

15

Finding locations

Jonathan Bailey

A feature is an object that has geometry. In most cases, this geometry is captured by digitizing or scanning paper maps. Sometimes, though, geographic data exists that indirectly captures geometry by describing locations, such as addresses, city names, or telephone numbers.

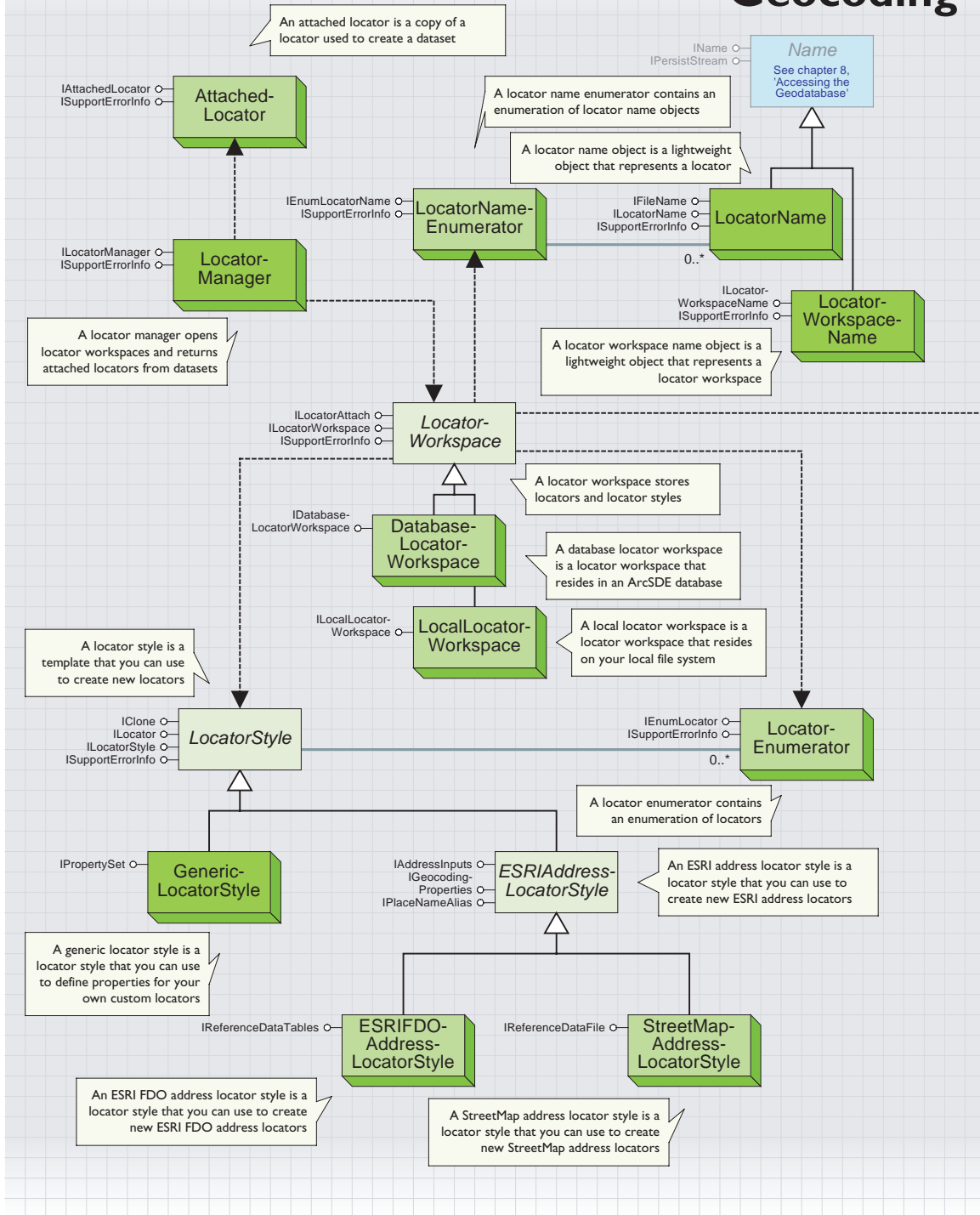
A locator is an object that can create geometric descriptions of locations (for example, points) from nonspatial descriptions of locations. Locators, which are stored in locator workspaces, use reference data that has both geometry and attributes in order to find locations.

ESRI® ArcGIS™ provides several locator styles for creating address locators. An address locator defines the location of reference data, rules and algorithms, and parameters for geocoding addresses.

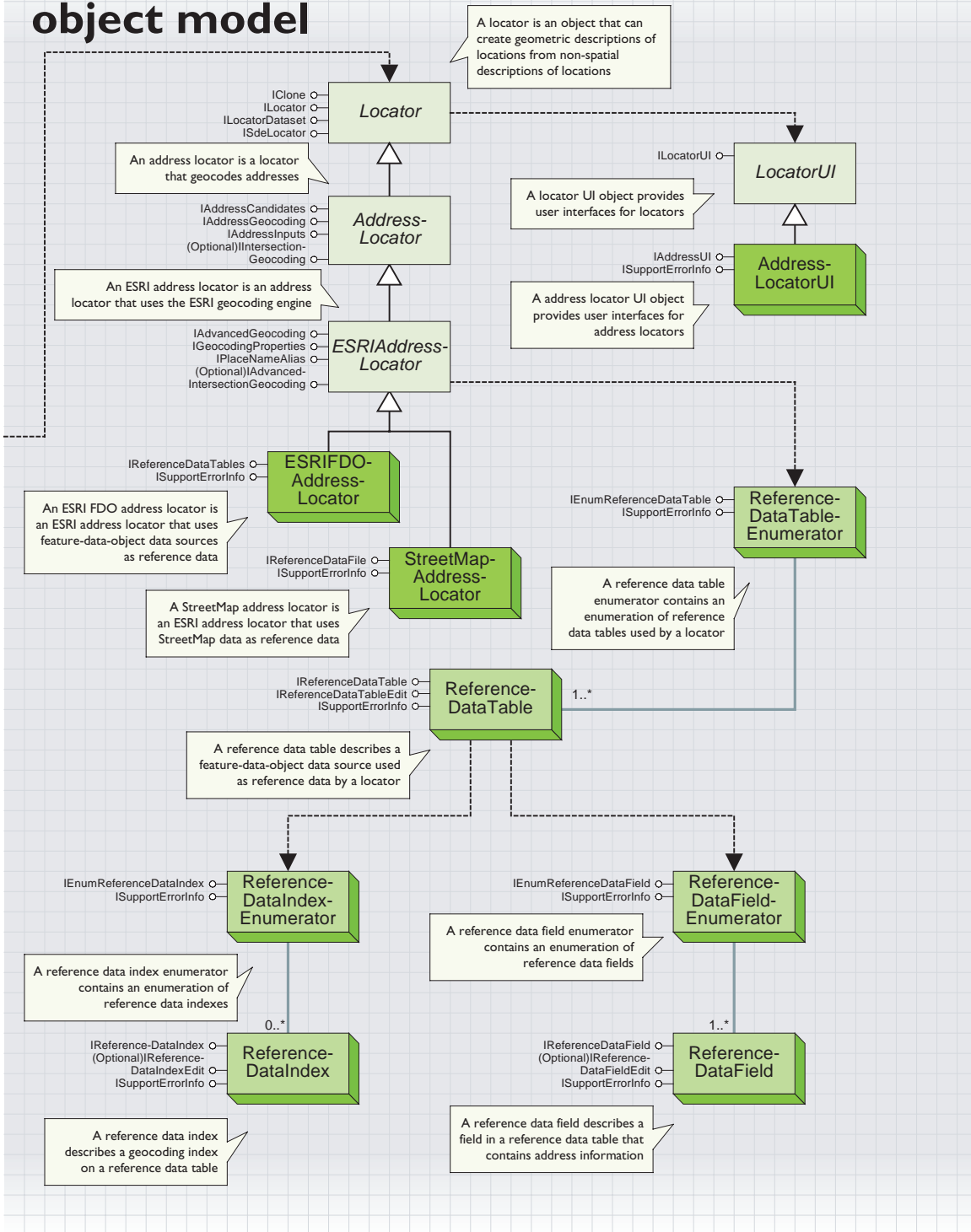
The locator framework provides a framework for creating custom locators. Developers can extend this framework to create their own types of locators. Custom locators might create geometry for localized or custom descriptions of locations, use specialized algorithms for matching location descriptions to reference data, use custom rules for standardizing and matching location descriptions, or use different types of reference data.



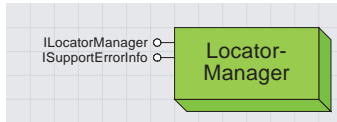
Geocoding



object model



LOCATORMANAGER COCLASS



The *LocatorManager* coclass opens *LocatorWorkspaces* and provides access to *Locators* attached to *Datasets*.

The *LocatorManager* coclass opens *LocatorWorkspaces* and provides access to *Locators* attached to *Datasets*.

ILocatorManager : IUnknown	Provides access to members for manipulating located datasets and locator workspaces.
← GetLocatorFromDataset (in Dataset: IDataset) : IAttachedLocator	Returns the locator attached to the dataset.
← GetLocatorWorkspace (in wks: IWorkspace) : ILocatorWorkspace	Returns the locator workspace for a workspace.
← GetLocatorWorkspaceFromPath (in Path: String) : ILocatorWorkspace	Returns the locator workspace for a path.
← HasLocatorAttached (in DatasetName: IDatasetName) : Boolean	Indicates if a dataset has an attached locator.

Use the *ILocatorManager* interface to open locator workspaces and to retrieve locators attached to datasets. Use the *GetLocatorWorkspace* and *GetLocatorWorkspacePath* methods to open locator workspaces. The first method opens a locator workspace contained by the given *Workspace* object; the second opens a locator workspace on your local file system in the path that you specify.

Datasets can have locators attached to them. In general, if a dataset has an attached locator, then the dataset was created using that locator. For example, when you create a feature class using a geocoding service in ArcMap™ or ArcCatalog™, a copy of the locator that you use to create the feature class is attached to the feature class. ArcMap and ArcCatalog then use this locator when you rematch the geocoded feature class. For more information on attaching locators to datasets, see the *AddressLocator* abstract class and *LocatorWorkspace* abstract class topics in this chapter.

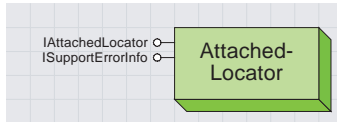
To retrieve a locator that was used to create a dataset, use the *HasAttachedLocator* method to determine whether or not the dataset was created using a locator. Then, use the *GetLocatorFromDataset* method to retrieve the locator that was used to create the dataset.

The following Visual Basic® for Applications (VBA) code can be used to determine if a dataset has an attached locator and, if so, to retrieve that locator from the dataset. This code assumes that you already have a reference to an object that supports the *IDatasetName* interface.

```
Function GetAttachedLocator(pDatasetName As IDatasetName) _
    As ILocator
    Dim pLocatorManager As ILocatorManager
    Dim pName As IName
    Dim pDataset As IDataset

    Set pLocatorManager = New LocatorManager
    If pLocatorManager.HasLocatorAttached(pDatasetName) Then
        Set pName = pDatasetName
        Set pDataset = pName.Open
        Set GetAttachedLocator = _
            pLocatorManager.GetLocatorFromDataset(pDataset)
    End If
End Function
```

ATTACHEDLOCATOR CLASS



An `AttachedLocator` provides information about a `Locator` that is attached to a `Dataset`. In general, it is a copy of the locator that was used to create the dataset and provides all of the information necessary to re-create the locating process.

The `AttachedLocator` class provides information about a `Locator` that is attached to a `Dataset`. In general, when you create a dataset using a locator, a copy of the locator that was used to create the dataset is attached to the dataset. You can use the `ILocatorManager::GetLocatorFromDataset` method to retrieve an attached locator from a dataset.

IAttachedLocator : IUnknown	Provides access to members that describe the process used to create the feature class.
InputFieldNamesList: String	Names of fields in the input table used by the locator.
InputJoinFieldName: String	Name of the ObjectID field in the input table.
InputTable: ITable	Table that was located.
Locator: ILocator	Locator used to create the feature class.
OutputFieldNamesList: String	Names of result fields in the output feature class.
OutputJoinFieldName: String	Name of the JoinOID field in the output feature class.
OutputTable: ITable	Feature class that was created.

Use the `IAttachedLocator` interface to retrieve information about the attached locator. This interface provides all the information necessary to re-create the locating process that was used to create the dataset. The `Locator` property returns a reference to the locator represented by the attached locator. This locator is a copy of the locator that was used to create the dataset and retains all of the locator settings that were used to create the dataset.

