchedLast

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcQuerySource::GetStatementText method

Returns the text of the SELECT statement for the SQL query source. You can call this method only after calling Start( ).

Syntax Function GetStatementText( ) As String

Returns The current SELECT statement as a string.

AcQuerySource::ObtainSelectStatement method

Returns the SELECT statement of the query source. Override ObtainSelectStatement( ) to create custom SQL SELECT statements.

Syntax Function ObtainSelectStatement( ) As String

Returns A SQL SELECT statement as a string.

AcQuerySource::SetupAdHocParameters method

Adds ad hoc parameters to the query. The AcQuerySource class calls SetupAdHocParameters( ) to enable a derived class to set the value of its ad hoc parameters. The framework typically generates this method for you. If you programmatically create a class that provides ad hoc parameters in the Requester, you must override this method. Your override calls SetAdHocParameters( ) to set the value of each parameter.

Syntax Sub SetupAdHocParameters( )

Class AcRectangleControl

Displays a rectangle in a report. Figure 7-86 shows the class hierarchy of AcRectangleControl.

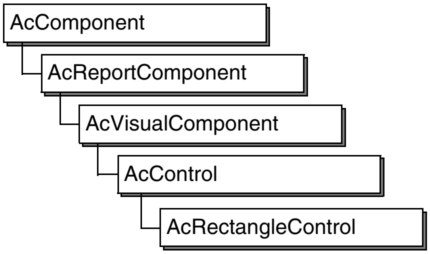


Figure 7-86 AcRectangleControl

Description A rectangle is a content control that is fully defined at design time. You can specify its color, size, geometry, and line style.

Variables

Table 7-79 lists AcRectangleControl variables.

Table 7-79 AcRectangleControl variables

Variable	Type	Description
FillColor	AcColor	The color with which to fill the shape.
ForcePage HeightTo Fit	Boolean	Determines whether to keep the text of a dynamic text control on a single page. Overrides the CanIncreaseHeight property.
ForcePage WidthToFit	Boolean	Determines whether to keep the text of a dynamic text control within specified margins. Overrides the CanIncreaseWidth property.
LineStyle	AcLineStyle	The style of line to draw around the shape.

Properties

Table 7-80 lists AcRectangleControl properties.

Table 7-80 AcRectangleControl properties

Property	Type	Description
FillColor	AcColor	The color with which to fill the shape.

(continues)

Table 7-80 AcRectangleControl properties (continued)

Property	Type	Description
Horizontal Size	AcHorizontal Size	Determines how the horizontal size of the control changes dynamically.
LineStyle	AcLineStyle	The style of line to draw around the shape.
IsFrame Decoration	Boolean	If True, the control can split across multiple pages when necessary.
Vertical Size	AcVerticalSize	Determines how the vertical size of the control changes dynamically.

Methods for Class AcRectangleControl

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcReport

The root object that contains all components in a report. Figure 7-87 shows the class hierarchy of AcReport.

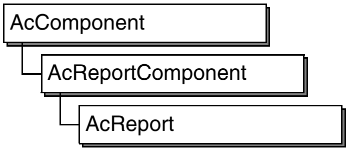


Figure 7-87 AcReport

Description AcReport is the root object that contains all other components in a report. Like the main method in a program, the AcReport object is the entry point to the report. In Report Structure, the report object appears at the top, as shown in Figure 7-88.

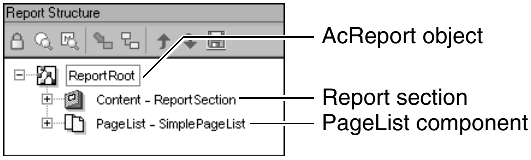


Figure 7-88 A report object

The following components are contained immediately within the report object:

- A section component, the topmost container of the content elements. The section can be a report section, conditional section, sequential section, parallel section, or group section.
- A PageList component, the topmost container of the display components.

About content and page creation

The report section starts the processes for building, running, and displaying the report. To build the report, the report object:

- Starts the PageList creation process
- Starts the content creation process

Because the report object controls the report-building process, exercise caution when overriding AcReport methods. Use Start( ) to add startup code that, for example, initializes global variables. The Factory calls Start( ) before it begins generating the report.

For more information about content or page creation, see Chapter 5, “Understanding report generation.”

Writing cleanup code

Use Finish( ) to add cleanup code or additional processing. For example, use Finish( ) to close files, send completion notices, or write statistics to a log file. The Factory calls Finish( ) after it generates the report but before closing the report file. Call Super::Finish( ) after your custom code to ensure that the Factory executes the original code.

Subclassing AcReport

e.Report Designer Professional generates a subclass of AcReport for every new report. You can subclass this subclass to make a copy of the original report.

Assigning and customizing a report name

The framework assigns a name to a report based on the value of the Output File Name parameter in Requester or a value you specify in the SuggestRoiName( ) method of AcReport.

When a report runs on iServer, Actuate software stores the report object instance (.roi) file in the Encyclopedia volume folder that contains the corresponding report object executable (.rox) file. If the report runs on a local machine, Actuate software uses the file protocol and stores the ROI in the current working directory. The ROI name is the value of the Output File Name parameter in Requester.

If you change the location of the ROI and the report runs on a local machine, the new directory must exist in the file system when the report runs. If the report runs on iServer, Actuate software creates the directory if it does not exist.

You can specify the ROI name using an absolute path in SuggestRoiName( ). When you set a path, the Factory uses the name you set in SuggestRoiName( ) for the ROI. If you use a relative path in SuggestRoiName( ), the Factory adds path information from SuggestRoiName( ) to any path information from the Output File Name parameter. For example, if the Output File Name parameter contains:

```
file:C:\Forecast\East\Quarterly.roi
```

and SuggestRoiName( ) returns:

```
Q1\Forecast.roi
```

The generated ROI name is:

```
file:C:\Forecast\East\Q1\Forecast.roi
```

Setting autoarchive rules for an AcReport object

An Encyclopedia volume administrator sets autoarchive rules for the files stored in the Encyclopedia volume. The iServer uses the autoarchive rules to determine when to delete files, how many versions of a report to keep, whether to archive the file before it is deleted, and whether to delete any dependent files at the same time. Autoarchive rules can be set for a file, folder, or the entire Encyclopedia volume. The framework searches the containment hierarchy for an autoarchive rule to use if no rule is specified for the file.

A user can enter parameter values at the Requester prompt to set autoarchive rules for the report object. For example, the user can specify that a report be deleted 10 days after it is created.

You can customize your report design to set autoarchive rules for the ROI files. The framework makes the autoarchive rules set by the user available as public variables on the AcReport component. You can change user-specified autoarchive rules in the following two ways:

- Override the Start( ) or Finish( ) method for the report and modify the autoarchive public variables.
- Override the SetROIAgingProperties( ) method.

You modify the public variables when the autoarchive rules are the same for all report objects generated by the report. If your report uses report bursting to produce multiple ROI files and you must specify different autoarchive rules for the individual files, override the SetROIAgingProperties( ) method to set the autoarchive rules. This method is called for each ROI file.

For more information about setting autoarchive rules, see *Managing an Encyclopedia Volume.* For information about using and creating an archive driver, see *Using BIRT iServer Integration Technology.*

Examples The following example shows how to change the autoarchive rules for a report. Overriding the Finish( ) method modifies the rule to automatically delete the ROI four hours after it is created. This code converts the amount of time from hours to minutes and sets the public variable ExpirationAge to the result.

```
Sub Finish
   ' Force the file to expire 4 hours after creation
   ExpirationAge = 4 * 60
End Finish
```

The following example shows how to ensure that the ROI is not removed as part of the aging and archiving process. The code sets the Aging_Options variable to the constant Age_NoExpiration.

```
Sub Finish
   ' Never delete the file during aging and archiving
   AgingOptions = Age_NoExpiration
End Finish
```

Variables

Table 7-81 lists AcReport variables.

Table 7-81 AcReport variables

Variable	Type	Description
Aging_Options	Integer	The autoarchive rules for this file. Contains a value corresponding to one of the following public variables: ■ Age_ArchiveBeforeDelete. File must be archived before it is deleted. If no archive driver is installed, this option is ignored. ■ Age_DeleteDependencies. When this file is deleted, delete any files that depend on it. ■ Age_NoExpiration. Do not delete this file. ■ Age_NoOptions. No autoarchive rules have been set for this file. The file inherits its autoarchive rules. Actuate software searches the Encyclopedia volume folder hierarchy to find autoarchive rules for the file.
BundleRox	Boolean	A parameter variable that indicates whether to bundle the report object executable (.rox) file with the report object instance (.roi) file.
DataFont	AcFont	Provided for backwards compatibility.
ExpirationAge	Integer	The number of minutes after creation that the file should be deleted. Null indicates that the file should not be deleted based on the elapsed time after creation.
ExpirationDate	Date	The date and time after which the file should be deleted. Null indicates that the file should not be deleted based on a specific date and time.
GlobalDHTML Code	String	Global custom browser code to append to every DHTML page the DHTML converter converts. The PDF converter ignores GlobalDHTMLCode.
Headline	String	A parameter variable. At report generation time, Headline appears in Requester for reports that use parameters.
Keywords	String	Defines the Keywords metadata for a rendered report.
LabelFont	AcFont	Provided for backwards compatibility.
Language	String	The default language of the report.
Layout Orientation	AcLayout Orientation	The layout orientation of the report.

(continues)

Table 7-81 AcReport variables (continued)

Variable	Type	Description
Locale	String	The locale of the report.
MaxVersCount	Integer	The maximum number of versions of the report to keep. Null indicates that no limit exists.
PageDecoration Font	AcFont	Provided for backwards compatibility.
PageHeight	AcTwips	Provided for backwards compatibility.
Pages	AcPageList	The object that holds the list of pages.
ReportType	AcReportType	Provided for backwards compatibility.
ROIName	String	The name of the report object instance (.roi) file.
Root	AcReport Component	A reference to the first element in the report. From this element, the application can traverse down to all other frame records in the report.
Summary	String	Defines the Subject metadata for a rendered report.
Title	String	Defines the Title metadata for a rendered report.
TitleFont	AcFont	Provided for backwards compatibility.
VersionName	String	The name to use for a version of the current report.
VersionRoi	Boolean	The version number to use for a version of the report.

Properties

Table 7-82 lists AcReport properties.

Table 7-82 AcReport properties

Property	Type	Description
Content	AcReport Component	The topmost report section for a report.
DataFont	AcFont	Provided for backwards compatibility.
GlobalDHTML Code	String	Global custom browser code to append to every DHTML page the DHTML converter converts. The PDF converter ignores GlobalDHTMLCode.
Keywords	String	Defines the Keywords metadata for a rendered report.
LabelFont	AcFont	Provided for backwards compatibility.

Table 7-82 AcReport properties (continued)

Property	Type	Description
Layout Orientation	AcLayout Orientation	The layout orientation of the report, regardless of the operating system or locale. Available settings are: ■ LeftToRight ■ RightToLeft The default value is LeftToRight.
Locale	String	The locale to use for the report. If an empty string, the current run-time locale is used. The default value is an empty string.
PageDecoration Font	AcFont	Provided for backwards compatibility.
PageHeight	AcTwips	Provided for backwards compatibility.
PageList	AcPageList	The PageList style to use when creating pages for a report.
PrintSize	AcSize	Provided for backwards compatibility.
RenderProfileId	String	Specifies which Render profile is to be used when report content is rendered to an output document.
Report Encoding	String	The encoding to use for report generation, viewing, and printing. The default value is the encoding of the current run-time locale.
ReportType	AcReportType	Provided for backwards compatibility.
SortParamsBy Alias	Boolean	Defines how the Requester orders the parameters. True if Requester sorts parameters by alias name. False if Requester sorts parameters by parameter name. The default value is False.
Summary	String	Defines the Subject metadata for a rendered report.
Title	String	Defines the Title metadata for a rendered report. Appears as the Title document property in PDF documents created using the PDF Writer. This value is also displayed as the window title when you view a PDF document in the Adobe Acrobat Reader.
TitleFont	AcFont	Provided for backwards compatibility.
XMLCharSet	String	The encoding declaration to be inserted in the XML prolog. If XMLCharSet is not specified, Actuate does not include an encoding declaration in the XML prolog.
XMLDocType	String	The declaration to appear after the XML DOCTYPE keyword.

(continues)

Table 7-82 AcReport properties (continued)

Property	Type	Description
XMLFile Description	String	The description of the XML file to build. The default value is XML Files.
XMLFile Extension	String	The file extension of the XML file to build. The default value is xml.
XMLIndent	Integer	The number of spaces to indent each level in the XML file. Set the value to 0 to improve the performance of XML generation and reduce the XML file size. The default value is 4. Use the default value when you view and debug the XML report.
XMLMimeType	String	The MIME type for the XML file. The default value is text/xml.

See also Class AcPageList

Methods for Class AcReport

Methods defined in Class AcReport

GenerateXMLDataFile, GetContent, GetCustomFormat, GetFactoryLocale, GetGlobalDHTMLCode, GetLanguage, GetLayoutOrientation, GetPrintLocale, GetReport, GetUserACL, GetViewLocale, HasPageSecurity, NewContent, NewPageList, OnFinishPrint, OnStartPrint, RoiIsTemporary, SetBurstReportPrivileges, SetGlobalDHTMLCode, SetLayoutOrientation, SetROIAgingProperties, SuggfestRoiName, TocAddComponent, XMLDataProlog

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcReport::AfterFinishingReport method

Performs work after the report has finished generating and the ROI file has been closed. When AfterFinishingReport is executed, the value of AcReport's ROIName member variable will include the file version number. Use IDAPI to manipulate the properties of the ROI file from within this method.

Syntax Sub AfterFinishingReport()

AcReport::BeforeStartingReport method

Performs work before the ROI file is opened and the report has begun generating.

Syntax Sub BeforeStartingReport()

AcReport::GenerateXMLDataFile method

Builds an XML file from the report, using the XML Data Group.

Syntax Function GenerateXMLDataFile(fileName As String) As Boolean

Parameter **fileName**
The name of the XML file to be generated.

Returns True if the XML file builds successfully.
False if the XML file cannot be built.

AcReport::GetContent method

Returns the component in the Content slot of the root report component.

Syntax Function GetContent( ) As AcReportComponent

AcReport::GetCustomFormat method

You use the Actuate Basic Excel classes to export report data to an Excel spreadsheet. Use GetCustomFormat( ) to retrieve the generated Excel file.

Syntax Sub GetCustomFormat( )

AcReport::GetFactoryLocale method

Specifies the locale to use for report generation. The value of the Locale property is used for report generation. Override GetFactoryLocale( ) to set a different locale for report generation.

The value this method returns replaces the value of the Locale property of the generated report.

Syntax Function GetFactoryLocale(defaultLocale As String) As String

Parameter **defaultLocale**
The locale to use for report generation.

Returns The locale name.

AcReport::GetGlobalDHTMLCode method

Returns the custom code from a browser scripting control and makes it available to every DHTML page the DHTML converter generates.

Syntax Function GetGlobalDHTMLCode( ) As String

AcReport::GetLanguage method

Returns the language of the report as a string.

Syntax Function GetLanguage( ) As String

Returns The language of the locale.

AcReport::GetLayoutOrientation method

Returns the report orientation. You can call GetLayoutOrientation( ) at report generation time but not for report viewing.

Syntax Function GetLayoutOrientation( ) As AcLayoutOrientation

Returns One of the following orientations for the report:

- LeftToRight
- RightToLeft

AcReport::GetPrintLocale method

Specifies the locale to use for printing the report on iServer. Override GetPrintLocale( ) to set a different locale for printing. The value this method returns is only for report printing on iServer. The return value does not replace the value of the Locale property of the generated report.

Syntax Function GetPrintLocale(defaultLocale As String) As String

Parameter **defaultLocale**
The locale to use for report printing.

Returns The locale name.

AcReport::GetReport method

Returns a reference to the root report component.

Syntax Function GetReport( ) As AcReport

AcReport::GetUserACL method

Returns the access control list (ACL) for the current user.

The view process or print process calls GetUserACL( ) to retrieve the list of security identifiers for the current user. Actuate software builds an ACL for the user that consists of his user ID, the roles associated with his group, and optionally, any security IDs the Report Server Security Extension (RSSE) supplies.

Override GetUserACL( ) if you want to modify or replace the ACL that Actuate software builds.

The AcReportComponent, AcReport, and AcSection components implement page-level security. For more information about page-level security, see AcReportComponent and AcSection.

Syntax Function GetUserACL(acl As String) As String

Parameter **acl**
The list of security IDs for the current user.

Returns The list of security identifiers separated by commas.
Nothing if no security identifiers are defined.

Example You can add virtual security IDs as well as valid roles or user IDs. A virtual security ID is a combination of valid roles in the Encyclopedia volume. Virtual security IDs help you create additional security IDs without having to update the Encyclopedia volume with additional roles. For example, you can restrict access to all sales managers that sell four-wheel drive vehicles. In this case, you create a virtual security ID that is represented by the sales manager role and a four-wheel drive vehicle product category role, as shown in the following example. This code combines the actual security IDs Manager and Product Category to create the virtual security ID Manager_Product Category.

```
Function GetUserACL( acl As String ) As String
    GetUserACL = Super::GetUserACL( acl )

   Dim tail   As String
   Dim mgr    As String
   Dim prod   As String
   Dim posn   As Integer
   Dim sid    As String
   ' Loop to get each SID and check if we want it.
   tail = acl
```

```
    Do While tail <> ""
       posn = InStr( tail, "," )
       If posn = 0 Then
          sid = Trim$( tail )
          tail = ""
       Else
          sid = Trim$( Left$( tail, posn - 1 ) )
          tail = Trim$( Mid$( tail, posn + 1 ) )
       End If
       ' Check if it is a manager SID or a product
       ' category SID.
       If InStr( sid, "Manager" ) > 0 Then
          mgr = sid
       ElseIf InStr( sid, "Product Category" ) > 0 Then
          prod = sid
       End If
    Loop
    ' Build the special ACL and add it to the list.
    If mgr <> "" And prod <> "" Then
       acl = acl & ", " & mgr & " " & prod
    End If
    GetUserACL = acl
 End Function
```

See also Class AcReportComponent
Class AcSection

AcReport::GetViewLocale method

Specifies the locale to use for report viewing. The value of the Locale property is for report viewing. To set a different locale for report viewing, override GetViewLocale( ).

The value this method returns is used only for report viewing. The return value does not replace the value of the Locale property of the generated report.

Syntax Function GetViewLocale(defaultLocale As String) As String

Parameter **defaultLocale**
The locale to use for report viewing.

Returns The view locale.

AcReport::HasPageSecurity method

Returns True if the report uses page-level security. A report uses page-level security if the access control list associated with any of its pages is not empty.

Page-level security is a technique for controlling user access to a report on a page-by-page basis. In a report that uses page-level security, a report user can view, search, and print only pages to which he has access.

Syntax Function HasPageSecurity( ) As Boolean

Returns True if the report uses page security.
False if the report does not use page security.

AcReport::NewContent method

A generated method that creates the top-level section. Typically, you do not need to override NewContent( ).

Syntax Function NewContent( ) As AcReportComponent

Returns A reference to the AcReportComponent object it creates.

AcReport::NewPageList method

A generated method that creates the page list for the report. Using the page list specified in e.Report Designer Professional, NewPageList( ) creates an instance of a subclass of AcPageList.

Typically, you do not override NewPageList( ). Instead, you use e.Report Designer Professional to specify the PageList class to use for the report. Alternatively, you can override NewPageList( ) to create the page list with a different page style. If you override this method, replace it. Do not call the superclass method.

Syntax Function NewPageList( ) As AcPageList

Returns A reference to the AcPageList object it creates.

See also Class AcPageList

AcReport::OnFinishPrint method

Override this method to perform tasks after printing, such as logging or sending a completion notification.

Syntax Sub OnFinishPrint( )

AcReport::OnStartPrint method

Called at the start of a print operation to perform custom tasks.

Syntax Sub OnStartPrint( )

AcReport::RoiIsTemporary method

Determines whether to keep the report object instance (.roi) file after the Factory generates the report. The default setting is to keep the ROI.

Syntax Function RoiIsTemporary( ) As Boolean

Returns True to discard the ROI.
False to keep the ROI.

AcReport::SetBurstReportPrivileges method

Override the SetBurstReportPrivileges( ) method of a burst report component to set all privileges on the current burst report. This method is called when each burst report begins building. The default setting is that the burst report has the same privileges as the report from which it originates.

Syntax Sub SetBurstReportPrivileges(row As AcDataRow)

Parameter **row**
The current data row instance.

AcReport::SetGlobalDHTMLCode method

Sets the custom code in a browser scripting control.

Syntax Function SetGlobalDHTMLCode(newValue As String)

AcReport::SetLayoutOrientation method

Sets the orientation of the report layout. Use the SetLayoutOrientation( ) method for right-to-left language support. This method sets the report and its subreports to right-to-left or left-to-right layout. The orientation is set regardless of the operating system or locale.

SetLayoutOrientation( ) can be called during report generation but not during report viewing.

Syntax Sub SetLayoutOrientation(newValue As AcLayoutOrientation)

Parameter **newValue**
The report orientation to set. Valid values are:

- RightToLeft
- LeftToRight

Example This example shows how to override the Finish method of a text control to set the report layout to right to left for an Arabic report. The layout is set at generation time.

```
Sub Finish( )
   Super::Finish( )
   ' Set Report layout to Right to left for Arabic data
   If DataValue = "Arabic" Then
      Container.GetReport( ).SetLayoutOrientation( RightToLeft )
   Else
      Container.GetReport( ).SetLayoutOrientation( LeftToRight )
    End If
End Sub
```

AcReport::SetROIAgingProperties method

Sets the autoarchive rules for a report object instance (.roi) file. Override SetROIAgingProperties( ) to change the autoarchive rules. You override SetROIAgingProperties( ) to set the autoarchive rules when you use report bursting to produce multiple ROIs and the individual files have different deletion or archive requirements.

To apply the rule to the individual file, call the Actuate Basic function SetStructuredFileExpiration, identifying the ROI to change. For additional information about SetStructuredFileExpiration, see *Programming with Actuate Basic.*

SetROIAgingProperties( ) affects only for reports running on iServer. The output file must be stored in an Encyclopedia volume.

Syntax Sub SetROIAgingProperties(fileID As Integer)

Parameter **fileID**
An identifier for the ROI.

Example The following example shows how to override the SetROIAgingProperties method to modify the file deletion rule for an ROI file. The file should be automatically deleted four hours after it is created. SetROIAgingProperties passes the identity of the file and the file deletion rule to the Actuate Basic function SetPOSMFileExpiration.

```
Sub SetROIAgingProperties( fileID As Integer )
   Dim expHours
   ' Force the file to expire 10 days after creation
   expHours = 10 * (24 * 60)
   SetPOSMFileExpiration( fileID, Age_NoOptions, Null, Null,
      expHours )
End Sub
```

AcReport::SuggestRoiName method

Specifies a name for the report object instance (.roi) file. The ROI name can include the protocol and path for this report. The ROI name also can include a value for a run-time parameter or data row variable, or the date the report runs.

The ROI name syntax is:

Syntaxes

[<protocol>:]/<path>/<report name>

Function SuggestRoiName( ) As String

Function SuggestRoiName(row As AcDataRow) As String

Parameters

<protocol>
The protocol to use to store the ROI. Table 7-83 lists the supported protocols.

Table 7-83 Supported protocols for storing an ROI

Protocol	Description
file	The destination is in a file system. The destination report appears in the viewing tool appropriate for the file type. For example, if the destination file is a PDF file, the report appears in Acrobat Reader. If the destination file is an Actuate report object instance (.roi) file, it appears in the view perspective.
http	The destination is on the web. The report appears in a web browser window.
none	If the source and destination are in the file system, the destination appears in the view perspective. If the destination is on the web, the destination appears in the web browser window.
other	The destination appears in a web browser window. Other protocols include FTP.

<path>/<report name>
The ROI path name. Paths can be absolute or relative.

Returns

The suggested ROI name specified using an absolute or relative path.

Examples

In the following example, the ROI name includes the value of the parameter, StateParam:

```
Function SuggestRoiName( ) As String
   SuggestRoiName = "State_" & CustQuery::StateParam & ".roi"
End Function
```

Custom ROI names generated by running this report application have names such as State_ca.roi or State_ny.roi.

In the following example, the report application uses report bursting to generate multiple ROIs from a single executable file. Each ROI contains a census report for a different state. Each state's report must be stored in an Encyclopedia volume folder named for the geographical region that contains the state. The Output File Parameter includes a folder name that shows the type of report, such as census reports.

```
Function SuggestRoiName( row As AcDataRow ) As String
   SuggestRoiName = row.GetValue( "Region" ) & "/" & "State_" &
      row.GetValue( "State" ) & ".roi"
End Function
```

Custom ROI names have the following form:

```
file:\C:\Census\Region\State.roi
```

where

- Region is the value of the region data row variable.
- State is the value of the state data row variable.

AcReport::TocAddComponent method

Adds the report to the table of contents.

Syntax Function TocAddComponent( ) As AcTocNodeType

AcReport::XMLDataProlog method

Creates the XML prolog for a custom XML data file. Override XMLDataProlog( ) to create a custom XML prolog in an XML data file. You can either completely replace the standard prolog or extend the standard prolog. To replace the standard prolog, do not call the superclass method. To extend the prolog, call the superclass method first and use the Actuate Basic Print statement to write additional prolog information to the channel.

Syntax Sub XMLDataProlog(channel As Integer)

Parameter **channel**
The Basic channel to which the XML prolog is written.

Class AcReportComponent

The base class for all sections, pages, frames, and controls. Figure 7-89 shows AcReportComponent.

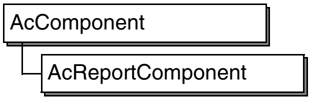


Figure 7-89 AcReportComponent

Description AcReportComponent is the base class for all reports, sections, frames, controls, page lists, flows, and pages. AcReportComponent establishes the core protocol for how components in a report are created and how they fit together in the report's containment structure. Build methods are key protocol elements. They specify the logic for creating the components and the components' contents. The container component calls Build( ). At the topmost level of the structure, the Build method for the report object creates the next level component, the report, and calls Build( ) for the report to create the report's contents. Each container component performs this task until all components and their contents are built.

Many Actuate report components, such as sections and frames, can contain one or more content components. AcReportComponent provides methods to identify containers and their contents. References to all these objects are stored in a list object. The AFC framework can easily traverse the list using the AcIterator class. Figure 7-90 shows the relationships between components in a report. It shows a group section, but the relationships are true for other types of sections as well.

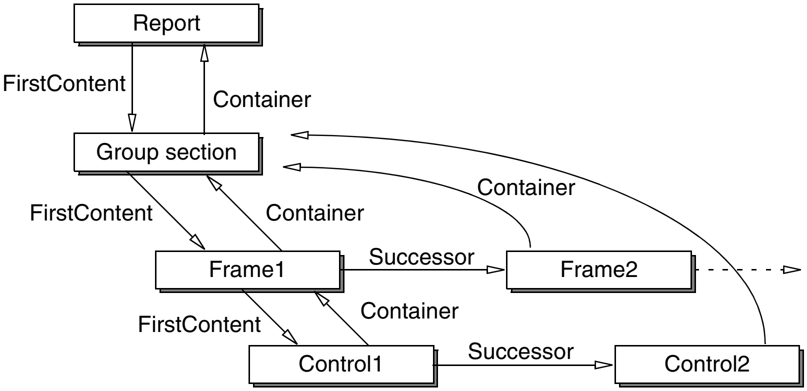


Figure 7-90 Relationships between report components

Customizing page-level security

The AcReportComponent, AcReport, and AcSection components implement page-level security. Page-level security is based on access control lists (ACLs), which are lists of security IDs. The Factory creates an ACL for the page and the view or print process creates an ACL for the current user. The view or print process determines whether the current user can view the page by comparing the page's ACL with the current user's ACL. If a page security ID matches a user's security ID, the page is visible to the current user. Developers modify the list of security IDs in the ACL for the page or the current user to customize page security.

Typically, you create an ACL by entering security IDs directly in the GrantExp property on the section or by entering an expression that evaluates to a list of security IDs. Nested sections inherit page security from their container components. The Factory builds an ACL for a page from the frame component's ACL. The Factory provides methods to dynamically change the contents of an ACL for a page. The view or print process provides a method to modify the content of the current user's ACL. The following process highlights the methods used by the Factory and the view process to build the ACL for the page and the current user.

About the Factory's role in page-level security

As the Factory builds a section, it passes the frames contained in the section to the page list. The Factory builds the ACL for the page in the following way:

- The Factory calls GetFullACL for the frame.
- The frame calls GetFullACL for its container section.
- The section calls GetComponentACL to get its own ACL and appends it to the ACL for the frame.
- Most of the time, the ACL corresponds to the GrantExp property for the section. If you want to customize the ACL for the section, override the SetSecurity method or create your own Actuate Basic function to do this.
- If the CascadeSecurity property is set to True (default), the section calls GetFullACL on its container section (if any). The section appends its ACL to the one returned by GetFullACL.
- The previous step is repeated until all container sections are processed. The ACL resulting from steps 1 through 4 is the ACL for the frame.
- If the ACL for the frame is different from the ACL for the previous frame, the Factory inserts a page break.
- The resulting ACL is assigned to the page.

To customize an ACL for a section and preserve the inheritance of security IDs from its containers, override the GetComponentACL( ) method. To customize an ACL for a section and prevent it from inheriting security IDs from its containers, set the CascadeSecurity property on the section to False.

About the view or print process roles in page-level security

- The view or print process obtains the current user's ACL.

  The view or print process obtains the current user's ACL from the Encyclopedia. The current user's ACL consists of the user's user ID, any roles associated with the user's group, and optionally, any security IDs supplied by the Report Server Security Extension (RSSE.) To create a custom ACL for the current user, override the GetUserACL method.
- The view or print process builds a list of visible pages for the current user.

  The view or print process compares the ACL for the current user to the ACL for the page to determine whether to add the page to the list of visible pages. If one of the security IDs for the current user is in the page's ACL, the view or print process adds the page to the list.

The view or print process uses the list of visible pages to build the table of contents and support display, print, and search operations.

Converting a report into XML

Actuate provides two ways to generate XML data from Actuate reports:

- Standard XML generation
- Custom XML generation

Generating standard XML

The XML Data property group on all content components, such as reports, sections, frames, and controls, specifies how to generate XML data for the component. The XML Data property group for the report component contains XML properties to generate the XML prolog and common characteristics for the XML file. The XML Data group for sections, frames, and control components consists of XMLAddContents, XMLAttributes, XMLTag, and XMLType. When you view a report containing XML data, you can select Save As XML Data to have the framework build an XML data output file.

Generating custom XML

Use AcReportComponent's GenerateXML( ) method to build XML elements, attributes, and text.

Subclassing AcReportComponent

Typically, you do not derive directly from AcReportComponent or override methods in this class. AcReportComponent establishes the containment and build protocols for all classes of persistent objects.

Variables

Table 7-84 lists AcReportComponent variables.

Table 7-84 AcReportComponent variables

Variable	Type	Description
Container	AcReport Component	The component that contains the current component.
RowCount	Integer	Counts the number of rows the section has processed. This variable is incremented at the start of BuildFromRow( ). Do not reset this counter. It is used internally by the section.
SearchTag	String	Identifies the component to search.
TocEntry	String	The table of contents entry text.

Properties

Table 7-85 lists AcReportComponent properties.

Table 7-85 AcReportComponent properties

Property	Type	Description
SearchTag	String	Uniquely identifies the component to search. To search multiple components as one group, specify the same value for each component. SearchTag is used in the user interface only if a value is specified and SearchAlias is not specified. If you specify a value for a report component, you can no longer use the scoped class name to identify the component in a search, such as in a URL. The default value is an empty string.
TocAdd Component	AcTOC NodeType	Determines whether the component name is added to the report's table of contents. The values are: ■ TOCAlwaysAdd. Always add the component to the table of contents. ■ TOCIfAllVisible. Add component name to the table of contents only if the user can view at least one page generated from the component based on page security. ■ TOCIfAnyVisible. Add component to table of contents even if the user cannot view any pages generated from the component based on page security.

(continues)

Table 7-85 AcReportComponent properties (continued)

Property	Type	Description
TocAdd Component *(continued)*	AcTOC NodeType *(continued)*	■ TOCSkip. Never add the component to the table of contents. Use this property to hide components such as parallel or sequential sections or detail frames from the user. The default value is TOCIfAllVisible.
TocAdd Contents	Boolean	Determines whether the component's contents are added to the report's table of contents.
TocValueExp	String expression	Returns a string to show as the table of contents entry for this object.
XMLAdd Contents	Boolean	Determines whether the Actuate XML includes the component's contents. The default value is True.
XML Attributes	String	A set of attribute values to add to the current XML element. You add attributes here instead of creating a control. These attributes are constant. They do not change based on data in the report. The following XMLType property settings determine where to add the attributes: ■ XMLElement. The attribute values are generated as attributes of the element before any attributes provided by controls. ■ XMLAttribute. The attribute values appear before the component's attribute value. ■ XMLIgnore. e.Report Designer Professional examines the value of the container's XMLType property. If the container's XMLType property is XMLElement, the attribute values generate as attributes of the container's element.
XMLTag	String	The name of the XML element or attribute for this component.
XMLType	String	The type of XML object, if any, the component represents. The values are: ■ XMLAttribute. The component is an XML attribute. ■ XMLCustom. A custom XML element. AFC calls GenerateXML( ) to generate the custom element. ■ XMLElement. The component is an XML element. ■ XMLEmptyElement. The component is an empty XML element. ■ XMLIgnore. The default setting. Do not generate XML for the component. ■ XMLText. The component is a text element.

See also Class AcPageList
Class AcSection
Class AcVisualComponent

Methods for Class AcReportComponent

Methods defined in AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcReportComponent::Abandon method

Removes a component that the report no longer needs. Abandon( ) removes and unpins the current component.

Syntax Sub Abandon( )

AcReportComponent::AddContent method

Adds a content component to the current component. This method raises an error if the current component does not allow contents.

Syntax Sub AddContent(component As AcVisualComponent)

Parameter **component**
The visual component to add to a container.

See also AcReportComponent::GetContainer method
AcReportComponent::GetContentIterator method
AcReportComponent::GetContents method
AcReportComponent::IsContainer method
AcReportComponent::IsLeaf method

AcReportComponent::Build method

Container objects call Build( ) and derived content classes override Build( ) to create their contents.

A report, for example, overrides Build( ) to create the pages in the report. Similarly, a frame overrides Build( ) to create the controls it contains.

Syntax Sub Build( )

See also AcReportComponent::BuildFromRow method

AcReportComponent::BuildFromRow method

Override this method to manipulate the data rows a report component processes. The framework calls BuildFromRow( ) so report components can use data from data rows to build themselves. This method is called for each data row a component is to contain. When BuildFromRow( ) returns False, the component did not process the row. You can override BuildFromRow( ) to change the way a report component processes data rows.

The framework calls BuildFromRow( ) in a report component's parent data section once with dataRow set to Nothing to tell a report component to finish building itself.

Within BuildFromRow( ) you can:

- Skip data rows by not calling Super::BuildFromRow( ) and returning ContinueBuilding.
- Create dynamic content based on values in a data row.
- Use calculated data in a component by creating your own data rows and calling Super::BuildFromRow( ) repeatedly.

Typically, it is easier to override a component's OnRow method instead of BuildFromRow( ) because OnRow( ) provides a simpler programming model. Only override BuildFromRow( ) if you cannot use OnRow( ).

When you override BuildFromRow( ), you must:

- Always handle the case where dataRow is Nothing.
- Always call Super::BuildFromRow(Nothing) to finish building the component.
- Always return FinishedBuilding when the component is done processing data rows.
- Always return FinishedBuilding if dataRow is Nothing.

Within BuildFromRow( ), you can use GetRowCount( ) to check how many rows the report component processes. The row count is incremented automatically when you call Super::BuildFromRow( ).

Syntax Function BuildFromRow(dataRow As AcDataRow) As AcBuildStatus

Parameter **dataRow**
A reference to a data row. If dataRow is Nothing, the report component must finish building itself.

Returns The build status of the report component:

- ContinueBuilding if the report component wishes to process further data rows. For example, a data control that is calculating an aggregate will return ContinueBuilding to indicate that it needs to see all the data rows in its parent section.
- FinishedBuilding if the report component is done processing data rows. For example, a data control that is not calculating an aggregate will return FinishedBuilding to indicate that it only needs to process a single data row.
- RejectedRow if dataRow does not belong to the report component. For example, a group section uses RejectedRow to indicate that a data row does not match its group key value.
- RejectedRow if dataRow is Nothing.

Examples By default, charts process multiple data rows. If a chart is placed in a Content frame, the result is a single chart that displays all the data rows for the Content frame's parent section.

In the following example, BuildFromRow( ) is overridden to make a chart process just one row. If the chart is in a Content frame, the result will be that a separate chart is displayed for each data row:

```
Function BuildFromRow( dataRow As AcDataRow ) As AcBuildStatus
   ' Process the first data row.
   BuildFromRow = Super::BuildFromRow( dataRow )
   If Not dataRow Is Nothing Then
      ' Force the chart to finish building itself.
      BuildFromRow = Super::BuildFromRow( Nothing )
   End If
End Function
```

In the following example, a frame's BuildFromRow( ) method has been overridden to add a data control to the frame if a customer's account is overdue. OverdueAmountControl is a control defined in a library; this control's value expression gets the overdue amount from the data row. GetRowCount( ) is used to avoid adding the control multiple times if the frame processes multiple data rows. The call to Super::BuildFromRow( ) must come after you add the control, to give the control a chance to process the data row:

```
Function BuildFromRow( dataRow As AcDataRow ) As AcBuildStatus
   If Not dataRow Is Nothing Then
      Dim myRow As DataRow
      If ( myRow.Customer_AccountStatus = "Overdue")
+     And ( GetRowCount( ) = 0 ) Then
         ' This is the first row - add the overdue amount
         ' control.
         Dim o As OverdueAmountControl
         Set o = New Persistent OverdueAmountControl
         ' Add the control to the frame.
         AddContent( o )
         ' Initialize the control.
         o.Start( )
      End If
   End If
   ' Pass the row to all the frame's contents (including the new
   ' control).
   BuildFromRow = Super::BuildFromRow( dataRow )
End Function
```

In the following example, BuildFromRow( ) filters out data rows for Massachusetts:

```
Function BuildFromRow( dataRow As AcDataRow ) As AcBuildStatus
   If Not dataRow Is Nothing Then
      Dim myRow As DataRow
      If ( myRow.customers_state = "MA" ) Then
         ' Do not process the row.
         BuildFromRow = ContinueBuilding
         Exit Function
      End If
   End If
   ' Process the row as usual.
   BuildFromRow = Super::BuildFromRow( dataRow )
End Function
```

See also AcReportComponent::GetRowCount method
AcReportComponent::OnRow method

AcReportComponent::DetachContent method

Most components are contained within another component. DetachContent( ) lets the container component, such as a frame, drop a contained component, such as a control. This method does not delete the component. The detached component remains in memory, which can lead to large amounts of memory consumption. The detached component remains in the report object instance (.roi) file but no longer appears in the report viewer.

Syntax Sub DetachContent(content As AcReportComponent)

Parameter **content**
The contained component to remove.

AcReportComponent::DetachFromContainer method

A content object, such as a control, calls DetachFromContainer( ) to detach the content object from its container, such as a frame. DetachFromContainer( ) does not delete the component. The component remains in the persistent report object instance (.roi) file but no longer appears in the report viewer.

Syntax Sub DetachFromContainer( )

AcReportComponent::FindContainerByClass method

Returns a reference to the named container object in the structure hierarchy. Use FindContainerByClass( ) to search the structure hierarchy for the container object with the named class. The class can be a member of the AFC library or a user-defined class. The search starts with the component initiating the search. If you search for the class corresponding to the component initiating the search, FindContainerByClass returns this component. To start the search on a higher level component, use the GetContainer method to position to the right level in the structure hierarchy.

Syntax Function FindContainerByClass(className As String) As AcReportComponent

Parameter **className**
The class name of the container object.

Returns A reference to the container object in the structure hierarchy with the named class.
Nothing if the container cannot be found.

See also For information about finding a container object in the page hierarchy, see AcVisualComponent::FindPageContainerByClass method.

AcReportComponent::FindContentByClass method

Use FindContentByClass( ) to search the structure hierarchy for a content component, such as a control or a section, by class name. The class can be a member of the AFC library or a user-defined class.

A search for the class name AcTextControl yields all the following components:

- TextControl
- PageHeaderFrame::TextControl
- OfficeGroup::PageHeaderFrame::TextControl

The search starts with the component initiating the search. For example, if a frame initiates the search, the framework looks first for controls within the frame. If the class is not represented in the current frame, the search extends outward to nested frames until either a matching class is found or there are no more objects to search.

To start the search on a higher level component, use GetContents( ) to position to the right level in the structure hierarchy.

Syntax Function FindContentByClass(className As String) As AcVisualComponent

Parameter **className**
The name of the class on which to base the search.

Returns Content components by class.

Example In the following example, FindContentByClass( ) finds a control within a flow and deletes it from the first page:

```
' Override the flow’s Finish( ) method to remove the control from
   the first page.

Sub Finish( )
   Super::Finish( )
   Dim iter As AcIterator
   Dim content As AcReportComponent
   Set iter = GetContentIterator( )
   Do while iter.HasMore()
      Set content = iter.GetNext( )
      Dim control As AcControl
      Set control = content.FindContentByClass( "LabelControl" )
      If (GetPageIndex( ) = 1) And Not (control Is Nothing)Then
         control.DetachFromContainer( )
         control.Abandon( )
      End If
   Loop
End Sub
```

See also AcReportComponent::GetContents method

AcReportComponent::Finish method

Contains the logic for completing an object.

Derived classes can override Finish( ) to do additional work when the Factory finishes processing the component. The derived version must always call the superclass version after doing the custom work.

After Finish( ) has finished processing, the Persistent Object Storage Mechanism (POSM) writes the objects to the report instance (.roi) file as needed. POSM optimizes memory usage by swapping objects in and out of memory.

Objects that are pinned to memory are not written to disk. Objects that hold references to transient objects must be pinned so they are not written to disk, maintaining the reference to the transient object. When an object is finished and ready to be written to disk, Finish( ) calls UnpinObject( ) to release the object.

UnpinObject( ) is an Actuate Basic function that works with POSM. UnpinObject( ) releases an object that was previously pinned to memory by PinObject( ).

Example See AcBaseFrame::GetControl method for an example showing how to use the Finish( ) method.

Syntax Sub Finish( )

See also AcBaseFrame::GetControl method
AcReportComponent::Start method

AcReportComponent::GenerateXML method

Generates XML for components that have an XMLCustom XML type. Override this method to generate custom XML for a component. You can build the XML attributes and elements by using the Actuate Basic Print statement to write the custom XML to the channel directly.

Syntax Sub GenerateXML(visitor As AcXMLDataVisitor)

Parameter **visitor**
The visitor component.

Example In this example, the code adds a comment to the custom XML that provides the date when the XML data was generated:

```
Sub GenerateXML( visitor As AcXMLDataVisitor )
   Dim channel As Integer
   channel = visitor.XMLFile
   Print #channel, "<!- Generated on "
   Print #channel, Today( ); " -!>"
End Sub
```

AcReportComponent::GetComponentACL method

Returns the access control list (ACL) for this component. Override GetComponentACL to modify the component's ACL. The ACL contains the security identifiers for users that can view pages built from the component. Most of the time, you enter the ACL in the GrantExp property for the section.

Syntax Function GetComponentACL( ) As String

Returns The ACL associated with the section component.
An empty string if the component is not a section.

See also AcReportComponent::GetFullACL method

AcReportComponent::GetConnection method

Returns the connection associated with this component. The GetConnection( ) method locates the connection by starting with the current component and looking upward through the structure hierarchy to find the first available connection. Sections can explicitly define a connection by placing the connection in the Connection slot, or implicitly by placing the connection inside the Connection slot of the data stream for that section. If a section does not have a connection available through one of these two means, then the framework continues searching with the next enclosing section until either a connection is found or until the search reaches the root of the report.