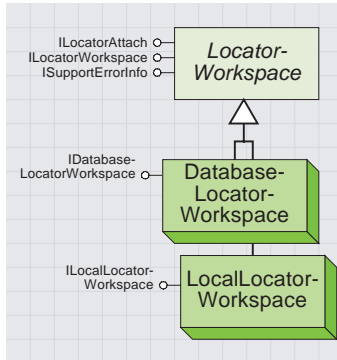
The `InputTable` property returns a reference to the `Table` that contains the address information, and the `OutputTable` property returns a reference to the `FeatureClass` that contains the located `Features`. Normally, both of these properties refer to the located dataset. The `InputJoinFieldName` and `OutputJoinFieldName` properties specify the names of the fields in the `InputTable` and the `OutputTable` that can be used to relate records between them. Normally, both of these properties return the name of the `FeatureID` field in the located dataset.

The `InputFieldNamesList` property returns a comma-delimited string that contains the names of the fields in `InputTable` that contain address information. The address fields in this string appear in the same order as specified by the `IAddressInputs::AddressFields` property of the locator. If you create a static feature class using a locator in ArcMap or ArcCatalog, then the located feature class contains two copies of the address fields. One copy maintains the attribute values that were present in the address table when it was originally located. The other copy is a set of fields that contain the address information that the locator uses to rematch addresses in the located feature class. The names of the fields in this second copy are prefixed with "ARC_" if you located the feature class using ArcMap or ArcCatalog. You can edit the values in these fields and then rematch the located feature class in order to update the located features. If you create a dynamic feature class, then the located feature class only contains the second copy of the address fields. The original copy remains in the address table.

ATTACHEDLOCATOR CLASS

The *OutputFieldNamesList* property returns a comma-delimited string that contains the names of the fields in the located feature class that contain the match information generated by the locator. The match fields in this string appear in the same order as specified by the *AddressGeocoding::MatchFields* property of the locator.

LOCATOR WORKSPACE CLASSES



The *LocatorWorkspace* provides access to the *Locators* and *LocatorStyles* in an *ArcSDE* database or in the local file system.

A *DatabaseLocatorWorkspace* is a *LocatorWorkspace* stored in an *ArcSDE* database.

A *LocalLocatorWorkspace* contains the *Locators* and *LocatorStyles* stored in a folder in the local file system.

The *LocatorWorkspace* abstract class provides access to the *Locators* and *LocatorStyles* in an *ArcSDE™* database or in a folder in the local file system. You can use the *LocatorManager* coclass to open a locator workspace.

ILocatorWorkspace : IUnknown	Provides access to members for managing the locators in the locator workspace.
<ul style="list-style-type: none"> ■ LocatorNames (in queryType: <i>esriLocatorQuery</i>, in Category: <i>String</i>) : <i>IEnumLocatorName</i> 	<i>LocatorName</i> objects in the locator workspace.
<ul style="list-style-type: none"> ■ Locators (in queryType: <i>esriLocatorQuery</i>, in Category: <i>String</i>) : <i>IEnumLocator</i> 	<i>Locators</i> in the locator workspace.
<ul style="list-style-type: none"> ■ Name: <i>ILocatorWorkspaceName</i> 	The <i>Name</i> object for the locator workspace.
<ul style="list-style-type: none"> ← AddLocator (in Name: <i>String</i>, in Locator: <i>ILocator</i>, in ConfigKeyword: <i>String</i>, CancelTracker: <i>ITrackCancel</i>) : <i>ILocator</i> 	Adds a locator.
<ul style="list-style-type: none"> ← AddLocatorStyle (in Name: <i>String</i>, in Category: <i>String</i>, in locatorStyle: <i>ILocatorStyle</i>) 	Adds a locator style.
<ul style="list-style-type: none"> ← CopyLocator (in srcName: <i>String</i>, in dstName: <i>String</i>) 	Copies a locator.
<ul style="list-style-type: none"> ← DeleteLocator (in Name: <i>String</i>) 	Deletes a locator.
<ul style="list-style-type: none"> ← GetLocator (in Name: <i>String</i>) : <i>ILocator</i> 	Gets a locator.
<ul style="list-style-type: none"> ← GetLocatorName (in Name: <i>String</i>) : <i>ILocatorName</i> 	Gets a <i>LocatorName</i> object.
<ul style="list-style-type: none"> ← GetLocatorStyle (in Name: <i>String</i>) : <i>ILocatorStyle</i> 	Gets a locator style.
<ul style="list-style-type: none"> ← RenameLocator (in oldName: <i>String</i>, in newName: <i>String</i>) 	Renames a locator.
<ul style="list-style-type: none"> ← UpdateLocator (in Locator: <i>ILocator</i>) 	Modifies the properties of a locator.

Use the *ILocatorWorkspace* interface to manage the set of locators and locator styles in the locator workspace. Using this interface, you can create, delete, and modify locators and locator styles.

For some of the members of this interface, you must specify a *Category* string parameter. You can use categories to organize your locators and locator styles. All of the locators and locator styles provided with *ArcGIS* have a category of “Address”.

To retrieve a subset of locators from the locator workspace, use the *Locators* property. You can restrict the types of locators that you want to retrieve from the locator workspace by type (locators, locator styles, or both) and by category. To retrieve locators or locator styles from all categories, specify a blank string for the *Category* parameter. You can also retrieve a set of *LocatorNames* from the locator workspace using the *LocatorNames* property. *LocatorName* objects are lightweight *Name* objects that you can use to get basic information about a locator. To get a *LocatorWorkspaceName* object for the locator workspace, use the *Name* property.

To retrieve an individual locator from the locator workspace, use the *GetLocator* method. Likewise, to retrieve an individual locator style, use the *GetLocatorStyle* method. You can also get *LocatorName* objects for individual locators or locator styles using the *GetLocatorName* method.

To add a locator to a locator workspace, use the *AddLocator* method. In general, when creating a locator, you will first retrieve a locator style from the locator workspace, set the properties on the locator style, then use this method to create a new locator. When you add a locator to an

ArcSDE locator workspace, you can specify a configuration keyword for the index tables that are created by the locator using the *ConfigKeyword* parameter. Using a configuration keyword specifies certain parameters for storing tables to the ArcSDE database, and the proper configuration can improve the performance of your locators. For more information on ArcSDE configuration keywords, refer to the *ArcSDE Configuration and Tuning Guide* for your relational database management system (RDBMS).

The following VBA code demonstrates how to create a new locator. This code assumes that you already have a reference to a locator workspace and to a feature class that the locator will use as reference data.

```
Sub CreateNewLocator(pLocatorWorkspace As ILocatorWorkspace, _
    pFeatureClassName As IName)
    Dim pLocatorStyle As ILocatorStyle
    Dim pLocator As ILocator
    Dim pReferenceDataTables As IReferenceDataTables
    Dim pEnumReferenceDataTable As IEnumReferenceDataTable
    Dim pReferenceDataTableEdit As IReferenceDataTableEdit

    Set pLocatorStyle = _
        pLocatorWorkspace.GetLocatorStyle("US Streets")
    Set pLocator = pLocatorStyle
    pLocator.Description = "Description of the geocoding service"
    Set pReferenceDataTables = pLocatorStyle
    Set pEnumReferenceDataTable = pReferenceDataTables.Tables
    With pEnumReferenceDataTable
        .Reset
        Set pReferenceDataTableEdit = .Next
    End With
    Set pReferenceDataTableEdit.Name = pFeatureClassName
    Set pLocator = pLocatorWorkspace.AddLocator _
        ("New US Streets Geocoding Service", pLocator, "", Nothing)
End Sub
```

To add a locator style to the locator workspace, use the *AddLocatorStyle* method. In general, you will only use this method if you have created your own custom locator style and want to add it to the locator workspace.

You can delete and rename locators and locator styles in the locator workspace using the *DeleteLocator* and *RenameLocator* methods. To change the settings for an existing locator, use the *UpdateLocator* method.

The following VBA code demonstrates how to change the settings for an existing locator. This code assumes that you already have a reference to the locator workspace that contains the locator that you want to modify.

```
Sub UpdateLocator(pLocatorWorkspace As ILocatorWorkspace)
```

LOCATOR WORKSPACE CLASSES

```

Dim pLocator As ILocator
Set pLocator = _
    pLocatorWorkspace.GetLocator("My Geocoding Service")
pLocator.Description = "New geocoding service description."
pLocatorWorkspace.UpdateLocator (pLocator)
End Sub

```

Enumeration <i>esriLocatorQuery</i>	Locator query types.
0 - <i>esriLocatorStyle</i>	Query for locator styles.
1 - <i>esriLocator</i>	Query for locators.
2 - <i>esriAllTypes</i>	Query for locators and locator styles.

The *esriLocatorQuery* enumeration lists the types of queries you can perform on a locator workspace. Use this enumeration to specify whether you want a member of the *ILocatorWorkspace* interface to return locators, locator styles, or both.

ILocatorAttach : IUnknown	Provides access to members that attach locators to datasets.
← AttachLocator (in Locator: ILocator, in attachToTable: ITable, in InputTable: ITable, in inputFieldNames: String, in InputJoinFieldName: String, in OutputTable: ITable, in outputFieldNames: String, in OutputJoinFieldName: String)	Attaches a locator to a table.

When you create a *FeatureClass* using a locator, you can attach a copy of the locator to the feature class. Doing so allows you to rematch the feature class that was created with the locator at a later time. You can attach a locator to a feature class using the *ILocatorAttach* interface.

The *ILocatorAttach* interface has only one method, *AttachLocator*. Refer to the *AttachedLocator* class and *AddressLocator* abstract class topics in this chapter for more information on how to attach and retrieve attached locators.

Database locator workspaces are locator workspaces that reside inside ArcSDE databases. These objects contain ArcSDE locators and locator styles. Each ArcSDE database has a database locator workspace, which is represented in ArcCatalog as the Geocoding Services folder inside an ArcSDE database connection.

IDatabaseLocatorWorkspace : IUnknown	Provides access to members for inspecting an ArcSDE locator workspace.
■ Workspace: IWorkspace	The ArcSDE workspace that contains the locator workspace.

DatabaseLocatorWorkspaces support the *IDatabaseLocatorWorkspace* interface. The *Workspace* property on this interface returns a reference to the ArcSDE *Workspace* that contains the locator workspace.

LocalLocatorWorkspaces are locator workspaces that reside on the local file system. These objects contain client-side locators and locator styles. Any folder in the local file system can contain a locator workspace. The default local locator workspace resides in your personal profile folder and

LOCATOR WORKSPACE CLASSES

is represented by the top-level Geocoding Services folder in ArcCatalog. This locator workspace is the default location for all of the locators that you create using ArcCatalog and contains all of the locator styles provided with ArcGIS. The folder is in one of the following locations, depending on your operating system.

For Windows NT:

C:\WINNT\Profiles\\Application Data\ESRI\Locators\

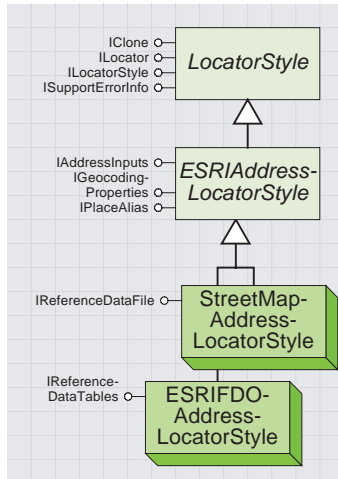
For Windows 2000:

C:\Documents and Settings\\Application Data\ESRI\Locators\

ILocalLocatorWorkspace : IUnknown	Provides access to members for inspecting a local locator workspace.
■ Path: String	The path to the folder that contains the locator workspace.

Local locator workspaces support the *ILocalLocatorWorkspace* interface. The *Path* property in this interface returns the path to the folder that contains the locator workspace.

LOCATOR STYLE CLASSES



A *LocatorStyle* is a template that you can use to create new Locators.

An *ESRIAddressLocatorStyle* is a template for creating new Locators that use the ESRI geocoding engine.

ESRIAddressLocatorStyle can be used to create new Locators that use feature data object data sources as reference data.

StreetMapAddressLocatorStyle can be used to create new Locators that use *StreetMap*[™] data as reference data.

A *LocatorStyle* is an object that you can use as a template to create a new *Locator*. See the *LocatorWorkspace* abstract class topic in this chapter to learn how to create a new locator using a locator style.

ILocatorStyle : IUnknown	<i>Indicator interface that identifies a locator style.</i>

The *ILocatorStyle* interface is an empty interface that distinguishes locator styles from locators. You can use the *ILocator* interface to specify general properties for the locator. For more information on using this interface, see the *Locator* abstract class topic in this chapter.

The *ESRIAddressLocatorStyle* abstract class is a *LocatorStyle* that you can use to create a *Locator* that uses ESRI's geocoding engine.

IGeocodingProperties : IAddressInputs	Provides access to members that control geocoding settings.
<ul style="list-style-type: none"> ■ AddPercentAlongToMatchFields: Boolean ■ AddReferenceIDToMatchFields: Boolean ■ AddStandardizeStringToMatchFields: Boolean ■ AddXYCoordsToMatchFields: Boolean → DefaultInputFieldNames (in addressField: String) : Variant ■ EndOffset: Long ■ IntersectionConnectors: String ■ MatchIfScoresTie: Boolean ■ MinimumCandidateScore: Long ■ MinimumMatchScore: Long ■ SideOffset: Double ■ SideOffsetUnits: esriUnits ■ SpellingSensitivity: Long ■ UseRelativePaths: Boolean 	<ul style="list-style-type: none"> <i>Indicates if the percentage along the reference feature at which the address is located is included in the geocoding result.</i> <i>Indicates if the feature ID of the matched feature is included in the geocoding result.</i> <i>Indicates if the standardized address is included in the geocoding result.</i> <i>Indicates if the x and y coordinates of the address location are included in the geocoding result.</i> <i>Recognized names for a required input field.</i> <i>End offset percentage.</i> <i>Connector strings used to designate intersections.</i> <i>Indicates whether addresses should be arbitrarily matched to a feature when two or more features have the same best score.</i> <i>Minimum candidate score setting.</i> <i>Minimum match score setting.</i> <i>Side offset distance.</i> <i>Units used for the side offset.</i> <i>Spelling sensitivity setting.</i> <i>Indicates if the paths to the reference data should be stored relative to the locator.</i>

Use the *IGeocodingProperties* interface to specify all of the geocoding properties for an *ESRIAddressLocator*. You can use the *DefaultInputFieldNames* property to specify the field names that the locator searches for in an address table when searching for the fields that contain address information. To specify the default input field names for an address field, get the address field name using the *AddressInputs::AddressFields* property, then construct an array of strings containing the default names for the field. The locator searches for the default field names in the address table in the same order in which they are specified in the array.

The following VBA code demonstrates how to add a default field name to the set of default field names for an address field. This code assumes that you have a reference to a locator style that supports the *IGeocodingProperties* interface and that you have already used the *AddressInputs::AddressFields* property to determine the address fields that the locator style uses.

LOCATOR STYLE CLASSES

```
Sub AddDefaultFieldName(pLocatorStyle As ILocatorStyle)
    Dim strDefaultFieldNames As Variant
    Dim pGeocodingProperties As IGeocodingProperties

    Set pGeocodingProperties = pLocatorStyle
    strDefaultFieldNames = _
        pGeocodingProperties.DefaultInputFieldNames("Address")
    ReDim Preserve _
        strDefaultFieldNames(UBound(strDefaultFieldNames) + 1)
    strDefaultFieldNames(UBound(strDefaultFieldNames)) = _
        "NewAddressFieldName"
    pGeocodingProperties.DefaultInputFieldNames("Address") = _
        strDefaultFieldNames
End Sub
```

The *SpellingSensitivity*, *MinimumCandidateScore*, and *MinimumMatchScore* properties let you control how the locator matches addresses to *Features* in the reference data. When matching addresses, locators assign a score to potential candidates between 0 and 100. The spelling sensitivity setting controls how closely the spelling of potential candidates for an address must match the spelling of the address. A higher value for the spelling sensitivity setting means that a greater penalty is assigned to the score of a potential address candidate for each spelling variation. The minimum candidate score setting is the minimum score a potential candidate for an address must have before it is considered as a candidate for the address. The minimum match score setting controls the minimum score that a candidate must have before the locator will automatically match it to the address.

If the locator finds two or more candidates with the same best candidate score, and that score is greater than the minimum match score, then you can use the *MatchIfCandidatesTie* property to specify whether or not the locator should arbitrarily match the address to one of those candidates. If so, the locator will match the address to the first of these candidates that it encounters.

Some locators support intersection geocoding. When geocoding addresses, locators search for certain connector strings that denote intersections, such as "Redlands Blvd. & New York St." or "Centre St. and Dundas St." The *IntersectionConnectors* property lets you specify the connectors that you use in your addresses to designate intersections. This property is a string value that contains all of the intersection connectors. Each connector is separated from other connectors by a space in this string.

The following VBA code can be used to specify the intersection connectors for a locator. This code assumes that you already have a reference to a locator style that supports the *IGeocodingProperties* interface.

```
Sub SpecifyIntersectionConnectors(pLocator As ILocator)
    Dim pGeocodingProperties As IGeocodingProperties
    Set pGeocodingProperties = pLocator
```

```

    pGeocodingProperties.IntersectionConnectors = "@ & AND"
End Sub

```

Some locators support geocoding addresses to a particular side of a street. For those locators, you can specify a side offset using the *SideOffset* property, which specifies how far from the street geocoded addresses should be offset. The *SideOffsetUnits* property specifies the units for the side offset. You can use these properties to improve the cartographic appearance of your geocoded feature classes. By default, the value of the *SideOffsetUnits* property is *esriUnknownUnits*, which indicates to use the reference data's units. You can also use the *EndOffset* property to improve the appearance of your geocoded feature classes. This property is expressed as a percentage of the length of the reference data feature, between 0 and 50. This setting applies a "squeeze factor" to the locations of geocoded addresses so that addresses geocoded at the end of a street segment don't appear to fall on top of a cross street.

Locators can also create additional match information in the fields contained by geocoding results. The *AddPercentageAlongToMatchFields* property specifies whether to add the percentage along a reference data feature at which an address is located to the geocoding output. This property is only applicable if the locator uses a reference data source with line geometry. The *AddReferenceIDToMatchFields* property specifies whether to add the ID of the reference data feature to which an address is matched to the geocoding output. The *AddStandardizeStringToMatchFields* property specifies whether to add the standardized address to the geocoding output. The *AddXYCoordsToMatchFields* property specifies whether to add the x,y coordinates of the geocoded addresses' locations to the geocoding output as two separate numeric fields.

Note that *IGeocodingProperties* inherits from *IAddressInputs*, so all of the read-only properties and methods of this interface are also available. If you want to modify the set of default field names for an address input field when creating a locator, it is best to declare a single *IGeocodingProperties* variable. Refer to the *AddressLocator* abstract class section in this chapter for more information on the *IAddressInputs* interface.

<p>IPlaceNameAlias : IUnknown</p> <ul style="list-style-type: none"> ■ AddressFields: String ■ AliasField: String ■ DefaultAliasFieldNames: Variant ■ Table: ITableName 	<p>Provides access to members that specify a place name alias table.</p> <p>Names of the address fields.</p> <p>Name of the alias field.</p> <p>Recognized names for the address fields in the place name alias table.</p> <p>Name object for the place name alias table.</p>
--	--

Use the *IPlaceNameAlias* interface to specify a place name alias table for the locator. Place name alias tables store aliases for addresses, such as "City Hall", so that you can geocode addresses using alias names rather than addresses. To specify a place name alias table for a locator, you need to indicate which table contains the place name alias information

using the *Table* property. You must also specify the name of the field in this table that contains the alias names. Use the *DefaultAliasFieldNames* property to get the list of default names for the alias field the locator recognizes. You can search the fields in the table for fields with these names, then indicate which field contains the alias names using the *AliasField* property.

You must also specify which fields in the alias table contain the address information. The *AddressFields* property is a comma-delimited string containing the names of the fields in the place name alias table that contain the address information. You can use the *IAddressInputs* interface to determine which fields contain the address information. The field names in this property are specified in the same order as in the *IAddressInputs::AddressFields* property.

The following VBA code demonstrates how to specify a place name alias table for a locator. This code assumes that you already have a reference to the *Name* object for the table that contains the alias information and a reference to a locator that supports the *IAddressInputs* and *IPlaceNameAlias* interfaces. This code also assumes that you have already used the *IAddressInputs* interface to determine which fields in the table contain the address information.

```
Sub SpecifyAliasTable(pLocator As ILocator, pName As IName)
    Dim pPlaceNameAlias As IPlaceNameAlias
    Dim pTable As ITable
    Dim pTableFields As IFields, pTableField As IField
    Dim strDefaultAliasFieldNames As Variant
    Dim i As Long, j As Long
    Dim binAliasFieldFound As Boolean

    Set pPlaceNameAlias = pLocator
    Set pPlaceNameAlias.Table = pName
    Set pTable = pName.Open
    Set pTableFields = pTable.Fields
    strDefaultAliasFieldNames = pPlaceNameAlias.DefaultAliasFieldNames
    For i = LBound(strDefaultAliasFieldNames) To _
        UBound(strDefaultAliasFieldNames)
        For j = 0 To pTableFields.FieldCount - 1
            Set pTableField = pTableFields.Field(j)
            If pTableField.Name = strDefaultAliasFieldNames(i) Then
                binAliasFieldFound = True
                Exit For
            End If
        Next j
        If binAliasFieldFound Then Exit For
    Next i
    pPlaceNameAlias.AliasField = pTableField.Name
    pPlaceNameAlias.AddressFields = "Address,ZIP"
End Sub
```

An *ESRIFDOAddressLocatorStyle* is a locator style that you can use to create locators that use feature data object data sources as geocoding

reference data.

IReferenceDataTables : IUnknown	Provides access to members for specifying the reference data used by the locator.
HasEnoughInfo: Boolean	Indicates if the locator has sufficient reference data in order to geocode.
Tables: IEnumReferenceDataTable	The reference data tables used by the locator.

When creating a locator, use the *IReferenceDataTables* interface to specify the feature data object reference data sources that the locator will use. The *Tables* property returns a *ReferenceDataTableEnumerator* that you can use to specify the reference data tables and feature classes that the locator will use. For more information on specifying the reference data sources for a locator, refer to the *LocatorWorkspace* abstract class and *ReferenceDataTable* abstract class topics in this chapter. Before storing the locator, use the *HasEnoughInfo* property to determine if enough information about the reference data has been specified so that the locator can geocode addresses. This property checks that all of the required reference data sources and required fields in the reference data sources have been specified.

The *StreetMapAddressLocatorStyle* class is a locator style that you can use to create locators that use StreetMap data as geocoding reference data.

IReferenceDataFile : IUnknown	Provides access to members for specifying location of custom reference data file.
Filters: IArray	Filters to use to browse for the reference data file.
PathName: String	Path to the custom reference data file.

Use the *IReferenceDataFile* interface to specify the file-based reference data source that the locator will use. The *Filters* property returns an *Array* of *GxObjectFilters* for this locator style. You can use these filters to set up a *GxDialog* object to allow users to browse for reference data for the locator. Use the *PathName* property to specify the path to the reference data file that the locator will use.

The following VBA code can be used to allow a user to browse for the reference data file that a StreetMap address locator will use. This code assumes that you already have a reference to a StreetMap locator style. This code also assumes that you are licensed to use the StreetMap USA extension and that the StreetMap USA extension is enabled.

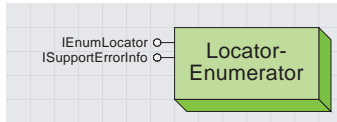
```
Sub SetStreetMapReferenceData(pLocatorStyle As ILocatorStyle)
    Dim pReferenceDataFile As IReferenceDataFile
    Dim pFilters As IArray
    Dim pGxDialog As IGxDialog
    Dim pGxObjectFilterCollection As IGxObjectFilterCollection
    Dim pGxObject As IGxObject
    Dim i As Long

    Set pReferenceDataFile = pLocatorStyle
    Set pFilters = pReferenceDataFile.Filters
    Set pGxDialog = New GxDialog
```

LOCATOR STYLE CLASSES

```
Set pGxObjectFilterCollection = pGxDialog
For i = 0 To pFilters.Count - 1
    pGxObjectFilterCollection.AddFilter pFilters.Element(i), False
Next i
With pGxDialog
    .AllowMultiSelect = False
    .DoModalOpen
    Set pGxObject = .FinalLocation
End With
pReferenceDataFile.PathName = pGxObject.FullName
End Sub
```

LOCATORENUMERATOR CLASS



A `LocatorEnumerator` contains an enumeration of `Locators`. You can obtain a locator enumerator from a `LocatorWorkspace`.

A `LocatorEnumerator` is an enumeration of `Locators` and `LocatorStyles` from a `LocatorWorkspace`. You can obtain a locator enumerator from a locator workspace using the `ILocatorWorkspace::Locators` property.

<code>IEnumLocator : IUnknown</code>	Provides access to members for retrieving a set of locators.
<code>Count: Long</code>	Number of locators in the enumeration.
<code>Clone: IEnumLocator</code>	Creates a copy of the enumeration.
<code>Next: ILocator</code>	Returns the next locator or locator style.
<code>Previous: ILocator</code>	Returns the previous locator or locator style.
<code>Reset</code>	Resets the enumeration.