The framework uses this method to locate the connection to use for a data stream when you do not explicitly specify a connection. This allows you to create a sequential report that will print five subreports about your customer database. If you place the database connection on the topmost sequential section, then all the nested reports share this connection by using this method to search upward through the hierarchy to find the connection.

Syntax Function GetConnection( ) As AcConnection

Returns The connection associated with this component.

AcReportComponent::GetContainer method

Returns a reference to the container object for this component.

Syntax Function GetContainer( ) As AcReportComponent

Returns A reference to the container object for this component.
Nothing if this component does not have a container object.

See also AcVisualComponent::GetPageContainer method

AcReportComponent::GetContentCount method

Returns the number of content items in a component. For example, if the component is a section, GetContentCount( ) returns the number of content components in the section, including any in the Before or After slots. If the component is a frame, GetContentCount returns the number of controls and nested frames in the frame. If a component does not have contents, for example, a control, GetContentCount( ) returns 0.

Syntax Function GetContentCount( ) As Integer

Returns An integer greater than 0 if the component has contents.
0 if the component does not have contents.

AcReportComponent::GetContentIterator method

Returns an iterator over the contents of this component. The returned value is never Nothing. If the component cannot have contents, the default behavior is to create an iterator over an empty list. This behavior supports creating iterators over all components in a uniform manner.

Syntax Function GetContentIterator( ) As AcIterator

Returns An iterator over the contents of this component.

AcReportComponent::GetContents method

Returns a handle to the collection of contents for this component.

Syntax Function GetContents( ) As AcOrderedCollection

Returns A handle to the collection of contents for this component.
Nothing if this component does not support contents.

AcReportComponent::GetDataStream method

Returns the data stream that is associated with this component. The GetDataStream( ) method locates the data stream by starting with the current component and looking upward in the structure hierarchy to find the first available data stream. The search stops when either it finds a section that has a data stream defined or when it reaches the root of the report.

You can use this method to make multiple passes over data. For example, you can have a report of orders for a customer in which you want to both chart the orders and print them in detail. First, create a grouped orders-by-customer report. Then, add a custom nested report that makes a second pass over the data to create the chart. You must ensure that the data stream can be rewound by inserting a memory buffer filter. Also, be sure that a nested report leaves the data stream positioned at the same row as it was before the nested report started or the outer report produces incorrect results.

Syntax Function GetDataStream( ) As AcDataStream

Returns The data stream associated with this component.

AcReportComponent::GetFirstContent method

Retrieves the first content component. GetFirstContent( ) looks for and returns the first content component, such as a control or a section, of a report component or the AcReport component. You can then perform an action on the first component.

All report components can have contents. The type of contents depends on the component type. A frame, for example, contains controls and other frames. A section can contain frames and other sections. If multiple content components exist, GetFirstContent( ) returns the first one.

Syntax Function GetFirstContent( ) As AcReportComponent

Returns The first content component.

AcReportComponent::GetFirstContentFrame method

Retrieves the first Content frame, if any, for the current component. GetFirstContentFrame( ) looks for and returns the first Content frame, if any, for a component of a report. This method returns Nothing if there is no Content frame.

If the current component is a report or group section, GetFirstContentFrame( ) returns the first Content frame in that component, skipping the Before frame.

Syntax Function GetFirstContentFrame( ) As AcFrame

Returns The first content frame.
Nothing if there are no content frames.

AcReportComponent::GetFlow method

Returns a handle to the flow for this component.

Syntax Function GetFlow( ) As AcFlow

Returns A handle to the flow for this component.
Nothing if this component has no flow.

AcReportComponent::GetFullACL method

Returns the access control list (ACL) for this component combined with the other container components in the structure hierarchy. Call this method to retrieve a combined ACL for the component and all the other container components in the structure hierarchy. For example, if the report contains multiple group sections nested in a report section, the report section and each group section can have a separate ACL. If you call GetFullACL( ) on the first group section, GetFullACL( ) returns the union of the ACL for that group section and the report section because the report section is a container component in the group section's structure hierarchy. The other group sections are not included because they are not in the first group section's structure hierarchy.

Syntax Function GetFullACL( ) As String

Returns The ACL for the component and any other container component in the structure hierarchy. The result is a list of security IDs separated by commas.
Nothing if the component is not a section.

AcReportComponent::GetPage method

Returns the page that contains the component, then displays the page for an object that the user selects in the table of contents. You can also use GetPage( ) to get the page for a structural object, such as a section. Because a section is not visual, GetPage( ) retrieves the page that shows the header for that section.

To retrieve only the number of the page that contains the component, use GetPageIndex( ).

Syntax Function GetPage( ) As AcBaseFrame

Returns The page that contains the component.

See also AcReportComponent::GetPageIndex method

AcReportComponent::GetPageIndex method

Returns the number of the page that contains the component. GetPageIndex( ) returns the page number in the report, starting with 1. To retrieve the page that contains the component, use the GetPage( ) method.

Syntax Function GetPageIndex( ) As Integer

Returns The number of the page that contains the object.

See also AcReportComponent::GetPage method

AcReportComponent::GetPageList method

Returns the page list associated with the report that contains this component. The framework uses this method to add a new frame to the page list. You can use this method to get the page list if you want to start a new page or add a custom frame to the page list.

Syntax Function GetPageList( ) As AcList

Returns The page list associated with the report that contains this component.

AcReportComponent::GetReport method

Returns the report that contains this component. You can use this method if you create variables on the report that you want to access elsewhere in your application. The procedure is:

- Declare a variable of the type of the report.
- Call GetReport( ) to get the report and assign it to the new variable.
- Use that object reference variable to access the report variables.

Syntax Function GetReport( ) As AcReport

Returns The report that contains this component.

Example

```
Dim rptAs MyReport
Set rpt = GetReport ( )
BackgroundColor = rpt.NextColor
```

AcReportComponent::GetRowCount method

Returns the number of rows that this component has processed. For example, call this method if you need to perform custom processing if a row is the first row.

Syntax Function GetRowCount( ) As Integer

Returns The number of rows this component has processed.

AcReportComponent::GetSearchTag method

Returns the value of the SearchTag property.

Syntax Function GetSearchTag( ) As String

Returns The value of the SearchTag property if it is set.
An empty string if the value is not set.

AcReportComponent::GetTocEntry method

Retrieves the text of the TOC entry for a component.

Syntax Function GetTocEntry( ) As String

AcReportComponent::GetVisiblePageIndex method

Returns the page number of the visible page that contains the object. To retrieve the page that contains the component, use GetPage( ).

Syntax Function GetVisiblePageIndex( ) As Integer

Returns The number of the visible page that contains the object.

See also AcReportComponent::GetPage method

AcReportComponent::GetXMLText method

Returns the value for an XML attribute or element. If the component is a data control, GetXMLText( ) returns the value of the GetText( ) method formatted for XML. Override GetXMLText( ) to modify the data value for a custom XML format. For example, you can encode numbers as strings or translate codes from one set of values to another. If you override GetXMLText( ), you must return the XML value as a string using the standard XML quotes. Call ConvertToXML( ) at the end of your code to escape characters within strings that have special meanings in XML.

Syntax Function GetXMLText( ) As String

Example The following example shows one way to translate codes from one set of values to another. A control in the report design displays transaction type as Credit or Debit. The XML DTD defines the" transaction format as TransType="C & D" or "E & F".

The following code translates the data values to be consistent with the DTD:

```
Function GetXMLText( ) As String
   If DataValue = "Credit" Then
      GetXMLText = ConvertToXML("C & D")
   Else
      GetXMLText = ConvertToXML("E & F")
   End If

End Function
```

Returns The XML value in string format. The default return value is the value of GetText( ) formatted for XML if the control is a data control.
If the component is not a data control, returns a blank string.

See also For information about ConvertToXML( ), see *Programming with Actuate Basic.*

AcReportComponent::HasContents method

Determines whether the component has at least one content object.

Syntax Function HasContents( ) As Boolean

Returns True if the component has at least one content.
False if either the component cannot have contents or the list of contents is empty.

AcReportComponent::IsContainer method

Determines whether a component can have contents. This method is the opposite of IsLeaf( ).

Syntax Function IsContainer( ) As Boolean

Returns True if the component can have contents.
False if the component cannot have contents.

See also AcReportComponent::IsLeaf method

AcReportComponent::IsFlow method

Determines whether the component is a flow.

Syntax Sub IsFlow( ) As Boolean

Returns True if the component is a flow.
False if the component is not a flow.

AcReportComponent::IsFrame method

Determines whether the component is a frame.

Syntax Sub IsFrame( ) As Boolean

Returns True if the component is a frame.
False if the component is not frame.

AcReportComponent::IsLeaf method

Determines whether a component cannot contain contents. This method is the opposite of IsContainer( ).

Syntax Function IsLeaf( ) As Boolean

Returns True if the component cannot have contents.
False if the component can have contents.

See also AcReportComponent::IsContainer method

AcReportComponent::IsPage method

Determines whether the component is a page.

Syntax Sub IsPage( ) As Boolean

Returns True if the component is a page.
False if the component is not a page.

AcReportComponent::IsSubpage method

Determines whether the component is a subpage.

Syntax Sub IsSubpage( ) As Boolean

Returns True if the component is a subpage.
False if the component is not a subpage.

AcReportComponent::IsVisual method

Determines whether the component is a visual component such as an image or a data control.

Syntax Function IsVisual( ) As Boolean

Returns True if the component is a visual component.
False if the component is not a visual component.

AcReportComponent::OnRow method

Called for each new row. The Factory calls OnRow( ) to assign the expression entered in the ValueExp property to the data control. Override the OnRow( ) method to implement custom code to assign a value to a data control.

Controls fall into the following three categories, depending on their relationship to a data row:

- Need no data. Controls such as graphic images and lines require no data from the data row. These controls are called constant controls.
- Use data from a single row. The most common control is a data control that displays data from a single data row.
- Use data from multiple rows. Some controls summarize data from a set of rows. These controls are called aggregate controls.

Table 7-86 summarizes how OnRow( ) is called.

Table 7-86 Calling OnRow( )

If the number of rows the control uses is…	**OnRow( ) is called…**
0	Once with row = Nothing
1	Once with a single row
n	*n* times, each time with a different row

Override OnRow( ) only when you need to take control of the process for setting values.

Syntax Sub OnRow( )

Example The following example shows how to create a distinctive look for the sales reports of three sales offices within the same company.

The following code sets a custom variable, ContentsFrame, in OfficeGroup. When the code is finished executing, the variable ContentsFrame contains one of three possible frames, BostonFrame, NewYorkFrame, or PhiladelphiaFrame. The choice of the correct frame depends on the value of the offices_officeID variable of the current row. For instance, if offices_officeID is 1, the variable ContentsFrame contains BostonFrame.

This code example does not instantiate a frame as content for the OfficeGroup group section in Condtnl.rod. Instead, the code merely identifies which frame to instantiate as content. The OfficeGroup component's NewContent( ) method determines the correct frame to instantiate by inspecting the ContentsFrame variable you set in OnRow( ).

```
Sub OnRow( row As AcDataRow )
   Dim currentRow As ConditionalExampleDataRow
   Set currentRow = row
   Select Case currentRow.offices_officeID
      Case 1
         Set ContentsFrame = New Persistent BostonFrame
      Case 2
         Set ContentsFrame = New Persistent NewYorkFrame
      Case 3
         Set ContentsFrame = New Persistent PhiladelphiaFrame
   End Select

   ' Notice that the call to the superclass occurs here, at the
   ' end of the custom code

   Super::OnRow( row )
End Sub
```

AcReportComponent::SetSearchTag method

Sets the value of the SearchTag property. SearchTag uniquely identifies the component to search. To search multiple components as one group, specify the same value for each component.

SearchTag is used in the search interface only if a value is specified and SearchAlias is not specified.

Syntax Sub SetSearchTag(newTag As String)

AcReportComponent::SetTocEntry method

A generated method that sets the text of a TOC entry. This method uses the TocValueExp property value to assign table of contents names.

Syntax Sub SetTocEntry( )

AcReportComponent::Start method

The Start( ) method calls the PinObject function and prepares an object for the build process.

Derived classes can override Start( ) to do additional work when the Factory starts processing the component. The derived version must always call the superclass version before doing the custom work.

PinObject is an Actuate Basic function that works with the Persistent Object Storage Mechanism (POSM). POSM writes persistent objects to the report instance (.roi) file. It also optimizes memory usage by swapping objects in and out of memory as needed.

PinObject pins an object to memory so that it is not written to disk. Objects that hold references to transient objects must be pinned so that they maintain their references to the transient object.

Syntax Sub Start( )

See also AcReportComponent::Finish method

Class AcReportSection

A class that builds a report section from a data stream. Figure 7-91 shows the class hierarchy of AcReportSection.

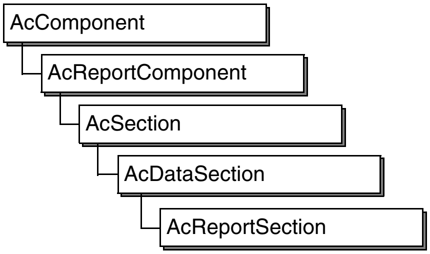


Figure 7-91 AcReportSection

Description AcReportSection builds a report using rows from a data stream. The report section is a type of data section. The report section provides Before and After slots, a Content slot, and page header and footer slots. The report section inherits a connection slot from AcSection and adds the slot to the specified data stream.

Class protocol

Table 7-87 shows how a report section works in the Factory.

Table 7-87 Class protocol for AcReportSection

Method	Task
New( )	Initializes the section.
Start( )	Prepares the report section for Factory processing. Start( ) instantiates and opens the connection, if any, and instantiates the data stream.
Build( )	Opens the data stream, builds the report by reading each row from the data stream, then processes it as described for AcDataSection.
BuildFromRow( )	Similar to Build( ), but provides the capability to create a nested report as described below.
Finish( )	Closes the data stream and connection.

Preparing the report section

Start( ) prepares the report section for processing each row. Start( ) manages the following tasks:

- Instantiates and opens the connection, if any, from AcSection::Start( )

- Obtains the data adapter by calling ObtainDataStream( )
- Sets the sort key by calling SetSortKey( )

Building the report

The Build( ) method builds the report using the following sequence:

- Starts the data stream by calling StartDataStream( ).
- Produces the PageHeader and Before components as described in AcDataSection.
- Reads a row from the data stream by calling its Fetch method. If there is no row, this method skips to the final step, producing the After and PageFooter components.
- If there is no current content, Build( ) calls NewContent( ) to create one.
- Passes the row to the content's BuildFromRow( ) method. If the content accepts the row, then loops back to step 3.
- Finishes the current content and instantiates a new one by calling NewContent( ).
- Calls the content's BuildFromRow( ) method. This time, the content must accept the row.
- Loops back to read the next row.
- Produces the After and PageFooter components as described in AcDataSection.

Build( ) is available for all reports except parallel reports. A parallel report calls BuildIntoFlow( ), not Build( ).

Working with data streams and connections

The report section centers on a data stream. Many data streams need a connection. The report section provides many options for assembling these components. This topic explains some options for placing a connection and controlling when the data stream is opened and closed.

Placing a connection

Typically when you create a report section, you place the connection in the Connection slot of the data stream itself. To share the connection with nested sections, you place the connection in the Connection slot of the report section.

If you place a connection in the Connection slot of a report section, the report opens the connection using the Start( ) method of the report section. To share a connection defined in a section that appears above the current report section in the structure hierarchy, leave both the report section and data stream Connection slots empty. The framework searches to find the shared connection. As described in AcSection, you can customize how the report obtains and opens the connection. For more information on this task, see the AcSection::ObtainConnection method.

Controlling the data stream

A report should contain a data adapter component in the Data Stream slot. If the report does not have a data adapter component, the report can still output the Before and After components but it produces no data rows.

A report's default behavior is to instantiate and opens the data adapter that appears in the Data Stream slot. To customize this behavior, you can override ObtainDataStream( ). Be sure the override returns an instantiated, started data adapter. If you override ObtainDataStream( ), you can also override FinishDataStream( ) to prevent the report from closing the data stream if your data stream is shared with other report sections.

Table 7-88 describes the life cycle of a data stream. You can override any of the methods in the middle column to customize the section or data adapter.

Table 7-88 Data stream life cycle

Protocol Method	Calls...	To...
Start( )	AcSection:: ObtainConnection( )	Create or locate the connection for this report.
	AcReportSection:: ObtainDataStream( )	Create or locate the data stream for this report. By default, this method calls NewDataStream( ) to instantiate the data stream.
	AcReportSection:: NewDataStream( )	Instantiate the data stream. By default, this method instantiates the data adapter specified in the Data Stream slot in Report Structure.
	AcReportComponent:: GetConnection( )	Called by ObtainDataStream( ), by default, to locate the connection to associate with the data stream.
Build( )	AcReportSection:: StartDataStream( )	Start the data stream, if any, returned by ObtainDataStream( ).
	AcDataAdapter::Fetch( )	Fetch each row from the data stream.
Finish( )	AcReportSection:: FinishDataStream( )	Finish processing the data stream, if any. By default, closes the data stream.
	AcSection:: FinishConnection( )	Finish processing the connection, if any. By default, closes the connection.

Creating nested reports

You sometimes need to create one report that nests inside another, even though the reports require different data sources. For example, suppose you have an Access database that lists the customers. You want to create a report that displays a list of the open orders for each customer. The orders reside in an Oracle database. To access data from both databases, take the following steps:

- Create a report section to query your Access database.
- In the Content slot of the outer report section, create another report section to query the orders you want to retrieve and print.

The inner report section opens its data stream at the beginning of its Build( ) method. To set a parameter on the inner query based on a value in the current data row in the outer query, override the BuildFromRow( ) method to get the customer ID from the outer data row, then pass this value to the query you created in the inner report section.

Properties

Table 7-89 lists AcReportSection properties.

Table 7-89 AcReportSection properties

Property	Type	Description
DataStream	AcDataAdapter	The data stream that provides rows for this report.
Sorting	AcSortingOptions	Determine whether the report section puts a sort filter in front of the data source in a particular report section. Valid values are: ■ AutoSort. The default setting. If AutoSort is set, the report section determines if the data source can sort data dynamically according to AddSortKey( ) calls. If the data source cannot sort dynamically, the report section instantiates a sort filter and places that filter in the chain of data adapters. The report section determines whether the data source yields data in the order expected by the group sections. ■ CompatibleSort. CompatibleSort provides backward compatibility with the AutoSort property of previous Actuate releases. If that AutoSort property is set to True to indicate that the data source can sort dynamically, Sorting is set to CompatibleSort. CompatibleSort means that the report section calls AddSortKey( ) but does not

(continues)

Table 7-89 AcReportSection properties (continued)

Property	Type	Description
Sorting *(continued)*	AcSortingOptions *(continued)*	invoke a sort filter. If the AutoSort property is set to False, the Sorting property is set to PreSorted. ■ PreSorted. If PreSorted is set, the data arrives from the data source already sorted in the order that data sections require the data. No attempt is made to tell the data source how to sort. The report section does not instantiate a sort filter.

Example The following examples show how to build nested reports using BuildFromRow( ):

```
Function BuildFromRow( row As AcDataRow) As AcBuildStatus
   If Not row Is Nothing Then
      CustomQuery::StateParam = row.GetValue("state")
   End If
   BuildFromRow = Super::BuildFromRow( row )
End Function
```

You can use a parameter with a connection as well as a query for a nested report. For example, you can have five department databases, each with the same schema, but with different names. You want to run an outer query that lists the server for each department, then an inner report that queries some data on that server. The report section normally opens its connection in the Start( ) method. For this example, you must write custom code that postpones opening the connection until your code reaches a call to BuildFromRow( ). To do so, you have to override two methods, ObtainConnection( ) and BuildFromRow( ).

```
Function ObtainConnection( ) As AcConnection
     'Instantiate, but do not open, the connection
     Set ObtainConnection = NewConnection( )
End Function

Function BuildFromRow( row As AcDataRow ) As Boolean
     Dim server        As ServerRow
     Dim deptConn      As DepartmentConnection
     'Get the data row
     Set server = row
     'Get the connection
     Set deptConn = GetConnection( )
     'Parameterize and open the connection.
     deptConn.ServerName = server.ServerName
     Verify( deptConn.Connect( ) )
     'Let the super class method do the actual building
     BuildFromRow = Super::BuildFromRow( row )
End Function
```

See also Class AcSection
AcReportComponent::GetConnection method

Methods for Class AcReportSection

Methods defined in Class AcReportSection

FinishDataStream, NewDataStream, ObtainDataStream, SetSortKey, StartDataStream

Methods inherited from Class AcDataSection

GetAfter, GetBefore, GetFirstPageFooter, GetFirstPageHeader, GetPageFooter, GetPageHeader, NewAfter, NewBefore, NewContent, NewPageFooter, NewPageHeader, OnEmptyGroup

Methods inherited from Class AcSection

CommittedToFlow, DeletePageFrame, FinishConnection, FinishFlow, FinishPage, GetComponentACL, GetCurrentRow, GetSearchValue, NewPage, ObtainConnection, PageBreakAfter, PageBreakBefore, SetSearchValue, SetSecurity, StartFlow, StartPage, StopAfterCurrentFrame, StopAfterCurrentRow, StopNow, TocAddComponent, TocAddContents

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcReportSection::FinishDataStream method

The default setting for FinishDataStream( ) closes the data stream for this report section by calling the data stream's Finish( ) method. You can override FinishDataStream( ) to keep the data stream open.

Syntax Sub FinishDataStream( )

See also AcReportSection::NewDataStream method
AcReportSection::ObtainDataStream method

AcReportSection::StartDataStream method

AcReportSection::NewDataStream method

Instantiates the component in the DataStream slot of the report section. The data stream component is transient. If you override NewDataStream( ), use the New keyword, not New Persistent.

Syntax Function NewDataStream( ) As AcDataAdapter

Returns The data adapter that was instantiated.

See also AcReportSection::FinishDataStream method
AcReportSection::ObtainDataStream method
AcReportSection::StartDataStream method

AcReportSection::ObtainDataStream method

Creates or locates the data stream to use for this report. The default setting for ObtainDataStream( ) instantiates the data adapter that you place in the DataStream slot. You can override this method to use another data stream.

ObtainDataStream( ) does not also open the data stream. To open the data stream, use StartDataStream( ).