The `IEnumLocator` interface allows you to inspect the items in the enumeration. The `Next` and `Previous` methods allow you to step through the enumeration sequentially. These methods return the next locator and previous locator, respectively, from the enumeration. The `Reset` method resets the enumeration so that the `Next` method will return the first locator in the enumeration. You should always call the `Reset` method immediately after obtaining the locator enumerator from the locator workspace. The `Count` property returns the number of locators in the enumeration.

If you want to create a copy of the enumeration, use the `Clone` method. This method returns an exact replica of the locator enumerator.

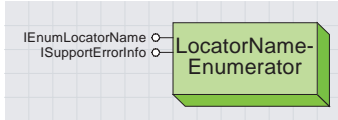
The following VBA code demonstrates how to step through a locator enumerator to retrieve the locators and locator styles that it contains. This code assumes that you have already obtained a locator enumerator from a locator workspace.

```

Sub InspectLocatorEnumerator(pEnumLocator As IEnumLocator)
    Dim pLocator As ILocator
    Dim i As Long

    pEnumLocator.Reset
    For i = 0 To pEnumLocator.Count - 1
        Set pLocator = pEnumLocator.Next
        If TypeOf pLocator Is ILocatorStyle Then
            Debug.Print i & ": Locator style "" & pLocator.Name & """"
        Else
            Debug.Print i & ": Locator "" & pLocator.Name & """"
        End If
    Next i
End Sub
  
```

LOCATORNAMEENUMERATOR CLASS



A *LocatorNameEnumerator* contains an enumeration of locator *Name* objects. You can obtain a locator *Name* object enumerator from a *LocatorWorkspace*.

A *LocatorNameEnumerator* is an enumeration of *LocatorNames* from a *LocatorWorkspace*. You can obtain a locator *Name* object enumerator from a locator workspace using the *ILocatorWorkspace::LocatorNames* property.

IEnumLocatorName : IUnknown	Provides access to members to retrieving a set of Name objects for locators.
Count: Long	Number of Name objects in the enumeration.
Clone: IEnumLocatorName	Creates a copy of the enumeration.
Next: ILocatorName	Returns the next Name object.
Previous: ILocatorName	Returns the previous Name object.
Reset	Resets the enumeration.

The *IEnumLocatorName* interface allows you to inspect the items in the enumeration. The *Next* and *Previous* methods allow you to step through the enumeration sequentially. These methods return the next locator *Name* object and previous locator *Name* object, respectively, from the enumeration. The *Reset* method resets the enumeration so that the *Next* method will return the first locator *Name* object in the enumeration. You should always call the *Reset* method immediately after obtaining the locator *Name* object enumerator from the locator workspace. The *Count* property returns the number of locator *Name* objects in the enumeration. If you want to create a copy of the enumeration, use the *Clone* method. This method returns an exact replica of the locator *Name* object enumerator.

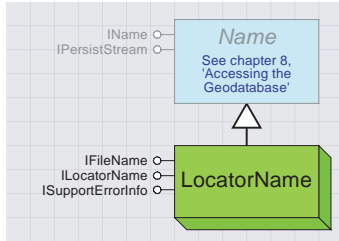
The following VBA code demonstrates how to step through a locator *Name* object enumerator to retrieve the locator *Name* objects that it contains. This code assumes that you have already obtained a locator *Name* object enumerator from a locator workspace.

```

Sub InspectLocatorNameEnumerator(pEnumLocatorName _
    As IEnumLocatorName)
    Dim pLocatorName As ILocatorName
    Dim i As Long
    pEnumLocatorName.Reset
    For i = 0 To pEnumLocatorName.Count
        Set pLocatorName = pEnumLocatorName.Next
        Debug.Print i & ": " & pLocatorName.Name & """"
    Next i
End Sub

```

LocatorName COCLASS



A *LocatorName* object is an object that represents a *Locator*. It is a lightweight object that you can use to maintain a reference to a *Locator*.

A *LocatorName* object is a lightweight object that represents a *Locator* or a *LocatorStyle*. You can get a locator *Name* object from a locator using the *ILocatorDataset::FullName* property. You can also get locator *Name* objects from a *LocatorWorkspace* using the *ILocatorWorkspace::LocatorNames* property or the *ILocatorWorkspace::GetLocatorName* method.

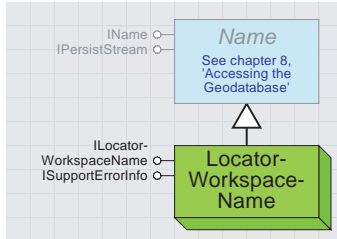
ILocatorName : IUnknown	Provides access to members that describe a locator.
■ Category: String	Category of the locator.
■ Description: String	Description of the locator.
■ LocatorWorkspaceName: ILocatorWorkspaceName	Name object for the locator workspace that contains the locator.
■ Name: String	Name of the locator.
■ Style: Boolean	Indicates if this is a locator style.

Use the *ILocatorName* interface to inspect the properties of the locator *Name* object. The *Name* and *Description* properties return the name and description of the locator or locator style that the locator *Name* object represents. The *Category* property returns the category of the locator or locator style that the locator *Name* object represents. For more information on categories, refer to the *LocatorWorkspace* abstract class topic in this chapter.

To determine if the locator *Name* object represents a locator or a locator style, use the *Style* property.

To get a reference to the locator workspace that contains the locator that the locator *Name* object describes, use the *LocatorWorkspaceName* property. This property returns a *LocatorWorkspaceName* object that represents the locator workspace.

LOCATORWORKSPACEName COCLASS



A *LocatorWorkspaceName* object is a lightweight object that represents a *LocatorWorkspace*. You can use a *LocatorWorkspaceName* object to maintain a reference to a *LocatorWorkspace* without having to open the *LocatorWorkspace*.

A *LocatorWorkspaceName* object is a lightweight object that represents a *LocatorWorkspace*. You can use a *LocatorWorkspaceName* object to improve the efficiency of your application by maintaining a reference to a locator workspace without having to open the locator workspace.

ILocatorWorkspaceName : IUnknown	Provides access to members that describe a locator workspace.
■ Path: String	The path to the folder that contains the locator workspace.
■ Type: esriLocatorWorkspaceType	The type of locator workspace.
■ WorkspaceName: IWorkspaceName	The Name object for the ArcSDE workspace that contains the locator workspace.

Use the *ILocatorWorkspaceName* interface to inspect the properties of the locator workspace *Name* object.

A locator workspace can be contained within an ArcSDE database or can reside on your computer's file system. The *Type* property returns a value from the *esriLocatorWorkspaceType* enumeration that indicates the type of locator workspace that the locator workspace *Name* object represents.

Once you have determined whether the locator workspace resides on an ArcSDE server or on your local file system, you can use either the *Path* property or the *WorkspaceName* property to get the location of the locator workspace. The *Path* property returns the path on the local file system that contains the locator workspace. The *WorkspaceName* property returns a *WorkspaceName* object that represents the ArcSDE workspace that contains the locator workspace.

Enumeration esriLocatorWorkspaceType	Locator workspace types.
0 - esriLocalSystemLocatorWorkspace	Default local locator workspace.
1 - esriFileSystemLocatorWorkspace	Local file system locator workspace.
2 - esriRemoteDatabaseLocatorWorkspace	ArcSDE locator workspace.

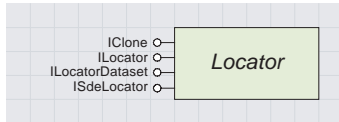
The *esriLocatorWorkspaceType* enumeration lists the types of locator workspaces. Locator workspaces can be contained within ArcSDE databases or on the local file system. The local file system has a default locator workspace that contains all of the locator styles provided with ArcGIS. In addition, any folder in your local file system can contain a locator workspace.

The following VBA code can be used to find the location of the locator workspace represented by a locator workspace *Name* object. This code assumes that you already have a reference to a locator workspace *Name* object.

```
Sub GetLocatorWorkspaceLocation(pLocatorWorkspaceName As _
    ILocatorWorkspaceName)
    Dim strPath As String
    Dim pWorkspaceName As IWorkspaceName

    With pLocatorWorkspaceName
        If .Type = esriFileSystemLocatorWorkspace Or _
            esriLocalSystemLocatorWorkspace Then
            Set strPath = .Path
        ElseIf .Type = esriRemoteDatabaseLocatorWorkspace Then
            Set pWorkspaceName = .WorkspaceName
        End With
    End Sub
```

Locator Abstract Class



A *Locator* creates geometric descriptions for nonspatial descriptions of locations such as addresses.

A *Locator* is an object that creates geometric descriptions for nonspatial descriptions of locations such as addresses. The *Locator* abstract class is the class from which all locators are subclassed.

ILocator : IUnknown	Provides access to members that describe general locator properties.
■ Category: String	Category of the locator.
■ Description: String	Description of the locator.
■ Name: String	Name of the locator.
■ UserInterface: ILocatorUI	User interface for the locator.

All locators support the *ILocator* interface. This interface reveals general properties about the locator. In general, you specify these properties when creating the locator by obtaining a *LocatorStyle* from a *LocatorWorkspace*, setting the properties on this interface, then adding the new locator to the locator workspace.

The *Name* and *Description* properties return the name and description, respectively, of the locator. The *Category* property returns the category of the locator. For more information on locator categories, refer to the *LocatorWorkspace* abstract class topic in this chapter.

The *UserInterface* property returns a reference to the user interface used by the locator. Users can use the locator's user interface to perform such tasks as creating locators, modifying the locator's properties, and matching tables of locations. For more information on locator user interfaces, see the *LocatorUI* abstract class topic in this chapter.

ILocatorDataset : IUnknown	Provides access to members for retrieving objects associated with the Locator.
■ FullName: ILocatorName	The Name object for the locator.
■ LocatorWorkspace: ILocatorWorkspace	The locator workspace that contains the locator.

All locators also support the *ILocatorDataset* interface. This interface provides access to objects that are related to the locator. The *FullName* property returns a *LocatorName* object that represents the locator. The *LocatorWorkspace* property returns a reference to the *LocatorWorkspace* that contains the locator.

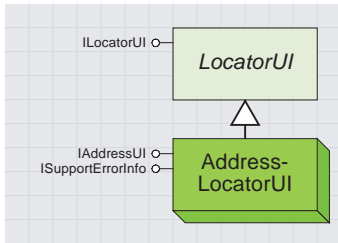
The *ISdeLocator* interface is designed specifically for locators that reside in an ArcSDE database.

ISdeLocator : ILocator	Provides access to locator properties specific to ArcSDE locators.
■ Owner: String	Name of the ArcSDE user who owns the locator.

Locators that are stored in an ArcSDE locator workspace also support the *ISdeLocator* interface. The *Owner* property returns the name of the ArcSDE user that created the locator.

The *ISdeLocator* inherits from *ILocator*, so all of the properties available on *ILocator* are also available on this interface. If you are working with a locator that resides in an ArcSDE database, it is best to declare a single *ISdeLocator* variable.

LOCATOR USER INTERFACE CLASSES



The locator user interface is a template for user interfaces for all locators. The address locator user interface provides access to user interfaces specific to address locators.

The *LocatorUI* abstract class is an abstract class on which user interfaces for *Locators* are based. When you create a custom locator, the user interface that the locator uses must be a subclass of this abstract class. You specify the user interface for a locator when you create it using the *ILocator::UserInterface* property.

ILocatorUI : IUnknown	Provides access to members that control the locator's user interface.
← CreateLocator (in parentWindow: Long, in locatorStyle: ILocatorStyle, in locWks: ILocatorWorkspace, in connectionName: String) : ILocator	Opens the user interface to create a new locator.
← LocatorProperties (in parentWindow: Long, in Locator: ILocator, Title: String) : Boolean	Opens the user interface to view or modify the properties of a locator.
← MatchTable (in parentWindow: Long, in tableOfLocations: ITable, in Locator: ILocator, pathForGxBrowser: String) : IName	Opens the user interface to locate a table.

The *ILocatorUI* interface is supported by all locator user interfaces. It provides methods for displaying the user interfaces that all locators must provide.

The *CreateLocator* method displays the user interface for creating a locator. This user interface allows the user to specify the reference data that the locator will use, set properties on the locator, and store the locator in a *LocatorWorkspace*. This method returns the locator that is created using the user interface.

The following VBA code demonstrates how to use the *CreateLocator* method to display the user interface to create a new locator. This locator assumes that you already have a reference to the *LocatorStyle* on which you want to base the new locator and a reference to the locator workspace that will contain the new locator.

```

Sub CreateLocator(pLocatorStyle As ILocatorStyle, _
    pLocatorWorkspace As ILocatorWorkspace)
    Dim pLocator As ILocator
    Dim pLocatorUI As ILocatorUI
    Dim pNewLocator As ILocator
    Set pLocator = pLocatorStyle
    Set pLocatorUI = pLocator.UserInterface
    Set pNewLocator = pLocatorUI.CreateLocator _
        (ThisDocument.Parent.hWnd, pLocatorStyle, pLocatorWorkspace, "")
End Sub

```

The *LocatorProperties* method displays the user interface for modifying the locator's properties. This method returns a boolean value that indicates if the user committed the changes to the locator workspace. For locator styles provided with ArcGIS, this method returns a value of *True* if the user clicks OK on the user interface and *False* if the user clicks Cancel.

The following VBA code can be used to display the locator properties user interface. This code assumes that you already have a reference to the locator whose properties the user will modify.

LOCATOR USER INTERFACE CLASSES

```

Sub LocatorProperties(pLocator As ILocator)
    Dim pLocatorUI As ILocatorUI
    Set pLocatorUI = pLocator.UserInterface
    If pLocatorUI.LocatorProperties(ThisDocument.Parent.hWnd, _
        pLocator, "Geocoding Service Properties") Then
        Debug.Print "Locator properties saved."
    Else
        Debug.Print "Locator properties not saved."
    End If
End Sub

```

Use the *MatchTable* method to display the user interface for matching a *Table* of locations. This method returns a *Name* object that represents the *FeatureClass* that is created by matching the table. This code assumes that you already have references to the locator and to the table that you want to match.

```

Sub MatchTableUI(pLocator As ILocator, pTable As ITable)
    Dim pLocatorUI As ILocatorUI
    Dim pName As IName
    Set pLocatorUI = pLocator.UserInterface
    Set pName = pLocatorUI.MatchTable(ThisDocument.Parent.hWnd, _
        pTable, pLocator, "")
End Sub

```

The *AddressLocatorUI* class controls the user interfaces for address locators.

IAddressUI : IUnknown	Provides access to members for opening locator user interface dialogs.
← InteractiveReview (in parentWindow: Long, in InputTable: ITable, in Query: IQueryFilter, in inputFieldNames: String, in InputJoinFieldName: String, in OutputTable: IFeatureClass, in outputFieldNames: String, in OutputJoinFieldName: String, in Locator: ILocator)	Opens the Interactive Rematch dialog.
← MatchTableFromSet (in parentWindow: Long, in tableChoices: ISet, in defaultTable: ITable, in bAllowBrowse: Boolean, in Locator: ILocator, pathForGxBrowser: String) : IName	Opens the Geocode Addresses dialog.
← RematchTable (in parentWindow: Long, in InputTable: IFeatureClass, in InputFieldNamesList: String, in InputJoinFieldName: String, in OutputTable: IFeatureClass, in OutputFieldNamesList: String, in OutputJoinFieldName: String, in Locator: ILocator)	Opens the Review/Rematch Addresses dialog.
← RuntimeOptions (in parentWindow: Long, in Locator: ILocator, in bCanChangeOutputFields: Boolean, Title: String) : Boolean	Opens the Geocoding Options dialog.

The *IAddressUI* interface provides access to the user interfaces specific to address locators.

The *RematchTable* interface displays the user interface for rematching a geocoded *FeatureClass*. The *InteractiveReview* method displays the user interface for interactively rematching a feature class of geocoded addresses. To rematch a geocoded feature class, obtain its

AttachedLocator using the *ILocatorManager::GetLocatorFromDataset* method. Most of the parameters for these methods can be obtained directly from the attached locator. For more information on attached locators, refer to the *AttachedLocator* class and *AddressLocator* abstract class topics in this chapter.

The following VBA code demonstrates how to use the *RematchTable* and *InteractiveReview* methods. This code assumes that you already have a reference to a geocoded feature class and that you have an edit session on the *Workspace* that contains the geocoded feature class. The *InteractiveReview* method uses a *QueryFilter* here to restrict the review process to those addresses in the feature class that are unmatched.

```
Sub RematchUI(pFeatureClass As IFeatureClass)
    Dim pLocatorManager As ILocatorManager
    Dim pAttachedLocator As IAttachedLocator
    Dim pLocator As ILocator
    Dim pAddressUI As IAddressUI
    Dim pQueryFilter As IQueryFilter

    Set pLocatorManager = New LocatorManager
    Set pAttachedLocator = _
        pLocatorManager.GetLocatorFromDataset(pFeatureClass)
    Set pLocator = pAttachedLocator.Locator
    Set pAddressUI = pLocator.UserInterface
    With pAttachedLocator
        pAddressUI.RematchTable ThisDocument.Parent.hWnd, .InputTable, _
            .InputFieldNamesList, .InputJoinFieldName, .OutputTable, _
            .OutputFieldNamesList, .OutputJoinFieldName, pLocator
    End With
    Set pQueryFilter = New QueryFilter
    pQueryFilter.WhereClause = "Status = 'U'"
    With pAttachedLocator
        pAddressUI.InteractiveReview ThisDocument.Parent.hWnd, _
            .InputTable, pQueryFilter, .InputFieldNamesList, _
            .InputJoinFieldName, .OutputTable, .OutputFieldNamesList, _
            .OutputJoinFieldName, pLocator
    End With
End Sub
```

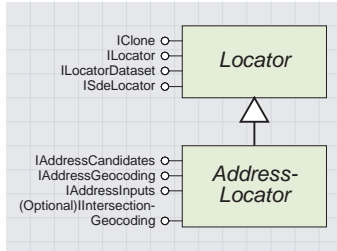
The *MatchTableFromSet* method allows you to select the table that you want to geocode from a *Set* of tables. You can use this method when you have a set of possible tables from which the user can choose to geocode, such as choosing from the set of tables in an ArcMap document. With this method, you can specify whether the user is allowed to browse for additional tables that are not contained in the set. You can also specify the initial path for the *GxDialog* when the user browses for a table.

The *RunTimeOptions* method displays a user interface that allows the user to modify the properties of the locator that can be changed at run time. This method returns a value of *True* if the user saves the changes to the locator and *False* if the user does not. Changing the locator options

LOCATOR USER INTERFACE CLASSES

using this user interface does not modify the locator stored by the locator workspace.

ADDRESSLOCATOR ABSTRACT CLASS



An *AddressLocator* is a *Locator* that can geocode addresses. It supports interfaces specific to geocoding addresses.

The *AddressLocator* abstract class provides access to methods and properties for geocoding addresses. All locators that can geocode addresses are subclasses of this class and are called *AddressLocators*.

IAddressInputs : IUnknown	Provides access to members that specify fields for address tables.
<ul style="list-style-type: none"> ■ AddressFields: IFields ■ DefaultInputFieldNames (in addressField: String) : Variant 	<p><i>Fields needed to geocode a table of addresses.</i></p> <p><i>Recognized names for an input field.</i></p>

The *IAddressInputs* interface provides methods for determining the address components that the locator requires to geocode addresses. The *AddressFields* property specifies the address components that the address locator uses to geocode addresses. You can use the *Required* property on each *Field* object contained in the *Fields* collection to determine whether or not the locator requires that field to geocode the address. You do not need to specify values for fields that are not required, but you can improve the quality of the matches found for your addresses if you use all of the fields that the locator can use. For more information on fields and field collections, see Volume II, chapter 8, 'Accessing the geodatabase'.

You can use the *DefaultInputFieldNames* method to help you in finding the fields in a *Table* that are likely to contain address information. For each address component that the locator uses, this method returns an *Array* of names that the locator uses to determine which fields in a table contain address information. You can use these names as well to find the fields in a table that contain address information.

The following VBA code can be used to determine which fields in a table contain address components that a locator uses. This code assumes that you have already obtained references to a locator and to a table and that the locator supports the *IAddressInputs* interface.

```

Sub FindAddressFields(pTable As ITable, pLocator As ILocator)
    Dim pAddressInputs As IAddressInputs
    Dim pAddressFields As IFields, pAddressField As IField
    Dim pDefaultFieldNames As IArray
    Dim pTableFields As IFields, pTableField As IField
    Dim strDefaultFieldName, strTableFieldName As String
    Dim binFieldFound As Boolean
    Dim i, j, k As Long

    Set pAddressInputs = pLocator
    Set pAddressFields = pAddressInputs.AddressFields
    Set pTableFields = pTable.Fields
    For i = 0 To pAddressFields.FieldCount - 1
        Set pAddressField = pAddressFields.Field(i)
        binFieldFound = False
        Set pDefaultFieldNames = _
            pAddressInputs.DefaultInputFieldNames(pAddressField.Name)
        For j = 0 To pDefaultFieldNames.Count - 1
  
```

```

strDefaultFieldName = pDefaultFieldNames.Element(j)
For k = 0 To pTableFields.FieldCount - 1
    Set pTableField = pTableFields.Field(k)
    If pTableField.Name = strDefaultFieldName Then
        binFieldFound = True
        strTableFieldName = pTableField.Name
        Exit For
    End If
Next k
If binFieldFound Then
    Debug.Print "Found address field '" & pAddressField.Name & _
        "' in table field '" & strTableFieldName & "'."
    Exit For
End If
Next j
If Not binFieldFound Then
    If pAddressField.Required Then
        Debug.Print "WARNING: Required address field '" & _
            pAddressField.Name & "' not found in table."
    Else
        Debug.Print "Optional address field '" & _
            pAddressField.Name & "' not found in table."
    End If
End If
Next i
End Sub

```

IAddressGeocoding : IUnknown	Provides access to members for geocoding addresses.
<ul style="list-style-type: none"> ▣ MatchFields: IFields 	<p><i>Fields contained in the geocoding result.</i></p>
<ul style="list-style-type: none"> ← MatchAddress (in address: IPropertySet) : IPropertySet 	<p><i>Geocodes a single address.</i></p>
<ul style="list-style-type: none"> ← MatchTable (in addressTable: ITable, in addressFieldNames: String, in WhereClause: String, in outputFeatureClass: IFeatureClass, in outputFieldNames: String, in fieldsToCopy: IPropertySet, CancelTracker: ITrackCancel) 	<p><i>Geocodes a table of addresses.</i></p>
<ul style="list-style-type: none"> ← Validate 	<p><i>Checks that the locator's reference data and indexes are present and accessible.</i></p>

The *IAddressGeocoding* interface provides members for geocoding addresses. Use the *MatchAddress* method to geocode a single address. To geocode an address, you must construct a *PropertySet* that contains address components as its properties. The names of the properties are the names of the fields returned by the *IAddressInputs::AddressFields* property. The values of the properties are the components of the address that you want to geocode. The *MatchFields* property returns a set of fields that defines the properties in the property set returned by this method.

The following VBA code demonstrates how to geocode a single address. This code assumes that you already have a reference to a locator that supports the *IAddressGeocoding* interface and that you have used the *IAddressInputs* interface to determine the address components that the locator uses.

```
Sub GeocodeSingleAddress(pLocator As ILocator)
    Dim pAddressGeocoding As IAddressGeocoding
    Dim pAddressPropertySet As IPropertySet
    Dim pMatchPropertySet As IPropertySet
    Dim pMatchFields As IFields, pMatchField As IField
    Dim i As Long
    Dim pGeometry As IGeometry

    Set pPropertySet = New PropertySet
    With pPropertySet
        .SetProperty "Address", "380 New York St."
        .SetProperty "Zone", "92373"
    End With
    Set pAddressGeocoding = pLocator
    Set pMatchPropertySet = _
        pAddressGeocoding.MatchAddress(pAddressPropertySet)
    Set pMatchFields = pAddressGeocoding.MatchFields
    For i = 0 To pMatchFields.FieldCount - 1
        Set pMatchField = pMatchFields.Field(i)
        If pMatchField.Type = esriFieldTypeGeometry Then
            Set pGeometry = _
                pMatchPropertySet.GetProperty (pMatchField.Name)
        Else
            Debug.Print pMatchField.Name & ": " & _
                pMatchPropertySet.GetProperty(pMatchField.Name)
        End If
    Next i
End Sub
```

Use the *MatchTable* method to geocode a table of addresses. Before you geocode a table of addresses, you must first create the *FeatureClass* that will contain the geocoded features, then define the fields that the feature class contains. You can also specify fields from the address table to copy to the geocoded feature class. By default, ArcGIS creates two copies of the address fields from the address table to the geocoded feature class. One copy is a static copy that contains the original values of the address components from the address table. The second copy is an editable copy that you can use to modify the address when rematching the geocoded feature class. In order to be able to rematch a geocoded feature class, you must attach a locator to the geocoded feature class.