Syntax Function ObtainDataStream( ) As AcDataAdapter

Example The following example shows how to query a database once and use the result set for more than one report or subreport. To view the code used in this example in a fuller context, open \Actuate11\eRDPro\Examples\DataAccess\ReuseQuery \ReuseQuery.rod. The example shows how a number of related methods work together in an Actuate report, in particular the ObtainDataStream( ), StartDataStream( ), and FinishDataStream( ) methods.

The report design in ReuseQuery.rod has a sequential section with two subreports, StateReport and CategoryReport. StateReport and CategoryReport use data from the same database query. So instead of each subreport making its own individual query, StateReport makes the query for both reports, stores the resulting data stream in BufDStream, and uses it. Then, CategoryReport picks up the data stream from BufDStream and reuses it.

In report component StateReport, override the Finish( ) method as shown in the following example:

```
Sub Finish( )
   ' At this point, we need to keep the data stream open.
   ' First instantiate sApp::BufDStream. BufDStream is a static
   ' variable of the sApp class. It is of type DataRowBuffer.
   ' Here, we prepare the variable to provide the datasource
   ' for CategoryReport later.
   Set sApp::BufDStream = DataSource
   'Rewind sApp::BufDStream now, so that when CategoryReport
   'uses it later, CategoryReport will start from the first row
   'of the buffer
   sApp::BufDStream.Rewind()
   Super::Finish( )
End Sub
```

To see BufDStream, open the Properties window on the sApp component, and choose the Variables tab. BufDStream is what holds the data for CategoryReport.

In CategoryReport, override the ObtainDataStream, StartDataStream, and FinishDataStream methods. The call to Super::ObtainDataStream( ) has been deleted in the following code, so that the overridden methods do not inherit the original ObtainDataStream( ) behavior:

```
Function ObtainDataStream( ) As AcDataAdapter
   ' CategoryReport will use sApp::BufDStream as its datastream.
   Set ObtainDataStream = sApp::BufDStream
End Function
```

The call to Super::StartDataStream( ) has been removed in the following code so that the method does not inherit the superclass' StartDataStream( ) behavior. The method intentionally does nothing.

```
Sub StartDataStream( stream As AcDataAdapter )
   ' We do not need to start the datastream since we are
   ' using sApp::BufDStream as CategoryReport's datastream.
End Sub
```

The following code performs cleanup tasks:

```
Sub FinishDataStream( stream As AcDataAdapter )
    Super::FinishDataStream( stream )
   ' We no longer need sApp::BufDStream, so set it
   ' to nothing.
   Set sApp::BufDStream = Nothing
End Sub
```

Returns The data adapter.

See also AcReportSection::FinishDataStream method
AcReportSection::NewDataStream method
AcReportSection::StartDataStream method

AcReportSection::SetSortKey method

Sorts the data rows by the keys specified in the data adapter. For example, you could have the data adapter sort first by customer ID, then by order number, and finally by line number. The default setting for SetSortKey( ) sets the sort key to the columns you specified in the Key property for any group sections in this report. You can override SetSortKey( ) to provide additional processing.

Syntax Sub SetSortKey (adapter As AcDataAdapter)

Parameters **adapter**
The data adapter that supplies the data rows.

AcReportSection::StartDataStream method

Opens the data stream for this report section. StartDataStream( ) prepares the data stream for reading. The default behavior for this method is to call the data stream's Start( ) method to open the data stream. You can override StartDataStream( ). For example, you can instruct StartDataStream( ) to do nothing if the data stream already exists and is open.

Syntax Sub StartDataStream(stream As AcDataAdapter)

Parameter **stream**
The data stream to use for this report section.

See also AcReportSection::FinishDataStream method
AcReportSection::NewDataStream method
AcReportSection::ObtainDataStream method

Class AcSection

The base class for all sections. Figure 7-92 shows the class hierarchy of AcSection.

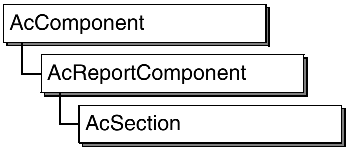


Figure 7-92 AcSection

Description A section is a structural component that builds the logical structure of the report. When you look at the report design in the layout window, or run and view the report, you do not see the sections. The visible sign of a section in a report is the effect it has on the organization of the visual components, such as frames and controls. In e.Report Designer Professional, sections appear in Report Structure.

A section has high-level control of the overall design of the report. It defines the structure of the report by determining when to open a data stream, what kind of processing to perform based on the rows in the data stream, and how and when to create frames. A section supports organizing data. It also supports viewing tasks such as searching, generating XML data, and extracting data from the report. Sections are persistent objects. They are written to the report object instance (.roi) file.

Using page-level security

The GrantExp property of AcSection and all the sections derived from AcSection determines which users can view a page the section produces. Using GrantExp, a report developer can specify an access control list (ACL) that consists of one security ID, a list of security IDs, or an expression that evaluates to one or more security IDs. A security ID can be either a user ID or a security role. Security IDs limit the visibility of pages to a certain user or set of users. If GrantExp is empty, any report user can view the pages that the section produces. An example of a GrantExp expression is

```
"Mgr" & [customers.State]
```

In this example, the roles on the Encyclopedia include managers at the state level, such as MgrCa and MgrFl for California and Florida, respectively. Using this page security scheme, managers can view pages showing data for their state only.

You can override the SetSecurity( ) method on AcSection to build a custom ACL. The default setting for SetSecurity( ) returns the security IDs in the GrantExp property. SetSecurity( ) provides access to the current data row to help you decide how to build the security IDs for the section.

If a report has nested sections, the default behavior is that nested sections inherit the ACL from their container sections. The CascadeSecurity property prevents page security on a section from being inherited from the section's container sections, if any. To define an ACL for a nested section and prevent container sections inheriting the ACL, take the following steps:

- Set the CascadeSecurity property of the container section to False.
- Set the GrantExp property to the security IDs for the nested section.

Understanding the types of sections

AcSection is the base class from which the Actuate framework sections are derived. Table 7-90 summarizes the types of sections. More detailed information about each type of section is provided in the individual class descriptions.

Table 7-90 Types of sections

Section type	Description	Example use
Report	Produces a series of frames from rows obtained from a data stream. Provides slots to create a connection and data stream. Contents are usually group sections, frames, or nested report sections.	Print a list of customers from a query against an ODBC database.
Group	Groups data on a common field, such as customers grouped by state.	Print orders for customers in various states.
Sequential	Contains several frames, charts, subreports, or sections that appear in a specified order.	Print two related reports, such as a sales history and a staffing history, one after the other.
Conditional	Uses a conditional expression to determine which of several frames, charts, subreports, or sections to include in the report.	Print a different frame for salaried, hourly, or commission employees.
Parallel	Contains two or more subreports that are displayed or printed simultaneously in different flows on the same page.	Present two related reports, such as employee addresses and salary histories, printed side-by-side for easy comparison.

Class protocol

AcSection provides a specific protocol for derived classes to follow. This protocol follows that set by AcReportComponent. The task descriptions in Table 7-91 identify the specific ways in which AcSection uses the standard protocol.

Table 7-91 Class protocol for AcSection

Method	Task
New( )	Initializes the section.

Table 7-91 Class protocol for AcSection

Method	Task
Start( )	Prepares the section for Factory processing.
Build( ) or	Builds the contents of the section. Called for the topmost section in a report and when no data row is available.
BuildFromRow( )	Builds the contents of the section when a data row is available. Called for sections nested inside a report or group section.
Finish( )	Finishes Factory processing.

Assigning a database connection to a section

The standard way to work with a database connection is to create the connection directly inside a data stream. This technique works if your report has a single data stream, or if each data stream uses a different connection, or if the data stream needs no connection. This technique is inefficient if your report has multiple data streams that work with the same connection and the connection is capable of processing multiple queries. It is more efficient to open the connection once and use it for multiple data streams. You do so by assigning the connection to a section instead of a data stream.

To determine the section to which to attach a connection, look in Report Structure. Find the section that is the common parent of all the data streams that need this connection. The common parent is often the topmost section of the report. Then, move the connection from the Connection slot of the data stream into the Connection slot of the parent section.

The section instantiates and opens the connection in its Start( ) method and closes the connection in its Finish( ) method. All nested sections call GetConnection( ) to search for this connection up the structure hierarchy. The connection is valid as long as its section is active.

Some reports use more than one type of connection. For example, in creating a sales report, you can find that most queries work for an Oracle sales database while one query requires an Access database. To use both databases, you can connect to the Oracle database in the topmost section, then, in the query that works with the Access database, create an ODBC connection specifically for that query. The local connection takes precedence over any connection defined higher in the structure hierarchy.

To keep two connections open, you must write custom code to maintain the second connection. In the preceding example, you can define a static variable to hold the connection, override the Start( ) method of your topmost section to open the additional ODBC connection, and override the Finish( ) method to close it. In the section where you need the second connection, override ObtainConnection( )

to return this second connection from the static variable you defined earlier, and override FinishConnection( ) to do nothing so that the nested section does not close this shared connection.

Interrupting a section

A section typically runs until it processes all the available data rows or section contents. In some circumstances, you want to stop processing early. For example, you can stop output after the first page or after a certain number of rows. The section class gives you three methods to stop processing. They differ in the amount of cleanup they perform, as follows:

- StopAfterCurrentRow( ) processes the current data row to completion before stopping. The section outputs the frame or frames, if any, for the row and outputs totals and other aggregates, depending on the kind of frame. You use this method to stop processing a section after a specific row or a specific number of rows. The resulting report looks as if the input data stream contained only rows up to the current row. All subsequent rows are silently ignored.
- StopAfterCurrentFrame( ) finishes the current frame by placing it on a page and produces no further output. The section does not display any totals. Note that finishing the frame can entail creating a new page to contain the frame.
- StopNow( ) stops the section. This method discards partially completed frames or partially processed data rows. Use this method to stop output at the end of a page. Do not create aggregates if you use this method because the aggregates will be incorrect.

Variables

Table 7-92 lists AcSection variables.

Table 7-92 AcSection variables

Variable	Type	Description
ContentList	AcList	The list of content component instances created for this section
SearchValue	String	The expression in the SearchValueExp property

Properties

Table 7-93 lists AcSection properties.

Table 7-93 AcSection properties

Property	Type	Description
Cascade Security	Boolean	Determines whether the subsection inherits the ACL(s) from its containers. Default is True. To enable or disable cascading page security, set the CascadeSecurity property of the container section. You can also disable cascading page security by overriding the AcReportComponent::GetFullACL method.
Connection	AcConnection Structure Reference	The connection, if any, to instantiate for this section.
GrantExp	Expression	The ACL for the section. The ACL can contain one or more security identifiers or an expression that evaluates to one or more security identifiers. If it contains multiple security identifiers, each security identifier must be separated by a comma. Spaces before or after a security identifier are ignored. The default for GrantExp is blank. Blank indicates that the section does not have any unique security restrictions. The section still inherits security restrictions, if any, from its containers in the structure hierarchy.
PageBreakAfter	Boolean Function	True if the following section or frame should start at the top of a new page.
PageBreakBefore	Boolean Function	True if the Factory should start a new page before starting this section. That is, True if this section should appear at the top of a new page.
SearchValueExp	Expression	The value to use to retrieve data. SearchValueExp can be a single data row or an aggregate expression.
Subpage	AcSubpage	The optional subpage to use with the parallel report. If there is no subpage, the report assumes that the flows are available on the page itself.

Methods for Class AcSection

Methods defined in Class AcSection

CommittedToFlow, DeletePageFrame, FinishConnection, FinishFlow, FinishPage, GetCurrentRow, GetSearchValue, NewPage, ObtainConnection, PageBreakAfter, PageBreakBefore, SetSearchValue, SetSecurity, StartFlow, StartPage, StopAfterCurrentFrame, StopAfterCurrentRow, StopNow, TocAddComponent, TocAddContents

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, BuildTocInfo, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcSection::CommittedToFlow method

The page list calls CommittedToFlow( ) for each registered section when the page list determines that the section is committed to the flow. The section is committed if the flow contains at least one content frame that is not a page decoration from the section or one of the section's content components, and there is no longer the possibility that the decoration can be removed and restarted on another flow.

Note that there is at least one call to StartFlow( ) before a call to CommittedToFlow( ) but there can be two calls, the first of which represents a failed attempt to start the section on an existing flow.

Syntax Sub CommittedToFlow(flow As AcFlow)

Parameter **flow**
The flow to which the section is committed.

AcSection::DeletePageFrame method

Deletes a frame in the section.

Syntax Sub DeletePageFrame(frame As AcFrame)

Parameter **frame**
The frame to delete.

AcSection::FinishConnection method

Closes the connection if the section has a connection. You override this method to do nothing if you override ObtainConnection( ) to return a shared connection. If the connection is shared, you can leave it connected so that a later section can continue to use it.

Syntax Sub FinishConnection(connection As AcConnection)

Parameter **connection**
The connection to close.

See also AcSection::ObtainConnection method

AcSection::FinishFlow method

Called at the end of each flow. The page list calls FinishFlow( ) for each active section at the end of each flow. There is one call to FinishFlow( ) for each call to StartFlow( ). This is the place to add page footers.

Syntax Sub FinishFlow( )

See also AcSection::FinishPage method
AcSection::StartFlow method
AcSection::StartPage method

AcSection::FinishPage method

Tells a section that a new page is finishing and provides an opportunity to insert custom code. If you override FinishPage( ), call the superclass version first.

In derived classes, the page list calls each component's FinishPage( ) method to check if the components add information to a page before the page finishes.

Syntax Sub FinishPage(page As AcBasePage)

Parameter **page**
A reference to the page that is ending.

See also AcSection::FinishFlow method
AcSection::StartFlow method
AcSection::StartPage method

AcSection::GetCurrentRow method

Returns the data row that the section is currently processing.

Syntax Function GetCurrentRow( ) As AcDataRow

Returns A reference to the current data row.

AcSection::GetSearchValue method

Gets the expression in the SearchValueExp property of the section.

Syntax Function GetSearchValue( ) As String

Returns The expression as a string.

AcSection::NewPage method

Determines which page type to use in this section. Page types include Letter, Legal, A4, A5, B4, B5, and custom types. The page you instantiate using this method takes precedence over the default page that the page list supplies.

Syntax Function NewPage( ) As AcPage

Returns An AcPage component that you subclass to choose the page type that conforms to your data set.

AcSection::ObtainConnection method

Creates a connection for this section. By default ObtainConnection( ) instantiates and opens the connection, if any, in the Connection slot of this section. You can override this method to return a shared connection.

Syntax Function ObtainConnection( ) As AcConnection

Returns The connection obtained for this section.

AcSection::PageBreakAfter method

Returns the value of the PageBreakAfter property. You can override this method to take control of the return value.

Syntax Function PageBreakAfter( ) As Boolean

Returns The value of the PageBreakAfter property.

See also AcSection::PageBreakBefore method

AcSection::PageBreakBefore method

Returns the value of the PageBreakBefore property. You can override this method to take control of the return value.

Syntax Function PageBreakBefore( ) As Boolean

Returns The value of the PageBreakBefore property.

Example This example shows how to conditionally set a page break. To view the code used in this example in a fuller context, perform the following steps:

1. Open Actuate11\eRDPro\Examples\DesignAndLayout\Detail\Detail.rod.
2. Choose View➛Libraries.
3. On Libraries, double-click PageBreakFrame.
4. On PageBreakFrame—Properties, choose Methods.
5. Scroll to find and inspect the frame’s overridden PageBreakBefore method.

The OfficeTitleFrame, SalesRepTitleFrame, CustomerTitleFrame, and OrderTitleFrame in the report design are all subclassed from this single PageBreakFrame component.

Suppose that there are many sales reps for any given office, and that in your sales report you want a page break before each new sales rep except the first. You want the first rep's information to appear on the same page as the office information.

The Detail report design uses the same logic to display information for each office, for sales reps, for customers, and for orders.

The sample report design performs the same kind of processing in four different contexts. For efficiency, it creates a new class that has the desired behavior, then subclasses it as needed. The code can be maintained in one place.

In this example, put the logic into a frame. The new class is called PageBreakFrame, and it is defined in the Sales.rol library.

The PageBreakBefore( ) method returns the value of the PageBreakBefore property. That means you can put conditional logic into the PageBreakBefore( ) method, and return True or False depending on whether or not this is the first frame instance for the current group. The following example shows how to make your report ignore the property setting and use only the programmed setting.

The call to Super::PageBreakBefore( ) is commented out.

```
Function PageBreakBefore( ) As Boolean
   'PageBreakBefore = Super::PageBreakBefore( )
   Dim myparent As AcDataSection

   PageBreakBefore = True

   'Inspect the parent of the containing group.
   'First assign an object reference.
   Set myparent = GetContainer( ).GetContainer( )

   'If it is the first row, do not perform a page break.
   If myparent.RowCount = 1 Then
      PageBreakBefore = False
   End If
End Function
```

The expression GetContainer( ). GetContainer( ) returns the container two levels up in the structure hierarchy.

When an instance of the frame OrderTitleFrame (subclassed from PageBreakFrame) evaluates this expression, the result is a handle to a CustomerGroup instance. When an instance of the frame SalesRepTitleFrame evaluates the same expression, the result is a handle to an OfficeGroup instance.

For example, in CustomerGroup, the variable RowCount tells how many orders that CustomerGroup instance has seen. The function of a CustomerGroup instance is to process all the orders for a single customer. If RowCount is 1, this is the first order. You do not want to set a page break before the first order. The same logic works for each of the four title frames you subclass from PageBreakFrame.

See also AcSection::PageBreakAfter method

AcSection::SetSearchValue method

Sets the value of the SearchValueExp property, using a data row as an argument.

Syntax Sub SetSearchValue(row As AcDataRow)

Parameter **row**
The data row on which to base the value of the SearchValueExp property.

AcSection::SetSecurity method

Generates the ACL from the GrantExp property for the section. Override SetSecurity( ) to generate a custom list of security identifiers. Set the ACL variable in SetSecurity( ) to a list of security IDs separated by commas. If no security restrictions exist, set the ACL variable to blank. The current data row is provided as input for generating the security identifiers.

Syntax Sub SetSecurity(row As AcDataRow)

Parameter **row**
The current data row for use in generating the ACL.

Example In the following example, the account type of the data row is used to determine if users having the role called MajorAccts, PrivateBanking, or Accounting can view the pages resulting from the section's content:

```
Sub SetSecurity( row As AcDataRow )
      Dim myRow As MyDataRow
      Set myRow = row

      If myRow.AccountType = "Commercial" Then
         ACL = "MajorAccts"
      ElseIf myRow.AccountType = "Private" Then
         ACL = "PrivateBanking"
      Else
         ACL = "Accounting"
      End If
End Sub
```

AcSection::StartFlow method

Called at the beginning of each flow. The page list calls StartFlow( ) for each registered section at the top of each new flow. There are one or more calls to StartFlow( ) for each call to StartPage( ). Override this method to add page headers and reserve space for page footers.

StartFlow( ) returns True if the section was successfully started on the flow, False if there was not enough room in the flow to contain the header, footer, or subpage for the section. In this case, the page list ends the current flow, starts a new flow, and calls this method again for the new flow.

Syntax Sub StartFlow(flow As AcFlow) As Boolean

Parameter **flow**
The flow that is starting.

See also AcSection::FinishFlow method
AcSection::FinishPage method
AcSection::StartPage method

AcSection::StartPage method

Called at the start of each new page.

Syntax Sub StartPage(page As AcBasePage)

Parameter **page**
A reference to the page that is starting.

See also AcSection::FinishFlow method
AcSection::FinishPage method
AcSection::StartFlow method

AcSection::StopAfterCurrentFrame method

Stops processing after the current frame is added to a page. StopAfterCurrentFrame( ) finishes the current frame by placing it on a page but then produces no more output. The section does not display totals. Finishing the frame can entail creating a new page to contain the frame.

Syntax Sub StopAfterCurrentFrame( )

See also AcSection::StopAfterCurrentRow method
AcSection::StopNow method

AcSection::StopAfterCurrentRow method

Stops processing after the current row is complete. The section processes the current data row before stopping. The section outputs the frame or frames, if any, for the row and totals, and so on, depending on the kind of frame. Call this method to stop processing a section after a given row or a given number of rows.

Syntax Sub StopAfterCurrentRow( )

See also AcSection::StopAfterCurrentFrame method
AcSection::StopNow method

AcSection::StopNow method

Stops the section from processing a data row. This method discards any partially completed frames or partially processed data rows. Use StopNow( ) when you want to stop output at the end of a page. Do not create aggregates if you use this method because the aggregates will be incorrect.

Syntax Sub StopNow( )

See also AcSection::StopAfterCurrentFrame method
AcSection::StopAfterCurrentRow method

AcSection::TocAddComponent method

TocAddComponent( ) adds the section to the table of contents.

Syntax Function TocAddComponent( ) As AcTOCNodeType

AcSection::TocAddContents method

If True, TocAddContents( ) adds the contents of a section to the table of contents.

Syntax Function TocAddContents( ) As Boolean

Class AcSequentialSection

A class that generates multiple, sequential components. Figure 7-93 shows the class hierarchy for AcSequentialSection.

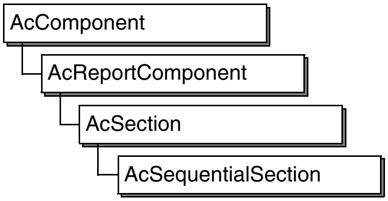


Figure 7-93 AcSequentialSection

Description Use AcSequentialSection to produce multiple reports within a single report object. The reports appear one after the other. For example, you can create a report that lists customer orders, followed by a report that shows overdue accounts. You can also produce multiple components for a single data row, such as a frame that displays customer information followed by a nested report that lists the customer's current orders.

The sequential section is a converter. It converts a slot that takes a single component, such as the Content slot of the top-level AcReport, into a slot that takes multiple components. The contents of a sequential section can be any kind of report component, including frames or other sections. The sequential section generates its contents in the order in which they appear in Report Structure.

You can write custom code that selects which components to generate and which components to skip by overriding the SelectContent( ) method.

Building a sequential section

The component that contains the sequential section calls Build( ) or BuildFromRow( ) for the sequential section. The container calls the Build( ) method if the container has no data row available, and calls BuildFromRow( ) if a data row is available. Within the sequential section, these two methods perform identical processing, except that Build( ) in turn calls the Build( ) method on its contents, and BuildFromRow( ) calls BuildFromRow( ) on its contents.