The following VBA code can be used to geocode an address table. This code assumes that you already have a reference to a locator that supports the *IAddressGeocoding* interface, a reference to the table (that has an *ObjectID* field) that you want to geocode, and a reference to the *Workspace* in which you want to create the geocoded feature class. This code also assumes that you have used the *IAddressInputs* interface to determine the fields in the table that contain the address components and that you have used the *MatchFields* property to determine the match fields that the locator generates.

```
Sub MatchTable(pLocator As ILocator, pTable As ITable, _
    pWorkspace As IWorkspace)
    Dim pAddressGeocoding As IAddressGeocoding
    Dim pTableFields As IFields, pMatchFields As IFields
    Dim pTableField As IField, pTableFieldEdit As IFieldEdit
    Dim pOutputFields As IFieldsEdit, pOutputField As IFieldEdit
    Dim pUID As UID
    Dim pFeatureWorkspace As IFeatureWorkspace
    Dim pFeatureClass As IFeatureClass
    Dim pMatchFields As IFields
    Dim pPropertySet As IPropertySet
    Dim pLocatorDataset As ILocatorDataset
    Dim pLocatorAttached As ILocatorAttach
    Dim strOutputFieldNames As String
    Dim i As Long

    Set pAddressGeocoding = pLocator
    Set pTableFields = pTable.Fields
    Set pMatchFields = pAddressGeocoding.MatchFields
    Set pOutputFields = New Fields
    pOutputFields.FieldCount = pTableFields.FieldCount + _
        pMatchFields.FieldCount + 1
    Set pOutputField = New Field
    With pOutputField
        .Type = esriFieldTypeOID
        .Name = "ObjectID"
    End With
    Set pOutputFields.Field(0) = pOutputField
    For i = 1 To pTableFields.FieldCount
        Set pTableField = pTableFields.Field(i - 1)
        If pTableField.Type = esriFieldTypeOID Then
            Set pTableFieldEdit = pTableField
            With pTableFieldEdit
                .Type = esriFieldTypeInteger
                .Name = "TABLE_" & pTableField.Name
            End With
        End If
        Set pOutputFields.Field(i) = pTableField
    Next i
    For i = pTableFields.FieldCount + 1 To pTableFields.FieldCount + _
        pMatchFields.FieldCount
        Set pOutputFields.Field(i) = _
            pMatchFields.Field(i - pTableFields.FieldCount - 1)
    Next i
    Set pUID = New UID
    pUID.Value = "esricore.Feature"
    Set pFeatureWorkspace = pWorkspace
    Set pFeatureClass = pFeatureWorkspace.CreateFeatureClass _
        ("Geocoding_Result", pOutputFields, pUID, Nothing, _
        esriFTSimple, "Shape", "")
```

```

For i = 0 To pMatchFields.FieldCount - 1
    strOutputFieldNames = strOutputFieldNames & _
        pMatchFields.Field(i).Name
    If Not (i = pMatchFields.FieldCount - 1) Then
        strOutputFieldNames = strOutputFieldNames & ","
    End If
Next i
Set pPropertySet = New PropertySet
For i = 0 To pTableFields.FieldCount - 1
    If pTableFields.Field(i).Type = esriFieldTypeOID Then
        pPropertySet.SetProperty "TABLE_" & _
            pTableFields.Field(i).Name, pTableFields.Field(i).Name
    Else
        pPropertySet.SetProperty pTableFields.Field(i).Name, _
            pTableFields.Field(i).Name
    End If
Next i
pAddressGeocoding.MatchTable pTable, "Address,ZIP", "", _
    pFeatureClass, strOutputFieldNames, pPropertySet, Nothing
Set pLocatorDataset = pLocator
Set pLocatorAttach = pLocatorDataset.LocatorWorkspace
pLocatorAttach.AttachLocator pLocator, pFeatureClass, _
    pFeatureClass, "Address,ZIP", "ObjectID", pFeatureClass, _
    strOutputFieldNames, "ObjectID"
End Sub

```

The *Validate* method checks all of the reference data and associated geocoding indexes used by the locator and determines if they are present and accessible. You should use this method before using a locator to geocode addresses.

IAddressCandidates : IUnknown	Provides access to members for generating candidates for an address.
▪ CandidateFields: IFields	Fields contained in a list of candidates.
◀ FindAddressCandidates (in address: IPropertySet) : IArray	Generates candidates for an address.

The *IIntersectionGeocoding* interface is designed specifically for geocoding intersection addresses. Only AddressLocators that support intersection geocoding support this interface.

IIntersectionGeocoding : IUnknown	Provides access to members for geocoding intersections.
▪ IntersectionCandidateFields: IFields	Fields contained by intersection candidates.
◀ IsIntersection (in address: IPropertySet) : Boolean	Indicates if an address is an intersection.

To inspect the candidates that a locator generates for an address, use the *IAddressCandidates* and *IIntersectionGeocoding* interfaces. The *IAddressCandidates* interface provides members for inspecting candidates for addresses. The *FindAddressCandidates* method generates candidates for an address. The *CandidateFields* property specifies the fields that address candidates contain. Locators that support intersection geocoding support the *IIntersectionGeocoding* interface, which provides properties for inspecting candidates for intersection addresses. The *IsIntersection* property indicates if an address specifies an intersection address. The *IntersectionCandidateFields* property specifies the fields that the

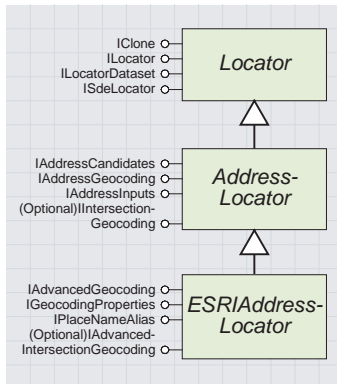
intersection address candidates contain.

The following VBA code can be used to inspect candidates for an address. This code assumes that you already have a reference to a locator and that you have already constructed a *PropertySet* containing the address components.

```
Sub InspectCandidates(pAddress As IPropertySet, _
    pLocator As ILocator)
    Dim pAddressCandidates As IAddressCandidates
    Dim pCandidates() As IPropertySet, pCandidate As IPropertySet
    Dim pIntersectionGeocoding As IIntersectionGeocoding
    Dim pCandidateFields As IFields, pCandidateField As IField
    Dim pGeometry As IGeometry
    Dim i As Long, j As Long

    Set pAddressCandidates = pLocator
    Set pCandidates = _
        pAddressCandidates.FindAddressCandidates(pAddress)
    Set pIntersectionGeocoding = pLocator
    If pIntersectionGeocoding.IsIntersection(pAddress) Then
        Set pCandidateFields = _
            pIntersectionGeocoding.IntersectionCandidateFields
    Else
        Set pCandidateFields = pAddressCandidates.CandidateFields
    End If
    For i = LBound(pCandidates) To UBound(pCandidates)
        Set pCandidate = pCandidates(i)
        Debug.Print "Candidate " & i & ":"
        For j = 0 To pCandidateFields.FieldCount - 1
            Set pCandidateField = pCandidateFields.Field(j)
            If pCandidateField.Type = esriFieldTypeGeometry Then
                Set pGeometry = pCandidate.GetProperty(pCandidateField.Name)
            Else
                Debug.Print pCandidateField.Name & ": " & _
                    pCandidate.GetProperty(pCandidateField.Name)
            End If
        Next j
        Debug.Print
    Next i
End Sub
```

ESRIADDRESSLOCATOR ABSTRACT CLASS



An *ESRIAddressLocator* is a *Locator* that geocodes addresses using the ESRI geocoding engine.

The *ESRIAddressLocator* abstract class defines *Locators* that use the ESRI geocoding engine. This class supports interfaces that have members specific to this geocoding engine.

IAdvancedGeocoding : IAddressGeocoding	Provides access to members for advanced geocoding functions.
StandardizeFields: IFields	Fields contained in a standardized address.
FindStandardizedAddressCandidates (in standardizedAddress: IPropertySet) : IArray	Generates candidates for a standardized address.
MatchStandardizedAddress (in standardizedAddress: IPropertySet) : IPropertySet	Geocodes a single standardized address.
RematchTable (in pInputTable: ITable, in inputFieldName: String, in inputJoinFieldName: String, in resultTable: IFeatureClass, in outputFieldNames: String, in outputJoinFieldName: String, in WhereClause: String, CancelTracker: ITrackCancel)	Rematches a geocoded feature class or shapefile.
StandardizeAddress (in address: IPropertySet, out isAnIntersection: Boolean) : IPropertySet	Standardizes an address.

The *IAdvancedGeocoding* interface provides access to members for standardizing addresses and for rematching geocoded *FeatureClasses*.

All of the locators provided with ArcGIS standardize addresses before matching them to the reference data. Sometimes, locators are not able to standardize addresses properly and thus cannot match them. If you find addresses that a locator cannot match, examining how they are standardized may help you determine why they cannot be matched. You may even be able to modify the standardization so that the addresses are standardized properly and thus improve the likelihood that the locator can match them. To standardize an address, use the *StandardizeAddress* method. You can inspect the results of this method using the *Fields* returned by the *StandardizeFields* property.

Advanced Geocoding inherits from *IAddressGeocoding*. All of the properties and methods that are available on *IAddressGeocoding* are also available here. If you want to inspect the standardization of addresses while geocoding, it is best to declare a single *IAdvancedGeocoding* variable.

The following VBA code can be used to inspect the standardization of an address. This code assumes that you already have a reference to a locator that supports the *IAdvancedGeocoding* interface and that you have used the *AddressInputs::AddressFields* property to determine which address fields the locator uses to geocode addresses.

```

Sub Standardize(pLocator As ILocator)
    Dim pAddressPropertySet As IPropertySet
    Dim pAdvancedGeocoding As IAdvancedGeocoding
    Dim pStandardizedPropertySet As IPropertySet
    Dim pStandardizeFields As IFields, pStandardizeField As IField
    Dim i As Long
    
```

```
Set pAddressPropertySet = New PropertySet
With pAddressPropertySet
    .SetProperty "Street", "380 New York St."
    .SetProperty "Zone", "92373"
End With
Set pAdvancedGeocoding = pLocator
Set pStandardizedPropertySet = _
    pAdvancedGeocoding.StandardizeAddress(pAddressPropertySet, _
    False)
Set pStandardizeFields = pAdvancedGeocoding.StandardizeFields
For i = 0 To pStandardizeFields.FieldCount - 1
    Set pStandardizeField = pStandardizeFields.Field(i)
    With pStandardizeField
        Debug.Print .AliasName & ": " & _
            pStandardizedPropertySet.GetProperty(.Name)
    End With
Next i
End Sub
```

Once you have inspected the standardization of the address and made any necessary changes to the address (by modifying the *PropertySet* returned by the *StandardizeAddress* method), you can use the *FindStandardizedAddressCandidates* method to generate a set of candidates for the address or use the *MatchStandardizedAddress* method to match the address. You can inspect the results of these methods using the *Fields* returned by the *IAddressGeocoding::MatchFields* property.

The following VBA code demonstrates how to find candidates as well as a match for a standardized address. This code assumes that you already have a reference to a locator that supports the *IAdvancedGeocoding* interface and that you have already used the *StandardizeAddress* method to get a property set containing the standardized address.

```
Sub MatchStandardizedAddress(pLocator As ILocator, _
    pStandardizedAddress As IPropertySet)
    Dim pAdvancedGeocoding As IAdvancedGeocoding
    Dim pAddressCandidates As IAddressCandidates
    Dim pCandidateFields As IFields, pCandidateField As IField
    Dim pCandidates As IArray, pCandidate As IPropertySet
    Dim pGeometry As IGeometry
    Dim pMatchFields As IFields, pMatchField As IField
    Dim pMatch As IPropertySet
    Dim i As Long, j As Long

    Set pAdvancedGeocoding = pLocator
    Set pAddressCandidates = pLocator
    Set pCandidateFields = pAddressCandidates.CandidateFields
    Set pCandidates = _
        pAdvancedGeocoding.FindStandardizedAddressCandidates _
        (pStandardizedAddress)
    For i = 0 To pCandidates.Count - 1
        Set pCandidate = pCandidates.Element(i)
```

```

Debug.Print
Debug.Print "Candidate " & CStr(i + 1) & ":"
For j = 0 To pCandidateFields.FieldCount - 1
    Set pCandidateField = pCandidateFields.Field(j)
    With pCandidateField
        If Not .Type = esriFieldTypeGeometry Then
            Debug.Print .AliasName & ": " & _
                pCandidate.GetProperty(.Name)
        Else
            Set pGeometry = pCandidate.GetProperty(.Name)
        End If
    End With
Next j
Next i
Set pMatchFields = pAdvancedGeocoding.MatchFields
Set pMatch = pAdvancedGeocoding.MatchStandardizedAddress _
    (pStandardizedAddress)
Debug.Print
Debug.Print "Match:"
For j = 0 To pMatchFields.FieldCount - 1
    Set pMatchField = pMatchFields.Field(j)
    With pMatchField
        If Not .Type = esriFieldTypeGeometry Then
            Debug.Print .AliasName & ": " & pMatch.GetProperty(.Name)
        Else
            Set pGeometry = pCandidate.GetProperty(.Name)
        End If
    End With
Next j
End Sub

```

You can rematch a geocoded feature class using the *RematchTable* method. To rematch a geocoded feature class, you first need to obtain the *AttachedLocator* from the feature class using the *LocatorManager::GetLocatorFromDataset* method. The attached locator provides all of the information required by the *RematchTable* method to rematch the geocoded feature class. For more information on attached locators and rematching geocoding feature classes, see the *AttachedLocator* class and *AddressLocator* abstract class topics in this chapter.