Constructing sections without input from data rows

Build( ) generates the contents of the sequential section in the order in which they appear in Report Structure. Because Build( ) takes no data rows, Build( ) calls the Build( ) method on each of the contents it builds. The following is the process that Build( ) uses to build the contents for the sequential section:

- Calls SelectContent( ) to determine whether to generate the first component

- Calls NewContent( ) to instantiate the component
- Calls the component's Start( ) method
- Calls the component's Build( ) method
- Calls the component's Finish( ) method
- Adds the component to the output page if necessary
- Loops back to process the next component

You typically do not override the Build( ) method. Instead, override SelectContent( ) or NewContent( ) on your subclass of AcSequentialSection, or override the Start( ), Build( ), or Finish( ) methods of the contained components.

Constructing sections with input from data rows

The processing for BuildFromRow( ) is identical to that for Build( ) except that this method calls the content's BuildFromRow( ) method instead of Build( ). It calls BuildFromRow( ) on the contents twice, once with the row passed to BuildFromRow( ), and a second time with a null data row to inform the content that no additional rows are available. Each sequential section can process only one data row. As a result, BuildFromRow( ) accepts the first row it receives, returning True. It rejects the second row, returning False.

You typically do not override BuildFromRow( ). Instead, override NewContent( ) or SelectContent( ) on the subclass of AcSequentialSection. Alternatively, you can override the Start( ), Build( ), or Finish( ) methods of the contents.

Property

Table 7-94 describes the AcSequentialSection property.

Table 7-94 AcSequentialSection property

Property	Type	Description
Content	AcReportComponent	Lists the sequential components to produce

See also Class AcSection

Methods for Class AcSequentialSection

Methods defined in Class AcSequentialSection

NewContent, SelectContent, StopAfterCurrentSection

Methods inherited from Class AcSection

CommittedToFlow, DeletePageFrame, FinishConnection, FinishFlow, FinishPage, GetComponentACL, GetCurrentRow, GetSearchValue, NewPage, ObtainConnection, PageBreakAfter, PageBreakBefore, SetSearchValue, SetSecurity, StartFlow, StartPage, StopAfterCurrentFrame, StopAfterCurrentRow, StopNow, TocAddComponent, TocAddContents

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, BuildTocInfo, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcSequentialSection::NewContent method

Instantiates one of the list of content components for the current section. The sequential section identifies its contents in order from 1 to the number of contents. The index passed to NewContent( ) corresponds to the position of the content component in the content list in Report Structure. The default behavior for NewContent( ) is to instantiate the component at the position given by index.

You can override NewContent( ) to decide which component to instantiate for a given index position. Note that overriding this method takes control of the component to instantiate for each index location. Any contents you specify in Report Structure are ignored. To change the set of components that the sequential section generates, you must change your code in this method. To indicate that there are no further contents in the sequential section, NewContent( ) returns Nothing. You therefore cannot instantiate components for indexes 1 and 2, none for 3, and instantiate a component for index 4. The sequential section never calls this method with index 4 if index 3 returns Nothing.

You do not need to completely replace the default behavior. You can, for example, control which component to instantiate for index 1 and call the superclass method to handle all other components.

Syntax Function NewContent(index As Integer) As AcReportComponent

Parameter **index**
The number of the content component in the content list.

Returns The component instance.
Nothing if the index is one greater than the number of contents in the section.

AcSequentialSection::SelectContent method

Supports conditionally selecting the contents of the sequential section to generate. The index passed to SelectContent( ) has the same meaning and value as the index passed to NewContent( ). The sequential section calls SelectContent( ) to determine whether the section must call NewContent( ) to instantiate the indexed component. SelectContent( ) returns True to instantiate the component, False to skip the component. If False, the sequential section increments the index and calls SelectContent( ) again. As a result, if you override SelectContent( ), you must ensure that it returns True for at least one index value for which NewContent( ) returns Nothing. Otherwise, the sequential section is locked in an infinite loop.

As an alternative to using SelectContent( ) to decide whether to produce a specific content, you can insert a conditional section between the sequential section and the component to conditionally select.

Syntax Function SelectContent(index As Integer, row As AcDataRow) As Boolean

Parameters **index**
The index of the sequential section.

row
If the sequential section is built using BuildFromRow( ), the row parameter gives you access to the data row provided by the container. If the section is built using Build( ), the row variable is Nothing.

Returns True if the section can include the specified component in the report.
False if the section cannot include the specified component in the report.

AcSequentialSection::StopAfterCurrentSection method

Stops processing the sequential section after the current frame.

Syntax Sub StopAfterCurrentSection( )

Example To stop after the current frame, you must tell the nested section, if there is one, to terminate, as shown in the following example:

```
Sub StopAfterCurrentFrame( )
   AcSection::StopAfterCurrentFrame( )
   If Not CurrentContent Is Nothing Then
      CurrentContent.Terminate( )
   End If
   StopAfterCurrentSection( )
End Sub
```

Class AcSimplePageList

Builds a page list that has pages of a single style. Figure 7-94 shows the class hierarchy for AcSimplePageList.

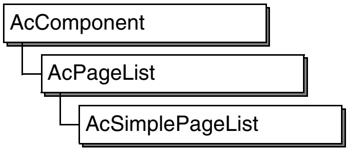


Figure 7-94 AcSimplePageList

Description Provides a report style in which all pages have the same layout.

Property

Table 7-95 describes the AcSimplePageList property.

Table 7-95 AcSimplePageList property

Property	Type	Description
PageStyle	AcPage	Specifies the single page style to use when creating the report

Methods for Class AcSimplePageList

Methods inherited from Class AcPageList

AddFrame, EjectPage, Finish, GetContentIterator, GetContents, GetCurrentFlow, GetCurrentPage, GetCurrentPageACL, GetEstimatedPageCount, GetFirstPage, GetLastPage, GetPage, GetPageCount, GetPageList, GetReport, HasPageSecurity, NeedCheckpoint, NeedHeight, NewPage, Start, UseAcceleratedCheckpoints

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcSingleInputFilter

A data filter that accepts input from one data adapter, processes the data, then passes it to the next data adapter or the report. Figure 7-95 shows the class hierarchy for AcSingleInputFilter.

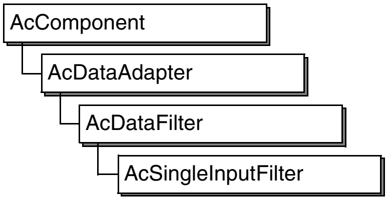


Figure 7-95 AcSingleInputFilter

Description AcSingleInputFilter is a data filter that accepts one data adapter as its input and filters each data row. You can create a derived class to define the filtering.

You can create filters to:

- Select certain rows and reject others. This type of filter is a selection filter.
- Convert a row from one format to another. This type of filter is a projection filter.
- Split large input rows into smaller rows needed by your report. For example, if an input row gives twelve months of financial data for each data row but your report needs the data organized as one month for each row, you can create a filter to split up the row.
- Combine data rows into a larger aggregate row. For example, you can combine data rows that contain one month of financial data for each row into a large row that contains twelve months of data.
- Add to fields in a data row by doing a lookup on an in-memory or disk-based table. This type of filter is a lookup filter. For example, you can do an in-memory lookup of a transaction code on each row to find its description, then copy the description into the data row.
- Sort rows. This type of filter is a sort filter.

There are many uses of a single input filter. If a report needs to combine several of the above transformations into a single data stream, report is easier to build, maintain, and understand if you create a separate filter for each transformation, then chain these transformations together to form the data stream.

If the data source is an SQL query, you can increase the performance of the report by doing as much filtering as possible in the SQL query. Note that data filters are optional, and that a data stream can have multiple data filters, as shown in Figure 7-96.

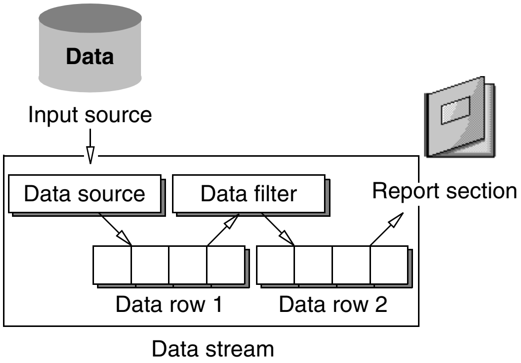


Figure 7-96 A data stream with multiple data filters

Using the input adapter

A data filter reads data from another data adapter called the input adapter. You can specify the input adapter in one of two ways. You can place the input adapter in the Input slot of the data filter in Report Structure. Alternatively, you can call SetInput( ) from code.

If you place the input adapter in the Input slot, the single input filter instantiates the input adapter. The filter provides the input adapter with a connection, starts the input adapter when the filter starts, and finishes the input adapter when the filter finishes.

If you set the input adapter with a call to SetInput( ), you can pass either an open or unopened data adapter. If you pass an opened data adapter, the single input filter assumes that the report will close this adapter, so the single input filter does not close the input adapter for you. If you pass an unopened input adapter, the single input filter takes responsibility for starting the input adapter when the filter starts and for finishing the input adapter when the filter finishes. If you call SetInput( ), do so before calling the Start( ) method for the filter.

Regardless of how you specify the input adapter, use the GetInput( ) method to access the input adapter.

If you specify an input adapter in Report Structure and also call SetInput( ) in your code, the input adapter passed to SetInput( ) takes precedence.

Creating a filter

To define a filter, override the Fetch( ) method of the data adapter. For more information on using this method, see the AcDataAdapter::Fetch method.

Variable

Table 7-96 describes the AcSingleInputFilter variable.

Table 7-96 AcSingleInputFilter variable

Variable	Type	Description
InputAdapter	AcDataAdapter	Refers to the data adapter that supplies input to this data filter

See also Class AcDataAdapter
Class AcDataSource
Class AcMultipleInputFilter

Methods for Class AcSingleInputFilter

Methods defined in Class AcSingleInputFilter

GetInput, NewInputAdapter, SetInput

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcSingleInputFilter::GetInput method

Returns the input adapter associated with this data filter.

Syntax Sub GetInput( ) As AcDataAdapter

Returns The input adapter for this data filter.

AcSingleInputFilter::NewInputAdapter method

Instantiates the input adapter. The NewInputAdapter( ) method instantiates the data adapter class, if any, that you dropped into the Input slot of the single input filter in Report Structure. You can override this method to programmatically decide which adapter to instantiate.

Syntax Function NewInputAdapter( ) As AcDataAdapter

Returns The new data adapter, if any.

AcSingleInputFilter::SetInput method

Sets the input adapter for this data filter. The adapter specified here takes precedence over the adapter in the Input slot in Report Structure.

Syntax Sub SetInput(adapter As AcDataAdapter)

Parameters **adapter**
The data adapter that supplies data rows to this filter.

Class AcSingleList

Implements a singly-linked list. Figure 7-97 shows the class hierarchy for AcSingleList.

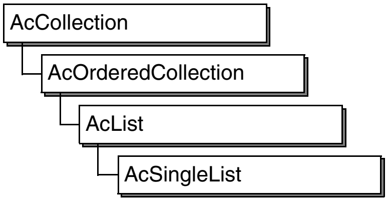


Figure 7-97 AcSingleList

Description AcSingleList, which derives from AcList, implements a singly-linked list. Use AcSingleList to process ordered lists, stacks, and queues. To randomly access collections of objects, use the AcObjectArray class. You should declare variables as AcList, then set them to AcSingleList to carry out the implementation of the singly-linked list methods.

You must subclass AcSingleList to look up items in a list by value. Override the inherited Compare( ) method to specify how to locate the objects by value. For an example of how to subclass AcSingleList, see AcList.

See also Class AcList

Methods for Class AcSingleList

Methods inherited from Class AcOrderedCollection

AddToHead, AddToTail, Copy, GetAt, GetHead, GetIndex, GetTail, InsertAfter, InsertAt, InsertBefore, RemoveHead, RemoveTail, SetAt

Methods inherited from Class AcCollection

Compare, Contains, Copy, FindByValue, GetCount, IsEmpty, NewIterator, Remove, RemoveAll

Class AcSqlQuerySource

A class that retrieves data from an SQL SELECT statement. Figure 7-98 shows the class hierarchy for AcSqlQuerySource.

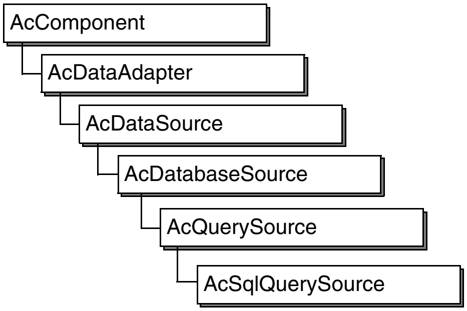


Figure 7-98 AcSqlQuerySource

Description AcSqlQuerySource is the base class for query data sources that you build in the query editor.

AcSqlQuerySource returns True from a call to CanSortDynamically( ) to indicate that custom sorting is supported. The AcDataAdapter class defines CanSortDynamically( ). If your custom subclass cannot support custom sorting, override CanSortDynamically( ) to return False.

You can create a query data source programmatically. You must either set the variables that hold the fragments of the SELECT statement or override ObtainSelectStatement( ) to return the complete statement. You must also override BindStaticParameters( ) to bind static parameters and BindDataRow( ) to bind the data row to the cursor.

Variables

Table 7-97 lists AcSqlQuerySource variables.

Table 7-97 AcSqlQuerySource variables

Variable	Type	Description
FromClause	String	The FROM clause
GroupByClause	String	The GROUP BY clause
HavingClause	String	The HAVING clause
OrderByClause	String	The ORDER BY clause
SelectClause	String	The SELECT clause
WhereClause	String	The WHERE clause

Property

Table 7-98 describes the AcSqlQuerySource property

Table 7-98 AcSqlQuerySource property

Property	Type	Description
Query	Pointer	An internal representation of the query that this query source class uses. The report developer uses Query Editor to assemble the query.

See also Class AcDataAdapter
Class AcDatabaseSource
Class AcDataRow
Class AcDataSource
Class AcDBConnection

Methods inherited from Class AcQuerySource

GetStatementText, ObtainSelectStatement, SetupAdHocParameters

Methods inherited from Class AcDatabaseSource

BindDataRow, BindStaticParameters, GetCursor, GetDBConnection, GetPreparedStatement, OpenCursor, SetStatementProperty

Methods inherited from Class AcDataSource

HasFetchedLast

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcStaticIndex

Implements a multi-layer n-way tree. Figure 7-99 shows the class hierarchy for AcStaticIndex.

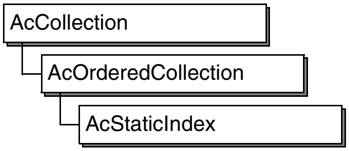


Figure 7-99 AcStaticIndex

Description AcStaticIndex provides fast indexing into a large collection of data. The primary use for AcStaticIndex is to index the list of pages for a report. The index is static because the number of contents must be known when you build the index and because you cannot insert objects into or remove objects from the index. You can, however, replace the object at a given index using the inherited SetAt( ) method.

In a static index, each node has a fixed number of child nodes. The default number is 100. You can change the default value. To ensure adequate performance, keep the node size reasonable.

You can create a static index by copying an existing collection into the index or building a new index. When you start from an existing collection, the index takes its size from the collection. When you build a new static index, you must specify the size.

See also Class AcCollection
Class AcOrderedCollection

Methods for Class AcStaticIndex

Methods defined in Class AcStaticIndex

AddLevel, New

Methods inherited from Class AcOrderedCollection

AddToHead, AddToTail, Copy, GetAt, GetHead, GetIndex, GetTail, InsertAfter, InsertAt, InsertBefore, RemoveHead, RemoveTail, SetAt

Methods inherited from Class AcCollection

Compare, Contains, Copy, FindByValue, GetCount, IsEmpty, NewIterator, Remove, RemoveAll

AcStaticIndex::AddLevel method

Called if necessary when building a static index of a particular size.

Syntax Sub AddLevel( )

AcStaticIndex::New method

Creates a new static index, setting the default size for each child node in the tree.

Syntax Sub New(theNodeSize As Integer)

Parameter **theNodeSize**
The default size for each child node in the tree.

Class AcStoredProcedureSource

Retrieves data from a stored procedure. Figure 7-100 shows the class hierarchy for AcStoredProcedureSource.

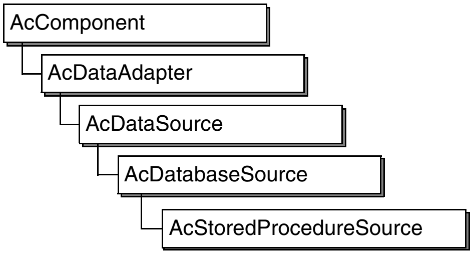


Figure 7-100 AcStoredProcedureSource

Description AcStoredProcedureSource is the base class for creating stored procedure data sources. You can access a result set returned from stored procedures on Oracle10g and higher clients using AcStoredProcedureSource. For information about using Actuate Basic to access complex result sets or execute stored procedures in other database environments, see *Accessing Data using e.Report Designer Professional.*

AcStoredProcedureSource contains the framework for executing a stored procedure including:

- Creating and managing parameters
- Creating and managing row variables
- Executing the stored procedure

About result sets

AcStoredProcedureSource supports procedures that return one result set. If your stored procedure returns multiple result sets and you need to process a result set other than the first one, you must override the OpenCursor( ) method to select a result set to be returned by name.

About parameters

AcStoredProcedureSource handles input, output and input-output stored procedure parameters. Stored procedure input parameters are similar to static parameters in the WHERE clause of a query.

If a stored procedure parameter has output parameters, AcStoredProcedureSource creates separate variables for these parameters. These parameter output variables are cleared when the data source is initialized and started to allow repeated execution of the stored procedure. AcStoredProcedureSource sets the variables as soon as the data is available from the stored procedure. From the standpoint of Actuate Basic, the data is available after the Finish( ) method on AcStoredProcedureSource completes.

The DefineInputParameter( ), DefineOutputParameter( ), DefineProcedureReturnParameter( ), GetOutputCount( ), and GetOutputParameter( ) methods help you define parameters and set and retrieve their values. These methods are only available if you are using a stored procedure data source. The methods are defined on AcDBCursor and AcDBStatement classes to support maximum programming flexibility.

About row variable creation

The AFC framework creates and names variables in the AcDataRow class for a stored procedure in a manner that is similar to the way that the variables are created when you use the AcSqlQuerySource class. When columns in the result set have no name, the framework creates variables in the AcDataRow class and names them sequentially starting with column1.

Variables

Table 7-99 lists AcStoredProcedureSource variables.

Table 7-99 AcStoredProcedureSource variables

Variable	Type	Description
CursorParameter	String	Specifies the name of the cursor for the result set to process. This name is only valid for Oracle stored procedures that use named cursors.
OwnerName	String	A database user name.
ProcedureName	String	Name of the stored procedure.
ProcedureStatus	Variant	Contains the return value or status for the stored procedure. Its value is available after the stored procedure source component’s Finish( ) method is complete.
QualificationOption	String	Specifies how the stored procedure call is to be qualified at report generation time.
QualifierName	String	A database qualifier for the stored procedure.

Property

Table 7-100 describes the AcStoredProcedureSource property.

Table 7-100 AcStoredProcedureSource property

Property	Type	Description
StoredProcedureDef	N/A	An internal representation of the stored procedure. You can edit this property only by using the Stored Procedure Builder.

After you select a stored procedure using the stored procedure browser, AcStoredProcedureSource retrieves the definition of the stored procedure from the database and stores the definition in StoredProcedureDef. This property is a complex structure that you can edit only by using the Stored Procedure Builder. AcStoredProcedureSource retrieves the following information from the database when a connection and stored procedure are specified:

- Procedure name. The name of the procedure.
- Owner name. A user name for the database within which the stored procedure is scoped.
- Qualifier name. A qualifier for the database within which the procedure is scoped. For Oracle10g stored procedures, the qualifier is the name of the database schema.
- Qualification option. Specifies how to qualify the stored procedure call at run time. The following three options are available:
  - Procedure name only
  - Owner.procedurename
  - Qualifiename.ownername.procedurename
- Return type information. The return type of the stored procedure's return value. Some stored procedures do not use a return value.
- Column information. Contains the following column information:
  - Column name. The name of an output column. This name is used to create a name for a variable in the row class.
  - Column type. Information about the data type of the column.
- Parameter information. Contains the following parameter information:
  - Parameter name. The name of a parameter in the stored procedure. This name is used to create a name for a parameter variable on the stored procedure class.
  - Parameter type. Information about the data type of the parameter.
  - Column kind. Parameters can be used for input, output, or both.

AcStoredProcedureSource uses column and parameter information to specify report data rows.

See also Class AcDataAdapter
Class AcDatabaseSource
Class AcDataRow
Class AcDataSource
Class AcDBConnection
Class AcSqlQuerySource

Methods for Class AcStoredProcedureSource

Method defined in Class AcStoredProcedureSource

GetOutputParameters

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcDataSource

HasFetchedLast

Methods inherited from Class AcDatabaseSource

BindDataRow, BindStaticParameters, GetCursor, GetDBConnection, GetPreparedStatement, OpenCursor, SetStatementProperty

AcStoredProcedureSource::GetOutputParameters

Gets the output parameters for the stored procedure. Does nothing if there are no output parameters.

Syntax Sub GetOutputParameters(cursor as AcDBCursor)

Parameter **cursor**
The cursor associated with the stored procedure's output parameters.

Class AcSubPage

A subpage fits inside a flow. Figure 7-101 shows the class hierarchy for AcSubPage.

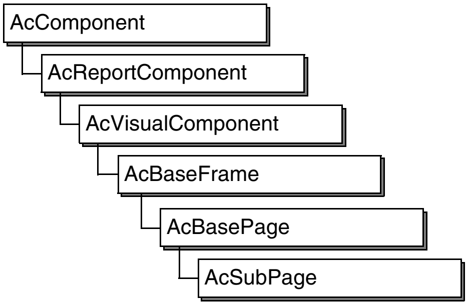


Figure 7-101 AcSubPage

Description AcSubpage supports a report developer using more than one set of flows within a page. Use a subpage to switch dynamically from one column to two columns on the same page. For example, consider a report that lists orders and the items on the order. You need the order information to fill the full width of the page. If the item information is short enough to list in two columns, you can add a subpage to the design and create two flows within the subpage. e.Report Designer Professional places the subpage inside the flow in the original page and all subsequent output goes into the subpage. You can ensure that the contents of each flow are as close as possible to the same height by setting the subpage's BalanceFlows property to True.

Methods for Class AcSubpage

Methods inherited from Class AcBasePage

BalanceFlows, GetFirstDataFrame, GetLastDataFrame

Methods inherited from Class AcBaseFrame

AddToAdjustSizeList, BindToFlow, FindContentByClass, FindContentByClassID, GetControl, GetControlValue, GetPageNumber, GetSearchValue, IsDataFrame, IsFooter, IsHeader, MakeContents, RebindToFlow, SearchAttributeName

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, BuildTocInfo, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcTextControl

Displays a String value. Figure 7-102 shows the class hierarchy for AcTextControl.

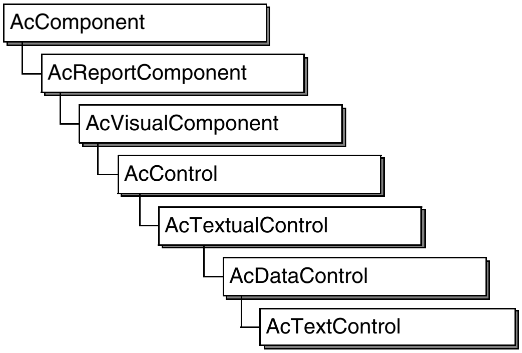


Figure 7-102 AcTextControl

Description Use the text control to display a String value. You can also use a dynamic text control or label control to display text.

See also Class AcControl
Class AcDataControl
Class AcDynamicTextControl
Class AcLabelControl
Class AcTextualControl

Variable

Table 7-101 describes the AcTextControl variable.

Table 7-101 AcTextControl variable

Variable	Type	Description
DataValue	String	Holds the string value

Methods for Class AcTextControl

Methods inherited from Class AcDataControl

Format, GetGroupKey, IsSummary

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, BuildTocInfo, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcTextQuerySource

Provides a way to write a textual SQL SELECT query. Figure 7-103 shows the class hierarchy for AcTextQuerySource.

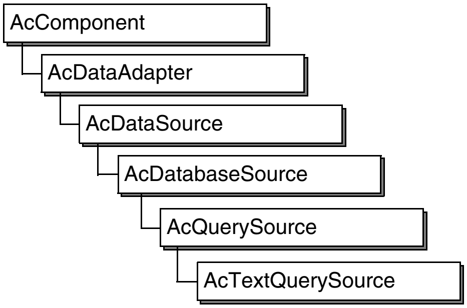


Figure 7-103 AcTextQuerySource

Description AcTextQuerySource is the class for query data sources that you build in the textual query editor.

You can create the query data source programmatically. In this case, you must override ObtainSelectStatement( ) to return the complete statement. ObtainSelectStatement( ) is inherited from AcQuerySource. You must also override BindStaticParameters( ) to bind static parameters and BindDataRow( ) to bind the data row to the cursor. BindStaticParameters( ) and BindDataRow( ) are defined in AcDatabaseSource.

Properties

Table 7-102 lists AcTextQuerySource properties.

Table 7-102 AcTextQuerySource properties

Property	Type	Description
CanModifyOrderByClause	Boolean	Specifies whether the application can modify the SELECT statement's Order By clause to provide custom sorting used by the corresponding report section. The default value is True.
Query	Pointer	An internal representation of the query that this query source class uses. The report user uses the textual query editor to specify the query.

Example The following example specifies ad hoc conditions by overriding the SQL statement. Then, the code overrides SetupAdHocParameters( ) to call SetAdHocCondition( ) with the appropriate input arguments.

```
Function ObtainSelectStatement( ) As String
   SelectStatement = "SELECT * from offices WHERE :?myOffice"
   Super::ObtainSelectStatement( )
End Function

Sub SetupAdHocParameters( )
   ' myOfficeID is a parameter defined in this class
   SetAdHocCondition( "myOffice", "officeID", "Integer",
+     myOfficeID )
End Sub
```

Methods for Class AcTextQuerySource

Methods inherited from Class AcQuerySource

GetStatementText, ObtainSelectStatement, SetupAdHocParameters

Methods inherited from Class AcDatabaseSource

BindDataRow, BindStaticParameters, GetCursor, GetDBConnection, GetPreparedStatement, OpenCursor, SetStatementProperty

Methods inherited from Class AcDataSource

HasFetchedLast

Methods inherited from Class AcDataAdapter

AddRow, AddSortKey, CanSeek, CanSortDynamically, CloseConnection, Fetch, Finish, FlushBuffer, FlushBufferTo, GetConnection, GetPosition, IsStarted, NewConnection, NewDataRow, OpenConnection, Rewind, SeekBy, SeekTo, SeekToEnd, SetConnection, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcTextualControl

The base class for label controls and data controls. Figure 7-104 shows the class hierarchy for AcTextualControl.

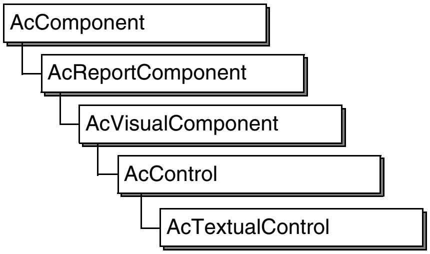


Figure 7-104 AcTextualControl

Description AcTextualControl is the base class for label controls and data controls.

Properties

Table 7-103 lists AcTextualControl properties.

Table 7-103 AcTextualControl properties

Property	Type	Description
BackgroundColor	AcColor	The background color of the control
Border	AcLineStyle	Defines the border, if any, around the control
Font	AcFont	The default font for text in the control
Margins	AcMargins	The margins around the text in the control
TextPlacement	AcTextPlacement	Specifies where to place the text within the control, and how wrapping and truncation are to be applied

Methods for Class AcTextualControl

Methods inherited from Class AcControl

BalloonHelp, GetControlValue, GetText, GetValue, PageNo, PageNo$, SetDataValue

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, BuildTocInfo, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcTitleBodyPageList

Builds a page list with a title page, followed by a simple page list. Figure 7-105 shows the class hierarchy for AcTitleBodyPageList.

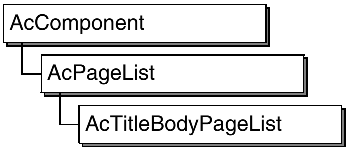


Figure 7-105 AcTitleBodyPageList

Description The AcTitleBodyPageList class builds a page list with a title page, followed by all other pages of another style.

You can insert title pages into your report. For example, you can design a report that prints the customer name in a large font on a single page, then produce body pages as necessary for that customer. You can repeat the title page for all customers.

Properties

Table 7-104 lists AcTitleBodyPageList properties.

Table 7-104 AcTitleBodyPageList properties

Property	Type	Description
BodyPage	AcPage	The page style to use for all pages other than the first page
TitlePage	AcPage	The page style to use for the first page

Methods for Class AcTitleBodyPageList

Methods inherited from Class AcPageList

AddFrame, EjectPage, Finish, GetContentIterator, GetContents, GetCurrentFlow, GetCurrentPage, GetCurrentPageACL, GetEstimatedPageCount, GetFirstPage, GetLastPage, GetPage, GetPageCount, GetPageList, GetReport, HasPageSecurity, NeedCheckpoint, NeedHeight, NewPage, Start, UseAcceleratedCheckpoints

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

Class AcTopDownFlow

Adds frames to a flow from the top to the bottom. Figure 7-106 shows the class hierarchy for AcTopDownFlow.

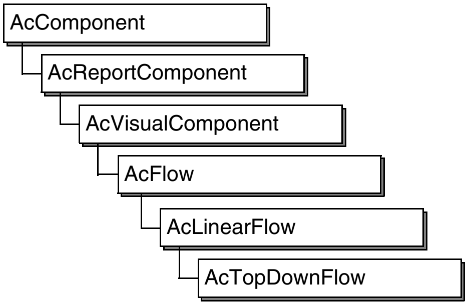


Figure 7-106 AcTopDownFlow

Description The top-down flow fills with frames in the standard top-down order. The first frame starts at the top of the flow. Each subsequent frame is placed just below the previous frame. When the framework receives a frame that does not fit in the remaining space in the flow, the frame advances to the next flow or page.

You subclass AcTopDownFlow when you drag a flow component from the toolbox and drop it in the report design.

Property

Table 7-105 describes the AcTopDownFlow property.

Table 7-105 AcTopDownFlow property

Property	Type	Description
Alignment	AcFlowPlacement	Specifies how to align a frame within a flow. Values are: ■ FlowAlignLeftOrTop. Causes the frame to appear left-justified within the flow. ■ FlowAlignCenter. Centers the frame in the flow. ■ FlowAlignCustom. Supports custom alignment. If you choose custom alignment, e.Report Designer Professional uses the value of the X property in the Position property group to align the frame in the flow. ■ FlowAlignRightOrBottom. Causes the frame to appear right-justified.

Methods for Class AcTopDownFlow

Method defined in Class AcTopDownFlow

AdjustFooter

Methods inherited from Class AcLinearFlow

GetInsideOrigin, GetInsideRect

Methods inherited from Class AcFlow

AddFooter, AddFrame, AddHeader, AddSubpage, AdjustFooter, CanFitFrame, CanFitHeight, GetFirstDataFrame, GetLastDataFrame, GetFreeSpace, GetInsideSize, IsEmpty, ReleaseSpace, ReserveSpace, ResetSpace, ResizeByConstrainedByContents, ShiftFooterUp

Methods inherited from Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, BuildTocInfo, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcTopDownFlow::AdjustFooter method

Adjusts the position of the top of the page footer frame to allow for size changes. If you use AdjustFooter( ), it is possible that some footer text is still not visible if the footer changes size. The best practice is to set the footer to the largest size necessary to accommodate all possible information.

Syntax Sub AdjustFooter(footer as AcFrame)

Parameter **footer**
The page footer frame to adjust.

Class AcVisitor

Provides a mechanism for creating a utility to visit and perform an action on report components. Figure 7-107 shows the class hierarchy for AcVisitor.

Figure 7-107 AcVisitor

Description

The AcVisitor class creates a mechanism that visits and performs an action on a report component. For example, use AcVisitor to perform data extraction or save the report to a different format, such as PostScript or a text file. AcVisitor methods provide subroutines to process a report component and its contents in hierarchical order. Override AcVisitor methods to provide specialized behavior for each component.

A report has the following two parallel structures:

- The structure hierarchy, composed of the report, sections, frames, and controls
- The page hierarchy, composed of the report, page list, pages, flows, frames, and controls

When you use the AcVisitor class, you determine which hierarchy to visit. The default behavior is to visit each component in the hierarchy you specify. If the component contains other components, the visitor also visits each of those components in the order in which they appear in Report Structure. You can derive AcVisitor classes that skip certain components or add behavior to specific components.

To use the AcVisitor methods, perform the following tasks:

- Open a report design in e.Report Designer Professional.
- Set up the visitor class:
    - Create a new Actuate Basic source file.
    - Create a new subclass of the AcVisitor class and instantiate it. The visitor class provides a visit method for each type of component. You can override the method for a particular component if you want to perform operations on it.
    - Decide which report component is the visitor's starting point. For example, to visit every component in a report, call the AcReport component.
    - Call ApplyVisitor( ) on the starting point. The default behavior for a component is to call Visit( ) for the superclass of the component. For example, VisitTextControl( ) calls VisitDataControl( ). Similarly, every method ultimately calls VisitComponent( ).
- Decide which components to visit:

  - If the application will visit the entire report, the derived visitor class must know which of the report's hierarchies to visit. Override VisitReport( ) to call either VisitContents( ) to visit the data hierarchy or VisitPage( ) to visit the page hierarchy. The default call is to VisitContents( ).
  - If the application will not visit the entire report, decide which components to include or exclude from processing. If the component has contents, the visit method for the contents is called by default. If you want to exclude the contents from being processed, override the component's visit method to inhibit the call to the component's superclass method.

- Add special behavior for the visited components.

  Override each component's visit method to add special behavior as needed. For example, suppose that you want to generate Postscript for a report. Override VisitPage( ) to eject each page, and override Visit( ) for each control to generate Postscript for that control.
- Establish a mechanism to trigger visitor subroutines.

Example The following example shows how to design a utility to perform data extraction using the AcVisitor class. The example uses visit methods to traverse the structure hierarchy in the Sales Detail report component to extract the data controls for each frame to a spreadsheet that can be viewed using Microsoft Excel. The user selects a visual control in the report component to initiate the extraction process. When data extraction is complete, the example application starts Microsoft Excel, which displays the spreadsheet.

The example features a report design that includes an Actuate Basic source file. The Actuate Basic source file performs the following programming actions:

- Initializes the Basic program environment by performing the following actions:
  - Declaring the class, AcDetailCsvVisitor, as a subclass of AcVisitor
  - Declaring state variables needed for internal processing
  - Creating an instance of the output text file (spreadsheet)
  - Declaring text file fields corresponding to the Sales Detail controls to be extracted
  - Declaring subroutines that convert the values of the data controls to the form needed in the output text file
  - Declaring subroutines to do special processing for different kinds of frames:

```
Class AcDetailCsvVisitor Subclass Of AcVisitor
   'State variables:
   'Channel contains system file number
   'NeedComma is needed to determine when to write a comma
     to the file FileName contains output text file name
```

```
Dim Channel As Integer
Dim NeedComma As Boolean
Dim FileName As String

' Variables that make up the data row.
Dim TotalSalesForecast As Currency
Dim OfficeName As String
Dim OfficePhone As String
Dim OfficeAddress1 As String
Dim OfficeAddress2 As String
Dim OfficeTotalForecast As Currency
Dim RepTotalForecast As Currency
Dim RepExtension As String
Dim RepEmail As String
Dim RepName As String
Dim CustomerContactName As String
Dim CustomerContactPhone As String
Dim CustomerName As String
Dim CustomerAddress1 As String
Dim CustomerAddress2 As String
Dim CustomerCreditRank As String
Dim CustomerPurchPattern As String
Dim CustomerTotalForecast As String
Dim OrderNumber As Integer
Dim OrderForecastDate1 As String
Dim OrderNeededDate As String
Dim OrderNote As String
Dim OrderForecastDate2 As String
Dim OrderStatus As String
Dim ItemCategory As String
Dim ItemCode As String
Dim ItemDescription As String
Dim ItemQuantity As Integer
Dim ItemPrice As Currency
Dim ItemExtension As Currency
Dim OrderTotalQuantity As Integer
Dim OrderTotal As Currency
Dim CustomerTotalQuantity As Integer
Dim RepTotalQuantity As Integer
Dim OfficeTotalQuantity As Integer
Dim TotalQuantity As Integer

' Create instance of output text file
Sub New( )
   FileName = "extract.csv"
End Sub
```

```
' Convert a number to a field in the output line.
Sub NumericField( value As Variant )
   If NeedComma Then
      Print #Channel, ",";
   End If
   Print #Channel, CStr( Value );
   NeedComma = True
End Sub

' Convert a string to a field in the output line. The
' string field is enclosed by quotation marks. Any
' quotation mark in the string is replaced by a two
' quotation marks. Newline characters are converted to
' spaces. So, a value of He said "Hi!" becomes:
' "He said ""Hi!!"""

Sub TextField( value As String )
   Dim i As Integer
   Dim length As Integer
   Dim c As String
   If NeedComma Then
      Print #Channel, ",";
   End If
   Print #Channel, """";
   length = Len( value )
   For i = 1 to length
      c = Mid$( value, i, 1 )
      Select Case c
         Case """"
            Print #Channel, """""";
         Case Chr$(13)
            Print #Channel, " ";
         Case Chr$(10)
            Print #Channel, " ";
         Case Else
            Print #Channel, c;
      End Select
   Next
   Print #Channel, """";
   NeedComma = True
End Sub
```

- Overrides the VisitReportSection( ) method to perform the following actions:
  - Opening the text file for output
  - Writing labels for the text fields
  - Calling the VisitDataSection( ) method to start the extraction

- Closing the text file after VisitDataSection( ) has completed data extraction:

```
Sub VisitReportSection( obj As AcReportSection )
' Open output text file
   Channel = FreeFile( )
   Open FileName For Output As #Channel

' Write column labels
   Print #Channel, "OfficeName,";
   Print #Channel, "SalesRepName,";
   Print #Channel, "ContactName,";
   Print #Channel, "ContactPhone,";
   Print #Channel, "CompanyName,";
   Print #Channel, "CreditRank,";
   Print #Channel, "PurchasingPattern,";
   Print #Channel, "OrderNumber,";
   Print #Channel, "ForecastOrderDate,";
   Print #Channel, "NeededOrderDate,";
   Print #Channel, "ForecastOrderShipDate,";
   Print #Channel, "OrderStatus,";
   Print #Channel, "ItemCategory,";
   Print #Channel, "ItemDescription,";
   Print #Channel, "ItemQuantity,";
   Print #Channel, "ItemPrice,";
   Print #Channel, "ItemExtendedPrice,";
   Print #Channel

' Start data extraction from report component to
' outputtext file
   VisitDataSection( obj )
' Close output text file and end processing after
' extraction is complete
   Close #Channel
End Sub
```

- Overrides VisitDataFrame( ) to detect the kind of frame being visited and calls a specialized subroutine to process the frame. This example creates special subroutines to extract data from each kind of frame, such as CustomerTitleFrame or ItemFrame. The example code for the VisitItemDetail subroutine is shown after the code that overrides the VisitDataFrame( ) method. The logic for the other kinds of frames is similar to the code in the VisitItemDetail subroutine.

```
' Map a generic data frame into one of types specific to
' the report. Note that we use a Case statement here,which
' is different from the way the Visitor handles AFC-provided
' classes.  A Case statement is slower at run time, but keeps
' data extraction code out of report component classes.
```

```
Sub VisitDataFrame( obj As AcDataFrame )
   Dim frameName As String
   Dim i As Integer

   frameName = GetClassName( obj )
   i = Len( frameName )
   Do While i > 1  And Mid$( frameName, i, 1 ) <> ":"
      i = i - 1
   Loop
   If i > 1 Then
      frameName = Mid$( frameName, i + 1 )
   End If
   Select Case frameName
      Case "ReportTitle1"
         VisitReportBefore( obj )
      Case "ReportTotals"
         VisitReportAfter( obj )

      Case "OfficeTitleFrame"
         VisitOfficeTitleFrame( obj )
      Case "OfficeGroupTotals"
         VisitOfficeGroupTotals( obj )

      Case "SalesRepTitleFrame"
         VisitSalesRepTitleFrame( obj )
      Case "SalesRepTotalsFrame"
         VisitSalesRepTotalsFrame( obj )

      Case "CustomerTitleFrame"
         VisitCustomerTitleFrame( obj )
      Case "CustomerGroupTotals"
         VisitCustomerGroupTotals( obj )

      Case "OrderTitleFrame"
         VisitOrderTitleFrame( obj )
      Case "OrderTotalsFrame"
         VisitOrderTotalsFrame( obj )

      Case "ItemFrame"
         VisitItemDetail( obj )
   End Select
End Sub
```

The following code sample extracts data from the content frame, ItemFrame. GetControlValue( ) accesses the data control's DataValue property. TextField and NumericField are data extraction utility functions that convert the format of each control's value before writing the value to the output text file. The subroutines, TextField and NumericField, are shown in the sample code for step 1.

```
Sub VisitItemDetail( frame As AcDataFrame )
   ItemCategory = frame.GetControlValue( "ItemCategory" )
   ItemCode = frame.GetControlValue( "ItemCode" )
   ItemDescription = frame.GetControlValue( "ItemCategory" )
   ItemQuantity = frame.GetControlValue( "IntegerControl" )
   ItemPrice = frame.GetControlValue( "IntegerControl1" )
   ItemExtension = frame.GetControlValue( "IntegerControl2" )

   ' Convert the control to the format required for the
   ' output text file
   TextField( OfficeName )
   TextField( RepName )
   TextField( CustomerContactName )
   TextField( CustomerContactPhone )
   TextField( CustomerName )
   TextField( CustomerCreditRank )
   TextField( CustomerPurchPattern )
   NumericField( OrderNumber )
   TextField( OrderForecastDate1 )
   TextField( OrderNeededDate )
   TextField( OrderForecastDate2 )
   TextField( OrderStatus )
   TextField( ItemCategory )
   TextField( ItemDescription )
   NumericField( ItemQuantity )
   NumericField( ItemPrice ) \
   NumericField( ItemExtension )

' Write converted field to output text file
   Print #Channel
   NeedComma = False
End Sub
```

- Overrides visit methods to prevent processing for the following components: sequential, parallel, and conditional sections. The example application excludes these components from being processed by overriding the visit method for the excluded component and by not calling the superclass. The following code sample shows how to prevent processing sequential sections:

  ```
  Sub VisitSequentialSection( obj As AcSequentialSection )
  End Sub
  ```

- Start the data extraction utility. The Visitor should be invoked during report generation, in the report root component's Finish( ) method, after Super::Finish( ):
  - Instantiate the report's visitor class, AcDetailCsvVisitor.
  - Set the text output file name.

- After data extraction is complete, start Microsoft Excel and display the spreadsheet, as follows:

```
Sub Finish( )
   Dim visitor As AcDetailCsvVisitor
   Super::Finish( )
   Set visitor = New AcDetailCsvVisitor
   visitor.FileName = "C:\Temp\Extract.csv"
   ApplyVisitor( visitor )
   Shell( "D:\Program Files\Microsoft Office\Office\Excel.exe
   C:\Temp\Extract.csv", 1 )
End Sub
```

Methods for Class AcVisitor

Methods defined in Class AcVisitor

VisitBaseFrame, VisitBasePage, VisitChart, VisitComponent, VisitConditionalSection, VisitContents, VisitControl, VisitCurrencyControl, VisitDataControl, VisitDataFrame, VisitDataSection, VisitDateTimeControl, VisitDoubleControl, VisitDynamicTextControl, VisitFlow, VisitFrame, VisitGroupSection, VisitImageControl, VisitIntegerControl, VisitLabelControl, VisitLeftRightPageList, VisitLeftToRightFlow, VisitLinearFlow, VisitLineControl, VisitPage, VisitPages, VisitPageList, VisitPageNumbercontrol, VisitParallelSection, VisitRectangleControl, VisitReport, VisitReportComponent, VisitReportSection, VisitSection, VisitSequentialSection, VisitSimplePageList, VisitSubpage, VisitTextControl, VisitTextualControl, VisitTitleBodyPageList, VisitTopDownFlow, VisitVisualComponent

AcVisitor::VisitBaseFrame method

Visits an AcBaseFrame component.