The following VBA code can be used to rematch the unmatched records in a geocoded feature class. This code assumes that you already have a reference to a geocoded feature class and that you have an edit session on the workspace that contains the geocoded feature class.

```

Sub RematchFeatureClass(pFeatureClass As IFeatureClass)
    Dim pLocatorManager As ILocatorManager
    Dim pAttachedLocator As IAttachedLocator
    Dim pAdvancedGeocoding As IAdvancedGeocoding

```

```

Set pLocatorManager = New LocatorManager
Set pAttachedLocator = _
    pLocatorManager.GetLocatorFromDataset(pFeatureClass)
Set pAdvancedGeocoding = pAttachedLocator.Locator
With pAttachedLocator
    pAdvancedGeocoding.RematchTable .InputTable, _
        .InputFieldNamesList, .InputJoinFieldName, .OutputTable, _
        .OutputFieldNamesList, .OutputJoinFieldName, "Status = 'U'", _
        Nothing
End With
End Sub

```

IGeocodingProperties : IAddressInputs	Provides access to members that control geocoding settings.
■ AddPercentAlongToMatchFields: Boolean	Indicates if the percentage along the reference feature at which the address is located is included in the geocoding result.
■ AddReferenceIDToMatchFields: Boolean	Indicates if the feature ID of the matched feature is included in the geocoding result.
■ AddStandardizeStringToMatchFields: Boolean	Indicates if the standardized address is included in the geocoding result.
■ AddXYCoordsToMatchFields: Boolean	Indicates if the x and y coordinates of the address location are included in the geocoding result.
— DefaultInputFieldNames (in addressField: String) : Variant	Recognized names for a required input field.
■ EndOffset: Long	End offset percentage.
■ IntersectionConnectors: String	Connector strings used to designate intersections.
■ MatchIfScoresTie: Boolean	Indicates whether addresses should be arbitrarily matched to a feature when two or more features have the same best score.
■ MinimumCandidateScore: Long	Minimum candidate score setting.
■ MinimumMatchScore: Long	Minimum match score setting.
■ SideOffset: Double	Side offset distance.
■ SideOffsetUnits: esriUnits	Units used for the side offset.
■ SpellingSensitivity: Long	Spelling sensitivity setting.
■ UseRelativePaths: Boolean	Indicates if the paths to the reference data should be stored relative to the locator.

Use the *IGeocodingProperties* interface to specify all of the geocoding properties for an *ESRIAddressLocator*. For information on using this interface, see the *ESRIAddressLocatorStyle* abstract class topic in this chapter.

IGeocodingProperties inherits from *IAddressInputs*. The read-only properties available on *IAddressInputs* are also available here.

IPlaceNameAlias : IUnknown	Provides access to members that specify a place name alias table.
■ AddressFields: String	Names of the address fields.
■ AliasField: String	Name of the alias field.
■ DefaultAliasFieldNames: Variant	Recognized names for the address fields in the place name alias table.
■ Table: ITableName	Name object for the place name alias table.

Use the *IPlaceNameAlias* interface to specify a place name alias table for the locator. For more information on using this interface, refer to the *ESRIAddressLocatorStyle* abstract class topic in this chapter.

The *IAdvancedIntersectionGeocoding* interface is designed specifically for working with standardized intersection addresses. Only *ESRIAddressLocators* that support intersection geocoding support this interface.

IAdvancedIntersectionGeocoding : IntersectionGeocoding	Provides access to members for advanced intersection geocoding functions.
StandardizeIntersectionFields: IFields	Fields contained in a standardized intersection.
FindStandardizedIntersectionCandidates (in standardizedIntersection: IPropertySet) : IArray	Generates candidates for a standardized intersection.
MatchStandardizedIntersection (in standardizedIntersection: IPropertySet) : IPropertySet	Geocodes a single standardized intersection.

Locators that use the ESRI geocoding engine and that support intersection geocoding support the *IAdvancedIntersectionGeocoding* interface. You can use this interface to inspect the standardization of intersection addresses and to find candidates for and then match standardized intersection addresses. Once you have determined if an address is an intersection and standardized it using the *IAdvancedGeocoding::StandardizeAddress* method, use the *Fields* returned by the *StandardizeIntersectionFields*. Once you have inspected the standardization of the intersection, you can find candidates for the standardized intersection address using the *FindStandardizedIntersectionCandidates* method, then match the address using the *MatchStandardizedIntersection* method.

IAdvancedIntersectionGeocoding inherits from *IIntersectionGeocoding*. All of the properties and methods available on *IIntersectionGeocoding* are also available here. If you want to inspect the standardization of intersection addresses while geocoding, it is best to declare a single *IAdvancedIntersectionGeocoding* variable.

The following VBA code can be used to find candidates for and match a standardized intersection address. This code assumes that you already have a reference to a *Locator* that supports the *IAdvancedIntersectionGeocoding* interface and that you have already obtained a property set containing a standardized intersection address.

```

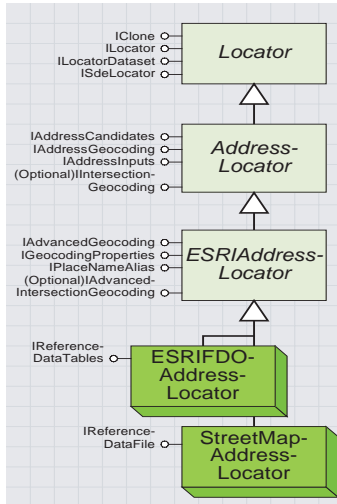
Sub MatchStandardizedIntersection(pLocator As ILocator, _
    pAddress As IPropertySet)
    Dim pAdvancedGeocoding As IAdvancedGeocoding
    Dim pStandardizedAddress As IPropertySet
    Dim pAdvIntersectionGeocoding As IAdvancedIntersectionGeocoding
    Dim pCandidateFields As IFields, pCandidateField As IField
    Dim pCandidates As IArray, pCandidate As IPropertySet
    Dim pGeometry As IGeometry
    Dim pMatchFields As IFields, pMatchField As IField
    Dim pMatch As IPropertySet
    Dim i As Long, j As Long

    Set pAdvancedGeocoding = pLocator
    Set pStandardizedAddress = _
        pAdvancedGeocoding.StandardizeAddress(pAddress, True)
    Set pAdvIntersectionGeocoding = pLocator
    Set pCandidateFields = _
        pAdvIntersectionGeocoding.IntersectionCandidateFields

```

```
Set pCandidates = _
pAdvIntersectionGeocoding.FindStandardizedIntersectionCandidates(pStandardizedAddress)
For i = 0 To pCandidates.Count - 1
  Set pCandidate = pCandidates.Element(i)
  Debug.Print
  Debug.Print "Candidate " & CStr(i + 1) & ":"
  For j = 0 To pCandidateFields.FieldCount - 1
    Set pCandidateField = pCandidateFields.Field(j)
    With pCandidateField
      If .Type = esriFieldTypeGeometry Then
        Set pGeometry = pCandidate.GetProperty(.Name)
      Else
        Debug.Print .AliasName & ": " & _
          pCandidate.GetProperty(.Name)
      End If
    End With
  Next j
Next i
Set pMatchFields = pAdvancedGeocoding.MatchFields
Set pMatch = _
pAdvIntersectionGeocoding.MatchStandardizedIntersection(pStandardizedAddress)
Debug.Print
Debug.Print "Match:"
For j = 0 To pMatchFields.FieldCount - 1
  Set pMatchField = pMatchFields.Field(j)
  With pMatchField
    If .Type = esriFieldTypeGeometry Then
      Set pGeometry = pCandidate.GetProperty(.Name)
    Else
      Debug.Print .AliasName & ": " & pMatch.GetProperty(.Name)
    End If
  End With
Next j
End Sub
```

ESRI ADDRESS LOCATOR CLASSES



An *ESRIAddressLocator* is a *Locator* that uses the ESRI geocoding engine and feature data object data sources as reference data.

A *StreetMapAddressLocator* is a *locator* that uses *StreetMap* data as reference data.

ESRIFDOAddressLocators are *AddressLocators* that use feature data object data sources as reference data.

IReferenceDataTables : IUnknown	Provides access to members for specifying the reference data used by the locator.
<ul style="list-style-type: none"> HasEnoughInfo: Boolean Tables: IEnumReferenceDataTable 	<p>Indicates if the locator has sufficient reference data in order to geocode.</p> <p>The reference data tables used by the locator.</p>

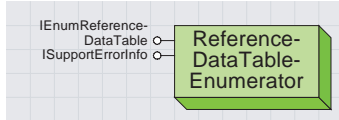
The *IReferenceDataTables* interface provides access to the feature data object data sources used as reference data by the locator. For more information on this interface, refer to the *Locator Style* classes topic in this chapter.

StreetMapAddressLocators are address locators that use *StreetMap* data sources as reference data.

IReferenceDataFile : IUnknown	Provides access to members for specifying location of custom reference data file.
<ul style="list-style-type: none"> Filters: IArray PathName: String 	<p>Filters to use to browse for the reference data file.</p> <p>Path to the custom reference data file.</p>

The *IReferenceDataFile* interface provides access to the file-based reference data source used as reference data by the locator. For more information on using this interface, see the *Locator Style* classes topic in this chapter.

REFERENCE DATATABLE ENUMERATOR CLASS



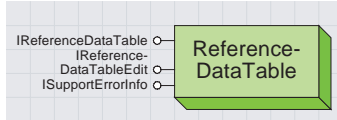
A `ReferenceDataTableEnumerator` contains an enumeration of `ReferenceDataTables` that the `Locator` uses.

The `ReferenceDataTableEnumerator` class contains an enumeration of feature data object data sources used by a `locator` as reference data. To get a reference data table enumerator from a `Locator` or `LocatorStyle`, use the `IReferenceDataTables::Tables` method.

<code>IEnumReferenceDataTable : IUnknown</code>	Provides access to members for retrieving the reference data tables.
<code>Count: Long</code>	<i>The number of reference data tables for the locator.</i>
<code>Next: IReferenceDataTable</code>	<i>Returns the next reference data table.</i>
<code>Reset</code>	<i>Resets the enumeration.</i>

The `IEnumReferenceDataTable` interface provides members for inspecting the `ReferenceDataTables` contained by the enumerator. The `Count` property indicates how many `ReferenceDataTables` are contained in the enumerator. When you obtain a reference data table enumerator from a `locator` or `locator style`, you must use the `Reset` method. This ensures that the first time you call the `Next` method, the first reference data table in the enumerator is returned.

REFERENCEDATATABLE CLASS



A *ReferenceDataTable* provides information about a feature data object data source that a *Locator* uses as reference data.

The *ReferenceDataTable* class provides information about a feature data object data source that a *Locator* uses as reference data.

IReferenceDataTable : IUnknown	Provides access to members for retrieving reference data table information.
<ul style="list-style-type: none"> ■ DisplayName: String ■ Fields: IEnumReferenceDataField ■ Filters: IArray ■ GeocodingIndexes: IEnumReferenceDataIndex ■ Name: ITableName 	<p><i>Name for the reference data table to display in the user interface.</i></p> <p><i>Reference data fields in the reference data table.</i></p> <p><i>Filters to use to browse for the reference data table.</i></p> <p><i>Geocoding indexes on the reference data table.</i></p> <p><i>Name of the reference data table.</i></p>

The *IReferenceDataTable* interface provides information about the data source used by the locator.

The *Name* property returns a *Name* object that represents the data source represented by the reference data table. The *DisplayName* property returns a name for the reference data table that describes the role it plays as a reference data source for the locator. Locators provided with ArcGIS have reference data tables with display names of “Primary table” for the main reference data sources used by locators and “Alternate Name table” for alternate street name tables used by locators. Locators that are based on ArcView® GIS 3 geocoding indexes use a display name of “Table1” for reference data sources.

The *Fields* property returns a *ReferenceDataFieldsEnumerator* object that you can use to specify and inspect the fields that the locator uses in the reference data table. The *GeocodingIndexes* property returns a *ReferenceDataIndexEnumerator* object that you can use to specify and inspect the geocoding indexes used by the locator.

IReferenceDataTableEdit : IReferenceDataTable	Provides access to members for modifying reference data table information.
<ul style="list-style-type: none"> ■ Name: ITableName 	<p><i>Name of the reference data table.</i></p>

The *IReferenceDataTableEdit* interface allows you to specify the reference data source that the *ReferenceDataTable* object represents. When creating a locator, you can use this interface to specify the reference data source that the locator will use or to repair the path to the reference data if it has changed. If you allow the user to browse for reference data, you can use the *IReferenceDataTable::Filters* property to determine which filters a *GxDialog* can use to browse for reference data.