Syntax Sub VisitBaseFrame(obj As AcBaseFrame)

Parameter **obj**
The AcBaseFrame component to visit.

AcVisitor::VisitBasePage method

Visits an AcBasePage component.

Syntax Sub VisitBasePage(obj As AcBasePage)

Parameter **obj**
The AcBasePage component to visit.

AcVisitor::VisitChart method

Visits an AcChart component.

Syntax Sub VisitChart(obj As AcChart)

Parameter **obj**
The AcChart component to visit.

AcVisitor::VisitComponent method

Visits the components of a report.

Syntax Sub VisitComponent(obj As AcReportComponent)

Parameter **obj**
The AcReportComponent component to visit.

AcVisitor::VisitConditionalSection method

Visits the AcConditionalSection component.

Syntax Sub VisitConditionalSection(obj As AcConditionalSection)

Parameter **obj**
The AcConditionalSection component to visit.

AcVisitor::VisitContents method

Visits the contents of a report's data hierarchy components. Use VisitContents( ) to recursively traverse all the components that comprise a report's data hierarchy. VisitContents( ) uses the AcIterator class methods to traverse the data hierarchy. VisitContents( ) calls ApplyVisitor( ) for each component in the data hierarchy.

Syntax Sub VisitContents(obj As AcReportComponent)

Parameter **obj**
The AcReportComponent component to visit.

AcVisitor::VisitControl method

Visits the AcControl component.

Syntax Sub VisitControl(obj As AcControl)

Parameter **obj**
The AcControl component to visit.

AcVisitor::VisitCurrencyControl method

Visits the AcCurrencyControl component.

Syntax Sub VisitCurrencyControl(obj As AcCurrencyControl)

Parameter **obj**
The AcCurrencyControl component to visit.

AcVisitor::VisitDataControl method

Visits the AcDataControl component.

Syntax Sub VisitDataControl(obj As AcDataControl)

Parameter **obj**
The AcDataControl component to visit.

AcVisitor::VisitDataFrame method

Visits the AcDataFrame component.

Syntax Sub VisitDataFrame(obj As AcDataFrame)

Parameter **obj**
The AcDataFrame component to visit.

Example This code sample overrides the VisitDataFrame( ) method to extract the contents of three different frames in the report component: OfficeTitleFrame, OfficeGroupTotals, and OfficeFrame. GetClassName( ) returns the name of the frame being visited. Then, a specialized subroutine that knows about the contents of the frame is called to perform the extraction. The code of this subroutine is provided in the class Example:

```
' Map a generic data frame into one of the report-specific
' types handled in the class example. This code uses a Case
' statement, which is different from the way the Visitor
' handles AFC-provided classes.  The Case statement approach is
' slower at run time, but keeps data extraction code out
' of the report component classes.

Sub VisitDataFrame( obj As AcDataFrame )
   Dim frameName As String
   Dim i As Integer

   frameName = GetClassName( obj )
   i = Len( frameName )
   Do While i > 1  And Mid$( frameName, i, 1 ) <> ":"
      i = i - 1
      Loop
   If i > 1 Then
      frameName = Mid$( frameName, i + 1 )
   End If
   Select Case frameName
      Case "OfficeTitleFrame"
         VisitOfficeTitleFrame( obj )
      Case "OfficeGroupTotals"
         VisitOfficeGroupTotals( obj )
      Case "OfficeFrame"
         VisitOfficeFrame( obj )
   End Select
End Sub
```

AcVisitor::VisitDataSection method

Visits an AcDataSection component.

Syntax Sub VisitDataSection(obj As AcDataSection)

Parameter **obj**
The AcDataSection component to visit.

AcVisitor::VisitDateTimeControl method

Visits an AcDateTimeControl component.

Syntax Sub VisitDateTimeControl(obj As AcDateTimeControl)

Parameter **obj**
The AcDateTimeControl component to visit.

AcVisitor::VisitDoubleControl method

Visits an AcDoubleControl component.

Syntax Sub VisitDoubleControl(obj As AcDoubleControl)

Parameter **obj**
The AcDoubleControl component to visit.

AcVisitor::VisitDynamicTextControl method

Visits an AcDynamicTextControl component.

Syntax Sub VisitDynamicTextControl(obj As AcTextControl)

Parameter **obj**
The AcTextControl component to visit.

AcVisitor::VisitFlow method

Visits an AcFlow component.

Syntax Sub VisitFlow(obj As AcFlow)

Parameter **obj**
The AcFlow component to visit.

AcVisitor::VisitFrame method

Visits an AcFrame component.

Syntax Function VisitFrame(obj As AcFrame)

Parameter **obj**
The AcFrame component to visit.

AcVisitor::VisitGroupSection method

Visits an AcGroupSection component.

Syntax Sub VisitGroupSection(obj As AcGroupSection)

Parameter **obj**
The AcGroupSection component to visit.

AcVisitor::VisitImageControl method

Visits an AcImageControl component.

Syntax Sub VisitImageControl(obj As AcImageControl)

Parameter **obj**
The AcImageControl component to visit.

AcVisitor::VisitIntegerControl method

Visits an AcIntegerControl component.

Syntax Sub VisitIntegerControl(obj As AcIntegerControl)

Parameter **obj**
The AcIntegerControl component to visit.

AcVisitor::VisitLabelControl method

Visits an AcLabelControl component.

Syntax Sub VisitLabelControl(obj As AcLabelControl)

Parameter **obj**
The AcLabelControl component to visit.

AcVisitor::VisitLeftRightPageList method

Visits an AcLeftRightPageList component.

Syntax Sub LeftRightPageList(obj As AcLeftRightPageList)

Parameter **obj**
The AcLeftRightPageList component to visit.

AcVisitor::VisitLeftToRightFlow method

Visits an AcLeftToRightFlow component.

Syntax Sub VisitLeftToRightFlow(obj As AcLeftToRightFlow)

Parameter **obj**
The AcLeftToRightFlow component to visit.

AcVisitor::VisitLinearFlow method

Visits an AcLinearFlow component.

Syntax Sub VisitLinearFlow(obj As AcLinearFlow)

Parameter **obj**
The AcLinearFlow component to visit.

AcVisitor::VisitLineControl method

Visits an AcLineControl component.

Syntax Sub VisitLineControl(obj As AcLineControl)

Parameter **obj**
The AcLineControl component to visit.

AcVisitor::VisitPage method

Visits an AcPage component.

Syntax Sub VisitPage(obj As AcPage)

Parameter **obj**
The AcPage component to visit.

AcVisitor::VisitPages method

Visits the contents of the report's page hierarchy components.

Syntax Sub VisitPages(obj As AcReport)

Parameter **obj**
The AcReport component to visit.

AcVisitor::VisitPageList method

Recursively traverses all the components that comprise a report's page hierarchy. VisitPageList uses the AcIterator class methods to traverse the page hierarchy. VisitPageList calls ApplyVisitor for each component in the page hierarchy.

Syntax Sub VisitPageList(obj As AcPageList)

Parameter **obj**
The AcPageList component to visit.

AcVisitor::VisitPageNumberControl method

Visits an AcPageNumberControl component.

Syntax Sub VisitPageNumberControl(obj As AcPageNumberControl)

Parameter **obj**
The AcPageNumberControl component to visit.

AcVisitor::VisitParallelSection method

Visits an AcParallelSection component.

Syntax Sub VisitParallelSection(obj As AcParallelSection)

Parameter **obj**
The AcParallelSection component to visit.

AcVisitor::VisitRectangleControl method

Visits an AcRectangleControl component.

Syntax Sub VisitRectangleControl(obj As AcRectangleControl)

Parameter **obj**
The AcRectangleControl component to visit.

AcVisitor::VisitReport method

Visits the report component.

Syntax Sub VisitReport(obj As AcReport)

Parameter **obj**
The AcReport component to visit.

AcVisitor::VisitReportComponent method

Visits an AcReportComponent component.

Syntax Sub VisitReportComponent(obj As AcReportComponent)

Parameter **obj**
The AcReportComponent component to visit.

AcVisitor::VisitReportSection method

Visits an AcReportSection component.

Syntax Sub VisitReportSection(obj As AcReportSection)

Parameter **obj**
The AcReportSection component to visit.

AcVisitor::VisitSection method

Visits an AcSection component.

Syntax Sub VisitSection(obj As AcSection)

Parameter **obj**
The AcSection component to visit.

AcVisitor::VisitSequentialSection method

Visits an AcSequentialSection component.

Syntax Sub VisitSequentialSection(obj As AcSequentialSection)

Parameter **obj**
The AcSequentialSection component to visit.

AcVisitor::VisitSimplePageList method

Visits an AcSimplePageList component.

Syntax Sub VisitSimplePageList(obj As AcSimplePageList)

Parameter **obj**
The AcSimplePageList component to visit.

AcVisitor::VisitSubpage method

Visits an AcSubpage component.

Syntax Sub VisitSubpage(obj As AcSubpage)

Parameter **obj**
The AcSubpage component to visit.

AcVisitor::VisitTextControl method

Visits an AcTextControl component.

Syntax Sub VisitTextControl(obj As AcTextControl)

Parameter **obj**
The AcTextControl component to visit.

AcVisitor::VisitTextualControl method

Visits an AcTextualControl component.

Syntax Sub VisitTextualControl(obj As AcTextualControl)

Parameter **obj**
The AcTextualControl component to visit.

AcVisitor::VisitTitleBodyPageList method

Visits an AcTitleBodyPageList component.

Syntax Sub VisitTitleBodyPageList(obj As AcTitleBodyPageList)

Parameter **obj**
The AcTitleBodyPageList component to visit.

AcVisitor::VisitTopDownFlow method

Visits an AcTopDownFlow component.

Syntax Sub VisitTopDownFlow(obj As AcTopDownFlow)

Parameter **obj**
The AcTopDownFlow component to visit.

AcVisitor::VisitVisualComponent method

Visits an AcVisualComponent component.

Syntax Sub VisitVisualComponent(obj As AcVisualComponent)

Parameter **obj**
The AcVisualComponent component to visit.

Class AcVisualComponent

AcVisualComponent is the base class for all classes in which the components are visual. Data controls and static controls are the primary visual components in a report. Figure 7-108 shows the class hierarchy of AcVisualComponent.

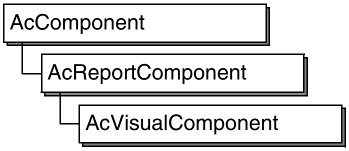


Figure 7-108 AcVisualComponent

Description AcVisualComponent defines the characteristics common to all visual components in a report. A report's visual components are:

- Frames
- Charts
- Controls
- Pages
- Flows

The primary characteristics that AcVisualComponent adds to those it inherits from AcReportComponent are visual attributes, such as position and size.

AcVisualComponent also defines a property, componentVariable, that supports access to visual components. When you assign a value to componentVariable, the framework generates a function to access the component. For example, if you set a control's componentVariable property to MyControl, the frame that contains that control provides a function called myControl( ) to access the control. The following statement is an example of how to access and modify the control using code you write for the containing frame:

```
MyControl.BackgroundColor = Teal
```

You can access the control only from its containing frame, not from another control. When you assign a value to componentVariable, if you use the name of an existing method, a compile-time error occurs.

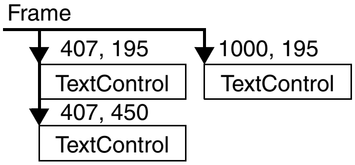
Subclassing AcVisualComponent

Do not subclass from AcVisualComponent. Instead, subclass from classes derived from AcVisualComponent, such as AcFrame, and the classes derived from AcDataControl or other concrete control classes.

Variables

Table 7-106 lists AcVisualComponent variables.

Table 7-106 AcVisualComponent variables

Variable	Type	Description
Content Offset	AcOffset	The offset to apply to the positions of all components the visual component contains during rendering. The effect of this setting on nested containers is cumulative. The default value is {0, 0}.
LinkTo	String	The value of the hyperlink, defined in the LinkExp property.
Position	AcPoint	The *x*- and *y*-coordinates, in twips, that specify the location of the component, relative to the top-left corner of its container component, as shown in Figure 7-109.  Frame 407, 195 1000, 195 TextControl TextControl 407, 450 TextControl **Figure 7-109** Positioning a visual component in its container component
Size	AcSize	The height and width of the component in twips. If the component is a circle or a line, the component's size is the size of the box that bounds it.

Properties

Table 7-107 lists AcVisualComponent properties.

Table 7-107 AcVisualComponent properties

Property	Type	Description
AnalysisType	AcAnalysis Type	Specifies how data is analyzed. The values are: ■ AnalyzeAsAutomatic. Numeric values are analyzed as measures. Non-numeric values are analyzed as dimensions. This is the default setting. ■ AnalyzeAsDimension. Numeric values are analyzed as dimensions. For example, a ZIP code can be analyzed as a dimension to enable sorting by ZIP codes. ■ AnalyzeAsMeasure. Numeric values are analyzed as measures.

(continues)

Table 7-107 AcVisualComponent properties (continued)

Property	Type	Description
CanIncrease Height	Boolean	Specifies whether the component can increase in height. The height cannot exceed 200 inches. The default value is True.
CanIncrease Width	Boolean	Specifies whether the component can increase in width. The default value is True.
CanMoveLeft	Boolean	Specifies whether the component can move left. The default value is False.
CanMoveUp	Boolean	Specifies whether the component can move up. For example, use CanMoveUp in conjunction with CanReduceHeight to suppress blank lines in addresses. The default value is False.
CanReduce Height	Boolean	Specifies whether the component can decrease in height. The default value is False.
CanReduce Width	Boolean	Specifies whether the component can decrease in width. The default value is False.
ForcePage HeightToFit	Boolean	Specifies whether the page resizes vertically to fit the height of the component.
ForcePage WidthToFit	Boolean	Specifies whether the page resizes horizontally to fit the width of the component.
Horizontal Position	Ac Horizontal Position	Specifies how the component's horizontal position is adjusted: ■ HorizontalPositionDefault. If the component's left edge is at or to the right of the horizontal midpoint of the reference component, the component is repositioned to keep the distance between its left edge and the right edge of the reference component constant. Otherwise, the component is not moved or resized. ■ HorizontalPositionFrameCenter. The component is repositioned to keep the distance between its horizontal midpoint and the horizontal midpoint of the frame constant. ■ HorizontalPositionFrameLeft. The component is not moved. ■ HorizontalPositionFrameRight. The component is repositioned to keep the distance between its right edge and the right edge of the frame constant.

Table 7-107 AcVisualComponent properties (continued)

Property	Type	Description
Horizontal Position *(continued)*	Ac Horizontal Position *(continued)*	■ HorizontalPositionLeft. If the component's left edge is to the left of the reference component's right edge, the component is not moved. Otherwise, the component is repositioned to keep the distance between its left edge and the right edge of the reference component constant. ■ HorizontalPositionRight. If the component's left edge is to the left of the reference component's left edge, the component is not moved. Otherwise, the component is repositioned to keep the distance between its left edge and the right edge of the reference component constant. Regardless of the HorizontalPosition setting, the component does not move if the HorizontalSize property is set to HorizontalSizeFrameRelative. The component does not move left if the CanMoveLeft property is set to False.
HorizontalSize	Ac Horizontal Size	Specifies how the component's horizontal size is adjusted: ■ HorizontalSizeFixed. The component is not resized. ■ HorizontalSizeFrameRelative. The component's width is adjusted to keep the distance between its right edge and the right edge of the frame constant. In this case, the component's HorizontalPosition property is ignored. ■ HorizontalSizeRelative. If the component's left edge is at or to the left of the reference component's left edge and its right edge is at or to the right of the reference component's right edge, the component's width is increased by the amount that the reference component's width has increased. If more than one dynamic content component exists, the component's width is increased in one of the following ways to give the greatest width increase: ■ The distance between the component's right edge and the right edge of the reference component remains constant. ■ The component's width is increased by the amount the reference component's width has increased. In this case, the component is also moved left if the component's CanMoveLeft property is set to True. The distance the component moves is the smallest of the following distances:

(continues)

Table 7-107 AcVisualComponent properties (continued)

Property	Type	Description
HorizontalSize *(continued)*	Ac Horizontal Size *(continued)*	❑ The component is moved left by the amount its width was increased. ❑ The distance between the component's right edge and the reference component's right edge remains constant. Regardless of the HorizontalSize setting: ■ The component does not decrease in width if CanDecreaseWidth is set to False or if MinimumWidth is greater than or equal to the component's initial width. ■ The component does not increase in width if CanIncreaseWidth is set to False or if MaximumWidth is less than or equal to the component's initial width but is not zero.
LinkExp	String	The expression defining a hyperlink for this component.
Maximum Height	AcTwips	Specifies the maximum height to which the component can grow automatically. If the component's initial height, specified by its Height property, is greater than MaximumHeight, the control does not shrink. In this case, the behavior is as if MaximumHeight is set to the initial height. Regardless of the MaximumHeight value, the component does not increase in height if CanInreaseHeight is set to False. The default value is zero, which means that the component can grow indefinitely.
Maximum Width	AcTwips	Specifies the maximum width to which the component can grow automatically. If the component's initial width, specified by its Width property, is greater than MaximumWidth, the control does not shrink. In this case, the behavior is as if MaximumWidth is set to the initial width. Regardless of the MaximumWidth setting, the component does not increase in width if CanInreaseWidth is set to False. The default value is zero, which means that the component can grow indefinitely.
Minimum Height	AcTwips	Specifies the minimum height to which the component can shrink automatically. If the component's initial height, specified by its Height property, is less than MinimumHeight, the control does not shrink. In this case, the behavior is as if MinimumHeight is set to the initial height.

Table 7-107 AcVisualComponent properties (continued)

Property	Type	Description
Minimum Height *(continued)*	AcTwips *(continued)*	Regardless of the MinimumHeight value, the component's height does not shrink if CanReduceHeight is set to False. The default value is zero, which means that the component height can shrink to zero.
Minimum Width	AcTwips	Specifies the minimum width to which the component can shrink automatically. If the component's initial width, specified by its Width property, is smaller than MinimumWidth, the control does not shrink. In this case, the behavior is as if MinimumWidth is set to the initial width. Regardless of the MinimumWidth value, the component's width does not shrink if CanReduceWidth set to False. The default value is zero, which means that the component width can shrink to zero.
Component Variable	Value	The name of an optional method in the frame that points to this component.
Position	AcPosition	The position of the component in its enclosing frame.
Searchable	AcSearch Type	The searching options for the component. Values for this property are: ■ NotSearchable. A search does not include this component. ■ SearchNoIndex. e.Report Designer Professional includes this component in a search and uses an indexed search to improve performance. ■ SearchWithIndex. e.Report Designer Professional includes this component in a search. To support selecting a component to add it to a search, set Selectable to True. The default value is SearchNoIndex.
SearchAlias	String	The name to display to the user when building a search for this component. Used only if the default value is used and SearchTag is not specified. The default value is the class name for the component.
Selectable	Boolean	True if the user can select this component.
ShowIn DHTML	Boolean	Determines whether to show the control when the report is displayed in DHTML format. False hides the control when the report displays in DHTML format.

(continues)

Table 7-107 AcVisualComponent properties (continued)

Property	Type	Description
ShowInPDF	Boolean	Determines whether to show the control when the report is displayed in PDF format. False hides the control when the report is printed in PDF format.
ShowIn Reportlet	Boolean	Determines whether to show the control when the report is displayed as a Reportlet.
ShowWhen Printing	Boolean	Sets whether to show the control when the report is printed. The default value is True.
ShowWhen Viewing	Boolean	Determines whether the user sees the control when the report appears in the report viewer. The default value is True.
Size	AcSize	The size of the visual component.
Target WindowName	String	The name of a target window in which the contents of a hyperlink appear.
Vertical Position	AcVertical Position	Specifies how the component's vertical position is adjusted: ■ VerticalPositionBottom. If the top of the component is above the top of the reference component, it is not moved. Otherwise, the component is repositioned to keep the distance between its bottom and the bottom of the reference component constant. ■ VerticalPositionDefault. If the top of the component is at or below the midpoint of the reference component, the behavior is the same as VerticalPositionBottom. Otherwise, the component is not moved. ■ VerticalPositionFrameBottom. The component is repositioned to keep the distance between its bottom and the bottom of the frame constant. ■ VerticalPositionFrameMiddle. The component is repositioned to keep the distance between its middle and the middle of the frame constant. ■ VerticalPositionFrameTop. The component is not moved. ■ VerticalPositionTop. If the top of the component is above the bottom of the reference component, it is not moved. Otherwise, the component is repositioned to keep the distance between its top and the bottom of the reference component constant. Regardless of the VerticalPosition setting: ■ The component does not move if the VerticalSize property is set to VerticalSizeFrameRelative.

Table 7-107 AcVisualComponent properties (continued)

Property	Type	Description
Vertical Position *(continued)*	AcVertical Position *(continued)*	■ The component does not move up if CanMoveUp property is set to False.
VerticalSize	AcVertical Size	Specifies how the component's vertical size is adjusted: ■ VerticalSizeFixed. The component is not resized in response to changes of its parent or content components. ■ VerticalSizeFrameRelative. The component is resized to keep the distance between its bottom and the bottom of the frame constant. ■ VerticalSizeRelative. If the top of the component is at or above the top of the reference component and its bottom is at or below the bottom of the reference component, the component's height is increased by the same amount as the reference component's height increased. If more than one dynamic content component exists, the component's height is increased in one of the following ways to give the greatest height increase: ■ The distance between the component's bottom and the bottom of the reference component remains constant. ■ The component height is increased by the same amount as the reference component's height increase. If the component's CanMoveUp property is set to True, the component is also moved up in one of the following ways to give the smallest movement: ❏ The distance between the component's bottom edge and the reference component's bottom edge remains constant. ❏ The component is moved up by the amount its height was increased. If the top of the component is below the top of the reference component or its bottom is above the bottom of the reference component, the component moves according to the value of its VerticalPosition property. Regardless of the VerticalSize setting: ■ The component does not decrease in height if CanDecreaseHeight is set to False or if MinimumHeight is greater than or equal to the component's initial height. ■ If CanIncreaseHeight is set to False or if MaximumHeight is less than or equal to the component's initial height but is not zero, the component does not increase in height.

See also Class AcControl
Class AcFrame
Class AcReportComponent

Methods for Class AcVisualComponent

Methods defined in Class AcVisualComponent

AdjustHorizontalGeometry, AdjustSize, AdjustVerticalGeometry, CanIncreaseHeight, CanIncreaseWidth, CanMoveLeft, CanMoveUp, CanReduceHeight, CanReduceWidth, CanSplitVertically, ComputeLowestSplit, FindLowestSplit, FindPageContainerByClass, GetBottom, GetFirstSlave, GetFrame, GetHeight, GetLastSlave, GetLeft, GetLinkTo, GetMaster, GetPageContainer, GetPixelSize, GetRect, GetRight, GetTop, GetVisualComponent, GetWidth, HorizontalPosition, HorizontalSize, IsFirstSlave, IsFrameDecoration, IsLastSlave, IsMaster, IsNormal, IsSlave, IsVisible, MaximumHeight, MaximumWidth, MinimumHeight, MinimumWidth, MoveBy, MoveByConstrained, MoveTo, MoveToConstrained, ResizeBy, ResizeByConstrained, ResizeTo, ResizeToConstrained, Searchable, SearchAlias, Selectable, SplitVertically, StatusText, TargetWindowName, VerticalPosition, VerticalSize

Methods inherited from Class AcReportComponent

Abandon, AddContent, Build, BuildFromRow, DetachContent, DetachFromContainer, FindContainerByClass, FindContentByClass, Finish, GenerateXML, GetComponentACL, GetConnection, GetContainer, GetContentCount, GetContentIterator, GetContents, GetDataStream, GetFirstContent, GetFirstContentFrame, GetFullACL, GetPage, GetPageIndex, GetPageList, GetReport, GetRowCount, GetSearchTag, GetTocEntry, GetVisiblePageIndex, GetXMLText, HasContents, IsContainer, IsFlow, IsFrame, IsLeaf, IsVisual, OnRow, SetSearchTag, SetTocEntry, Start

Methods inherited from Class AcComponent

ApplyVisitor, Delete, IsPersistent, New

AcVisualComponent::AdjustHorizontalGeometry method

Adjusts the width and horizontal position of the component relative to the width and horizontal position of a reference component, such as a frame.

Syntax Sub AdjustHorizontalGeometry(relativeTo As AcVisualComponent, hP As AcHorizontalPosition, hS As AcHorizontalSize)

Parameters

relativeTo
The reference component.

vP
The required horizontal positioning behavior.

vS
The required horizontal sizing behavior.

AcVisualComponent::AdjustSize method

Changes the size of the component. Override AdjustSize( ) to change the size of a component after it is built but before it is added to a page. For example, you can use AdjustSize( ) to perform the following actions:

- Expand a frame to show additional controls
- Contract a frame to hide empty controls

If the component is a control or a nested frame, you must add the component to its container's list of components to be resized using the AddToAdjustSizeList( ) method. This causes the AdjustSize( ) method to be called automatically.

Syntax Sub AdjustSize( )

Example In this example, the frame size is large enough to hold four controls. If more than four controls are needed to display the information, then AdjustSize dynamically changes the frame size based on the additional space requirements.

```
Sub AdjustSize( )
   ' Every frame can hold at least four controls. If there
   ' are more than four, widen the frame.
   If RowCount > 4 Then
      Size.Width = Size.Width + Offset * (RowCount - 4)
   End If
End Sub
```

See also AcBaseFrame::AddToAdjustSizeList method

AcVisualComponent::AdjustVerticalGeometry method

Adjusts the height and vertical position of the component relative to the height and vertical position of a reference component, such as a frame.

Syntax Sub AdjustVerticalGeometry(relativeTo As AcVisualComponent, vP As AcVerticalPosition, vS As AcVerticalSize)

Parameters

relativeTo
The reference component.

vP
The required vertical positioning behavior.

vS
The required vertical sizing behavior.

AcVisualComponent::CanIncreaseHeight method

Implements the CanIncreaseHeight property. The CanIncreaseHeight property determines whether the height of the component can increase when necessary.

Syntax Function CanIncreaseHeight( ) As Boolean

Returns True if the height of the component can increase.
False if the height of the component cannot increase.

AcVisualComponent::CanIncreaseWidth method

Implements the CanIncreaseWidth property. The CanIncreaseWidth property determines whether the width of the component can increase when necessary.

Syntax Function CanIncreaseWidth( ) As Boolean

Returns True if the width of the component can increase.
False if the width of the component cannot increase.

AcVisualComponent::CanMoveLeft method

Implements the CanMoveLeft property. The CanMoveLeft property specifies whether the component can move left when necessary.

Syntax Function CanMoveUp( ) As Boolean

Returns True if the component can move to the left.
False if the component cannot move to the left.

AcVisualComponent::CanMoveUp method

Implements the CanMoveUp property. The CanMoveUp property specifies whether the component can move up when necessary.

You can use CanMoveUp( ) in conjunction with CanReduceHeight( ) to suppress blank lines in an address.

Syntax Function CanMoveUp( ) As Boolean

Returns True if the component can move up.
False if the component cannot move up.

See also AcVisualComponent::CanReduceHeight method

AcVisualComponent::CanReduceHeight method

Implements the CanReduceHeight property. The CanReduceHeight property determines whether the height of the component can decrease when necessary.

Syntax Function CanReduceHeight( ) As Boolean

Returns True if the height of the component can decrease.
False if the height of the component cannot decrease.

AcVisualComponent::CanReduceWidth method

Implements the CanReduceWidth property. The CanReduceWidth property determines whether the width of the component can decrease when necessary.

Syntax Function CanReduceWidth( ) As Boolean

Returns True if the width of the component can decrease.
False if the width of the component cannot decrease.

AcVisualComponent::CanSplitVertically method

Determines whether the component can split across multiple pages.

Syntax Function CanSplitVertically( ) As Boolean

Returns True if the component can split vertically.
False if the component cannot split vertically.

AcVisualComponent::ComputeLowestSplit method

Determines the lowest point at which the component can be split. If the component can be split, ComputeLowestSplit( ) prepares the component to be split.

Syntax Function ComputeLowestSplit(upperLimit As AcTwips, lowerLimit As AcTwips,
splitIsNecessary As Boolean) As Boolean

Parameters **upperLimit**
The highest point at which the component can split.

lowerLimit
The lowest point at which the component can split.

splitIsNecessary
Determines whether it is necessary to split the visual component.

Returns True if the component can split.
False if the component cannot split.

AcVisualComponent::FindLowestSplit method

Establishes the lowest vertical point at which the component can split. You must implement FindLowestSplit( ) in all components that can split vertically. You must call FindLowestSplit( ) before calling SplitVertically( ).

Syntax Function FindLowestSplit(upperLimit As AcTwips, lowerLimit As ActTwips, splitIsNecessary As Boolean, fragment1Bottom As AcTwips, fragment2Top As AcTwips) As Boolean

Parameters **upperLimit**
The highest point at which the component can split.

lowerLimit
The lowest point at which the component can split.

splitIsNecessary
Determines if the visual component must be split if possible.

fragment1Bottom
Set to the position of the bottom of the first fragment after the component is split, relative the internal coordinate space of the component’s container.

fragment2Top
Set to the position of the top of the second fragment after the component is split, relative the internal coordinate space of the component’s container.

Returns True if the component can split.
False if the component cannot split.

See also AcVisualComponent::SplitVertically method

AcVisualComponent::FindPageContainerByClass method

Returns a reference to the named container component in the page hierarchy. Use FindPageContainerByClass( ) to search up the page hierarchy for the container component with the named class. The class can be an AFC or user-defined class. The search starts with the component initiating the search. If you search for the class corresponding to the component initiating the search, the return value is a reference to this component. If you start the search on a higher level component, use the GetPageContainer( ) method to position to the right level in the page hierarchy.

Syntax Function FindPageContainerByClass(className As String) As AcReportComponent

Returns A reference to the container component in the page hierarchy with the named class.
Nothing if the container component cannot be found.

Example In this example, the report design calls FindPageContainerByClass( ) to return a handle to the page list component before updating each page in the report with relative page numbers:

```
Function GetValue( ) As Variant
   Dim pageListAs AcPageList
   Set pageList = FindPageContainerByClass( "AcPageList" )
   If pageList Is Nothing Then
      GetValue = Null
   Else
      GetValue = pageList.GetPageCount( )
   End If
End Function
```

See also AcReportComponent::FindContainerByClass method

AcVisualComponent::GetBottom method

Returns the position of the bottom of the component, in twips, relative to the top of its container frame.

Syntax Function GetBottom( ) As Integer

AcVisualComponent::GetFirstSlave method

Returns the handle to the component's first slave component. A master component is a component that has been split across multiple pages. A slave component is a fragment that results from the split.

Syntax Function GetFirstSlave( ) As AcVisualComponent

Returns If the component is a master component, returns the handle to the component's first slave visual component.
If the component is not a master component, returns Nothing.

AcVisualComponent::GetFrame method

Returns a reference to the frame containing the visual component. A visual component is typically contained in a frame. In a derived class, you can call GetFrame( ) to find out which frame contains the current component.

Syntax Function GetFrame( ) As AcFrame

Returns A reference to the frame if the current component is a control.
A reference to itself if the current component is a frame.

AcVisualComponent::GetHeight method

Returns the height of the component in twips.

Syntax Function GetHeight( ) As Integer

AcVisualComponent::GetLastSlave method

Returns the handle to the component's last slave component. A master component is a component that splits across multiple pages. A slave component is a fragment that results from the split.

Syntax Function GetLastSlave( ) As AcVisualComponent

Returns If the component is a master component, returns the handle to the component's first slave visual component.
If the component is not a master component, returns Nothing.

AcVisualComponent::GetLeft method

Returns the position of the left edge of the component, in twips, relative to the left edge of the container frame.

Syntax Function GetLeft( ) As Integer

AcVisualComponent::GetLinkTo method

Returns the value of the hyperlink expression contained in the LinkTo variable. The LinkExp property generates the LinkTo variable's value.

Syntax Function GetLinkTo( ) As String

AcVisualComponent::GetMaster method

Returns a handle to the component's master component. A master component is a component that splits across multiple pages. A slave component is a fragment that results from the split.

Syntax Function GetMaster( ) As AcVisualComponent

Returns If the component is a slave component, returns the handle to the component's master component.
If the component is not a slave component, returns Nothing.

AcVisualComponent::GetPageContainer method

Returns a reference to the container component in the page hierarchy for this component.

At report generation time, the Factory builds two component hierarchies: the structure hierarchy and the page hierarchy. When report generation begins, components in the report design are stored in the structure hierarchy and the page hierarchy is empty. As frames are built, the Factory places the visual components in the page hierarchy. The frame's container component in the page hierarchy is different from its container component in the structure hierarchy. For example, in the structure hierarchy, a frame's component container may be another frame, section, or report component. When the Factory builds the frame, the flow component in the page hierarchy contains the frame.

Syntax Function GetPageContainer( ) As AcVisualComponent

Returns A reference to the container component in the page hierarchy.
Nothing if the frame has not yet been assigned to a page by the Factory.

See also AcReportComponent::GetContainer method

AcVisualComponent::GetPixelSize method

Gets the size of the component in pixels.

Syntax Function GetPixelSize(twipsPerPixel As Integer) As AcSize

Parameter **twipsPerPixel**
The number of twips per pixel.

AcVisualComponent::GetRect method

Returns right, left, top, and bottom coordinates of the component, in twips, relative to its container frame. GetRect( ) uses the GetTop( ), GetBottom( ), GetLeft( ), and GetRight( ) methods to calculate the coordinates.

Syntax Function GetRect( ) As AcRectangle

Returns The coordinates in a structure that has the data type AcRectangle.

See also Class AcRectangleControl

AcVisualComponent::GetRight method

Returns the position of the right edge of the component, in twips, relative to the left edge of its container frame.

Syntax Function GetRight( ) As Integer

AcVisualComponent::GetTop method

Returns the position of the top of the component, in twips, relative to the top of its container frame.

Syntax Function GetTop( ) As Integer

AcVisualComponent::GetVisualComponent method

Returns a reference to the current visual component.

Syntax Function GetVisualComponent( ) As AcVisualComponent

AcVisualComponent::GetWidth method

Returns the width of the component in twips.

Syntax Function GetWidth( ) As Integer

AcVisualComponent::HorizontalPosition method

Implements the HorizontalPosition property. The HorizontalPosition property determines how to position the component horizontally.

Syntax Function HorizontalPosition( ) As AcHorizontalPosition

AcVisualComponent::HorizontalSize method

Implements the HorizontalSize property. The HorizontalSize property determines how to adjust the component's horizontal size.

Syntax Function HorizontalSize( ) As AcHorizontalPosition

AcVisualComponent::IsFirstSlave method

Determines whether the component is the first slave of the master component. A master component is a component that splits across multiple pages. A slave component is a fragment that results from the split.

Syntax Function IsFirstSlave( ) As Boolean

Returns True if the component is the first slave of its master component.
False if the component is not the first slave of its master component.

AcVisualComponent::IsFrameDecoration method

Determines whether the component is a frame decoration. Frame decoration components include controls, such as a page number or date, that appear on a page. For more information about IsFrameDecoration, see *Developing Reports using e.Report Designer Professional.*

Syntax Function IsFrameDecoration( ) As Boolean

AcVisualComponent::IsLastSlave method

Determines whether the component is the last slave of the master component. A master component is a component that splits across multiple pages. The last slave component is the last fragment that results from the split.

Syntax Function IsLastSlave( ) As Boolean

Returns True if the component is the last slave of its master component.

False if the component is not a slave or is not the last slave of its master component.

AcVisualComponent::IsMaster method

Determines whether the component is a master component. A master component is a component that splits across multiple pages. A master component can have slave components, or fragments, that result from the split.

Syntax Function IsMaster( ) As Boolean

Returns True if the component is a master component.

False if the component is not a master component.

AcVisualComponent::IsNormal method

Determines that the component is neither a master component nor a slave component. A component is normal if it does not split across multiple pages.

Syntax Function IsNormal( ) As Boolean

Returns True if the component does not split across multiple pages.

False if the component is either a master or a slave component.

AcVisualComponent::IsSlave method

Determines whether the component is a slave component. A master component is a component that splits across multiple pages. A master component can have slave components, or fragments, that result from the split.

Syntax Function IsSlave( ) As Boolean

Returns True if the component is a slave component.

False if the component is not a slave component.

AcVisualComponent::IsVisible method

Determines whether the component is completely or partially visible to the user. For example, if you set the TocIfAnyVisible property on a component to True, then IsVisible returns True for the component.

Syntax Function IsVisible( ) As Boolean

Returns True if the component is visible.
False if the component is not visible.

AcVisualComponent::MaximumHeight method

Implements the MaximumHeight property. The MaximumHeight property specifies the maximum height to which the component can grow automatically.

Syntax Function MaximumHeight( ) As AcTwips

AcVisualComponent::MaximumWidth method

Implements the MaximumWidth property. The MaximumWidth property specifies the maximum width to which the component can increase when necessary.

Syntax Function MaximumWidth( ) As AcTwips

AcVisualComponent::MinimumHeight method

Implements the MinimumHeight property. The MinimumHeight property specifies the minimum height to which the component can shrink when necessary.

Syntax Function MinimumHeight( ) As AcTwips

AcVisualComponent::MinimumWidth method

Implements the MinimumWidth property. The MinimumWidth property specifies the minimum width to which the component can shrink when necessary.

Syntax Function MinimumWidth( ) As AcTwips

AcVisualComponent::MoveBy method

Moves a control or nested frame within its container frame or flow by the given distances. The distances can be positive or negative.

Syntax Sub MoveBy(deltaX As Integer, deltaY As Integer)

Parameters **deltaX**
The horizontal distance, in twips, to move the component. The distance is relative to the current position.

deltaY
The vertical distance, in twips, to move the component. The distance is relative to the current position.

See also AcVisualComponent::MoveTo method

AcVisualComponent::MoveByConstrained method

Specifies the horizontal and vertical distances by which to move the component. This method also uses the value of the component's CanMoveUp property to determine the amount by which to move the component.

Syntax Sub MoveByConstrained(deltaX As Integer, deltaY As Integer)

Parameters **deltaX**
The horizontal distance, in twips, to move the component. The distance is relative to the component's current position.

deltaY
The vertical distance, in twips, to move the component. The distance is relative to the component's current position.

See also AcVisualComponent::MoveToConstrained method

AcVisualComponent::MoveTo method

Changes the position of a control or nested frame within its container frame or flow.

Syntax Sub MoveTo(newX As Integer, newY As Integer)

Parameters **newX**
The new *x*-position of the control, in twips, relative to the left edge of the enclosing frame.

newY
The new *y*-position of the control, in twips, relative to the top edge of the enclosing frame.

See also AcVisualComponent::MoveBy method

AcVisualComponent::MoveToConstrained method

Moves the component within the container frame. This method also uses the value of the component's CanMoveUp property to determine the amount by which to move the component.

Syntax Sub MoveToConstrained(newX As Integer, newY As Integer)

Parameters **newX**
The new *x*-position of the component, in twips, relative to the left edge of the container frame.

newY
The new *y*-position of the control, in twips, relative to the top edge of the container frame.

See also AcVisualComponent::MoveByConstrained method

AcVisualComponent::ResizeBy method

Resizes a component by the given distances. Negative amounts make the component smaller. Positive amounts make the component larger.

Syntax Sub ResizeBy(deltaWidth As Integer, deltaHeight As Integer)

Parameters **deltaWidth**
The amount, in twips, by which to resize the width of the component.

deltaHeight
The amount, in twips, by which to resize the height of the component.

See also AcVisualComponent::ResizeTo method
AcVisualComponent::ResizeByConstrained method

AcVisualComponent::ResizeByConstrained method

Specifies the amount by which to resize a component. This method also uses the values of the component's CanIncreaseHeight, CanReduceHeight, MinimumHeight, and MaximumHeight properties to determine the amount by which the component is resized. Negative amounts make the component smaller. Positive amounts make the component larger.

Syntax Sub ResizeByConstrained(deltaX As Integer, deltaY As Integer)

Parameters **deltaX**
The amount, in twips, by which to resize the width of the component.

deltaY
The amount, in twips, by which to resize the height of the component.

See also AcVisualComponent::ResizeToConstrained method

AcVisualComponent::ResizeTo method

Resizes a frame or control to the specified size.

Syntax Sub ResizeTo(newWidth As Integer, newHeight As Integer)

Parameters **newWidth**
The new width of the component in twips.

newHeight
The new height of the component in twips.

See also AcVisualComponent::ResizeBy method

AcVisualComponent::ResizeToConstrained method

Resizes the component to the given size. This method also uses the values of the component's CanIncreaseHeight, CanReduceHeight, MinimumHeight, and MaximumHeight properties to determine the amount by which the component is resized.

Syntax Sub ResizeToConstrained(newWidth As Integer, newHeight As Integer)

Parameters **newWidth**
The new width of the component in twips.

newHeight
The new height of the component in twips.

See also AcVisualComponent::MoveToConstrained method

AcVisualComponent::Searchable method

Implements the Searchable property. This method specifies whether and how a user can search for a component. You can disable searching, enable searching, or enable high performance searching using an indexed search.

Syntax Function Searchable( ) As AcSearchType

See also AcVisualComponent::SearchAlias method

AcVisualComponent::SearchAlias method

Implements the SearchAlias property. The SearchAlias property specifies the name to display in the Search dialog when a user creates a search criteria for a component. The default value is the component's class name.

Syntax Function SearchAlias( ) As String

See also AcVisualComponent::Searchable method

AcVisualComponent::Selectable method

Implements the Selectable property. The Selectable property specifies whether a user can select the visual component in the report viewer.

Syntax Function Selectable( ) As Boolean

Returns True if the component is selectable.
False if the component is not selectable.

See also AcVisualComponent::Searchable method

AcVisualComponent::SplitVertically method

Overridden by AFC classes to split visual components vertically. SplitVertically( ) throws a ClassProtocolError. It is overridden in other AFC classes to split the component vertically so that it can be spread across multiple flows.

Before calling SplitVertically( ) you must successfully call the component's FindLowestSplit( ) or ComputeLowestSplit( ) method.

When overridden, SplitVertically( ) sets the values of the arguments to two slave components. A slave component is a fragment that is produced after a master component has been split. The first slave component is placed in the current flow and the second slave component is placed in a subsequent flow.

If the original component is neither a master nor a slave component, it must first be converted to a master component. The two returned components become its slaves.

If the original component is a master component, SplitVertically( ) must return a ClassProtocolError. If the original component is a slave component, SplitVertically( ) must return that component in either the first or the second argument.

Syntax

```
Sub SplitVertically( fragment1 As AcVisualComponent, fragment2 As
    AcVisualComponent )
```

Parameters

fragment1
The fragment to fit into the current flow.

fragment2
The fragment to fit into the subsequent flow.

AcVisualComponent::StatusText method

Returns the value of GetLinkTo( ) if there is a hyperlink or the help text associated with this component.

Syntax

```
Function StatusText( ) as String
```

Returns

The value of the hypertext link if one exists.
Help text if any exists.
An empty string if neither a hypertext link nor help text exist.

AcVisualComponent::TargetWindowName method

Implements the TargetWindowName property. Override TargetWindowName( ) to specify a target window in which to display the new report. When you hyperlink from the current report to a new report, you can display the new report in the same window or in a different window. To display the new report in a different window, set TargetWindowName to the name of the window. If the target window name is blank, the new report displays in the current window.

Syntax Function TargetWindowName( ) As String

Returns The name of the target window.
Blank to display the report in the current window.

AcVisualComponent::VerticalPosition method

Implements the VerticalPosition property. The VerticalPosition property specifies how to adjust the component's vertical position.

Syntax Function VerticalPosition( ) As AcVerticalPosition

AcVisualComponent::VerticalSize method

Implements the VerticalSize property. The VerticalSize property specifies how to adjust the component's vertical size.

Syntax Function VerticalSize( ) As AcVerticalSize

Index