IReferenceDataTableEdit inherits from *IReferenceDataTable*. All of the read-only properties available on *IReferenceDataTable* are also available here. When specifying a *ReferenceDataTable* for a locator, it is best to declare a single *IReferenceDataTableEdit* variable.

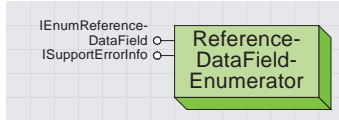
The following VBA code demonstrates how to specify a reference data table for a locator. This code assumes that you already have a reference

to a *ReferenceDataTable* object.

```
Sub BrowseForReferenceDataTable _
    (pReferenceDataTableEdit As IReferenceDataTableEdit)
    Dim pFilters As IArray
    Dim pGxDialog As IGxDialog
    Dim pGxObjectFilterCollection As IGxObjectFilterCollection
    Dim pGxCatalog As IGxCatalog
    Dim pGxDataset As IGxDataset, pTableName As ITableName
    Dim i As Long

    Set pFilters = pReferenceDataTableEdit.Filters
    Set pGxDialog = New GxDialog
    Set pGxObjectFilterCollection = pGxDialog
    For i = 0 To pFilters.Count - 1
        pGxObjectFilterCollection.AddFilter pFilters.Element(i), False
    Next i
    With pGxDialog
        .AllowMultiSelect = False
        .DoModalOpen
        Set pGxCatalog = .InternalCatalog
    End With
    Set pGxDataset = pGxCatalog.SelectedObject
    Set pTableName = pGxDataset.DatasetName
    Set pReferenceDataTableEdit.Name = pTableName
End Sub
```

REFERENCE DATA FIELD ENUMERATOR CLASS



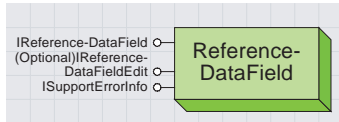
A `ReferenceDataFieldEnumerator` contains an enumeration of `ReferenceDataFields` that describes `Fields` in a `ReferenceDataTable` for a `Locator`.

The `ReferenceDataFieldEnumerator` class contains an enumeration of `ReferenceDataFields` that describe `Fields` used by a `Locator` in a `ReferenceDataTable`. To get a reference to a reference data field enumerator, use the `IReferenceDataTable::Fields` property. When you specify a reference data table for a locator using the `IReferenceDataTableEdit::Table` property, the locator attempts to find fields in the table that contain address information. You can determine whether the locator was able to find all of the fields it requires using the `IReferenceDataTables::HasEnoughInfo` property. If this property returns a value of `False`, you may have to inspect the reference data fields, then specify which fields the locator should use.

IEnumReferenceDataField : IUnknown	Provides access to members for retrieving the reference data fields.
Count: Long	The number of reference data fields in the reference data table.
Next: IReferenceDataField	Returns the next reference data field.
Reset	Resets the enumeration.

The `IEnumReferenceDataField` interface provides members for inspecting the reference data fields contained by the enumerator. The `Count` property indicates how many reference data fields are contained in the enumerator. When you obtain a reference data field enumerator from a reference data table, you must use the `Reset` method. This ensures that the first time you call the `Next` method the first reference data field in the enumerator is returned.

REFERENCEDATAFIELD CLASS



A *ReferenceDataField* describes a Field in a *ReferenceDataTable* that a *Locator* uses to find candidates for an address.

A *ReferenceDataField* describes a *Field* in a *ReferenceDataTable* that a *Locator* uses to find candidates for addresses. You can obtain reference data fields from a *ReferenceDataFieldsEnumerator*.

IReferenceDataField : IUnknown	Provides access to members for inspecting a reference data field.
■ DisplayName: String	Name for the reference data field to display in the user interface.
■ InternalName: String	Internal name for the reference data field.
■ IsObjectID: Boolean	Indicates if the field is the <i>ObjectID</i> field.
■ IsShape: Boolean	Indicates if the field is the <i>Shape</i> field.
■ Name: String	Name of the field in the reference data table.
■ Required: Boolean	Indicates whether the field is a required by the locator.

The *IReferenceDataField* interface provides properties that describe the field in a reference data table that the reference data field represents. The *Name* property is the name of the field in the reference data table that the reference data field represents. The *DisplayName* property is the name of the field that appears in the user interface for the locator. The *InternalName* property is a name for the field that is used internally by the locator. This property is provided for developers so that you can refer to a reference data field in different locators by the same name, regardless of the field's name in the reference data table or the user interface.

To determine if a field is required, use the *Required* property. If a reference data field is required by the property but is not specified, then the *IReferenceDataTables::HasEnoughInfo* property will return a value of *False* and you will not be able to store the locator in a *LocatorWorkspace*.

The *IsShape* property indicates whether or not the reference data field describes a shape field in the reference data table, and the *IsObjectID* property indicates whether or not it describes the *ObjectID* field in the reference data table.

The following VBA code can be used to inspect the reference data fields in a reference data table. This code assumes that you already have a reference to a reference data fields enumerator.

```

Sub InspectReferenceDataFields(pEnumReferenceDataField _
    As IEnumReferenceDataField)
    Dim pReferenceDataField As IReferenceDataField
    Dim i As Long

    pEnumReferenceDataField.Reset
    For i = 0 To pEnumReferenceDataField.Count - 1
        Set pReferenceDataField = pEnumReferenceDataField.Next
        Debug.Print
        With pReferenceDataField
            Debug.Print "Display name: " & .DisplayName
            Debug.Print "Internal name: " & .InternalName
            If .IsObjectID Then Debug.Print "(ObjectID field)"
            If .IsShape Then Debug.Print "(Shape field)"
            Debug.Print "Name: " & .Name
            If .Required Then Debug.Print "(Required)"
        End With
    Next i
End Sub

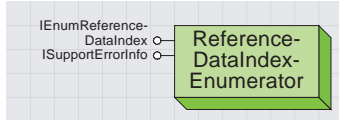
```

IReferenceDataFieldEdit : IReferenceDataField	Provides access to members for modifying reference data field information.
■ DefaultNames: Variant	Recognized names for the reference data field in the reference data table.
■ Name: String	Name of the field in the reference data table.

Use the *IReferenceDataFieldEdit* interface to specify which fields in the reference data table the locator will use as reference data. The *DefaultNames* parameter provides an array of field names that the locator recognizes for the reference data field in the reference data table. The *Name* property lets you specify the name of the field in the reference data table that the locator will use for the reference data field. Ordinarily, you will not need to use this interface, except in cases where the locator cannot find the field in the reference data table from the list of default names.

IReferenceDataFieldEdit inherits from *IReferenceDataField*. All of the read-only properties available on *IReferenceDataField* are also available here. When you are specifying a reference data field, it is best to declare a single *IReferenceDataFieldEdit* variable.

REFERENCE DATA INDEX ENUMERATOR CLASS



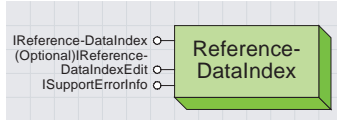
A `ReferenceDataIndexEnumerator` contains an enumeration of `ReferenceDataIndexes` for a `ReferenceDataTable` used by a `Locator`.

The `ReferenceDataIndexEnumerator` class contains an enumeration of `ReferenceDataIndexes` used by the `Locator`. To get a reference data index enumerator, use the `IReferenceDataTable::GeocodingIndexes` property.

<code>IEnumReferenceDataIndex : IUnknown</code>	Provides access to members for retrieving the geocoding indexes.
Count: Long	The number of geocoding indexes on the reference data table.
Next: <code>IReferenceDataIndex</code>	Returns the next geocoding index.
Reset	Resets the enumeration.

The `IEnumReferenceDataIndex` interface provides members for inspecting the `ReferenceDataIndexes` contained by the enumerator. The `Count` property indicates how many reference data indexes are contained in the enumerator. When you obtain a reference data index enumerator from a `ReferenceDataTable`, you must use the `Reset` method. This ensures that the first time you call the `Next` method, the first reference data index in the enumerator is returned.

REFERENCE DATA INDEX CLASS



A *ReferenceDataIndex* describes an index that a *Locator* uses to quickly find candidates for addresses in *ReferenceDataTables*.

The *ReferenceDataIndex* class describes a geocoding index for a *ReferenceDataTable*. A *Locator* uses geocoding indexes to quickly find candidates for addresses that you want to geocode.

IReferenceDataIndex : IUnknown ■— DisplayName: String ■— Exists: Boolean ■— Name: String ← Build (in ConfigKeyword: String, CancelTracker: ITrackCancel)	Provides access to members for retrieving the geocoding index. Name for the geocoding index to display in the user interface. Indicates if the geocoding index exists. Name of the geocoding index. Builds the geocoding index.
---	--

The *IReferenceDataIndex* interface provides information about the geocoding index that it describes and allows you to build or rebuild the geocoding index. The *Name* property returns the name of the geocoding index table or file contained in the same *Workspace* as the reference data table. The *DisplayName* property is a name that can be used in the user interface to describe the table.

The *Exists* property indicates whether or not the geocoding index exists. If it does not exist, or if you want to rebuild it because edits have been made to the reference data table, use the *Build* method. This method allows you to specify an ArcSDE configuration keyword using the *ConfigKeyword* parameter to specify the configuration that an ArcSDE server will use to store the geocoding index table. This parameter is only relevant if the reference data table is contained in an ArcSDE database. For more information on ArcSDE configuration keywords, refer to the *ArcSDE Configuration and Tuning Guide* for your RDBMS.

IReferenceDataIndexEdit : IReferenceDataIndex ■— Name: String	Provides access to members for editing the geocoding index. Name of the geocoding index.
---	--

The *IReferenceDataIndexEdit* interface has only one member, the *Name* property. This property lets you specify the name of the table or file in the same workspace as the reference data table that the *Locator* will use as a geocoding index for the reference data table.

IReferenceDataIndexEdit inherits from *IReferenceDataIndex*. All of the methods and properties on *IReferenceDataIndex* are also available here. When you are specifying a reference data index, it is best to declare a single *IReferenceDataIndexEdit* variable.

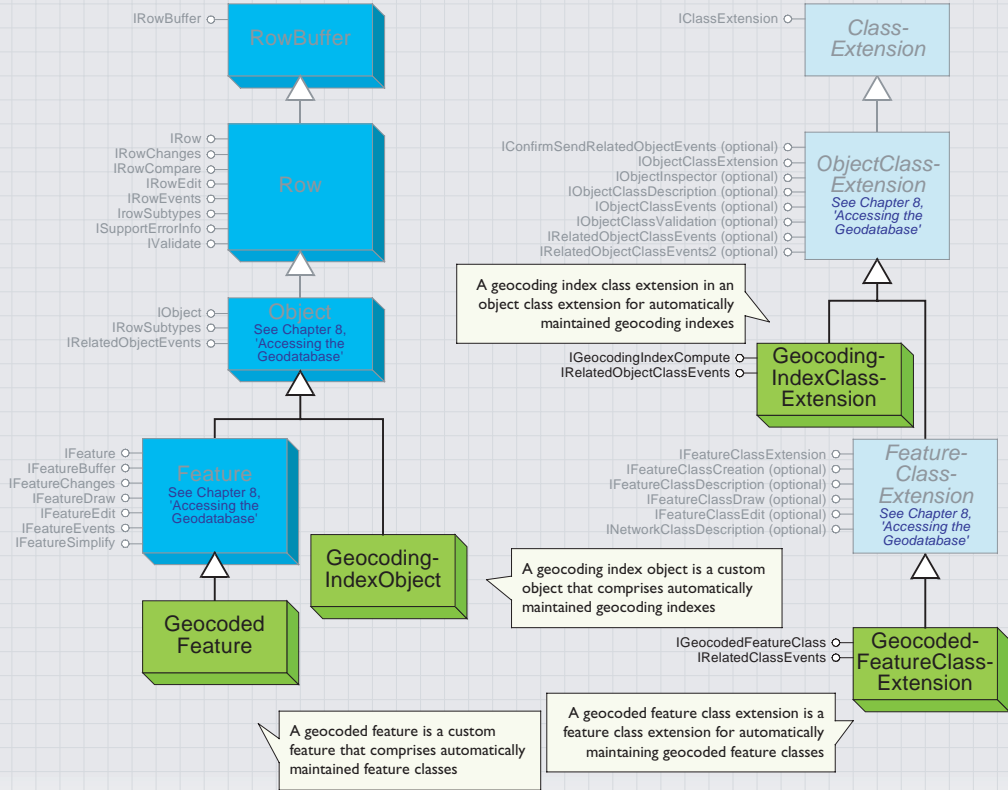
The following VBA code demonstrates how to build the geocoding indexes for a reference data table. This code assumes that you already have a reference to the reference data index enumerator for a reference data table and that you have permissions to create items in the workspace that contains the reference data table.

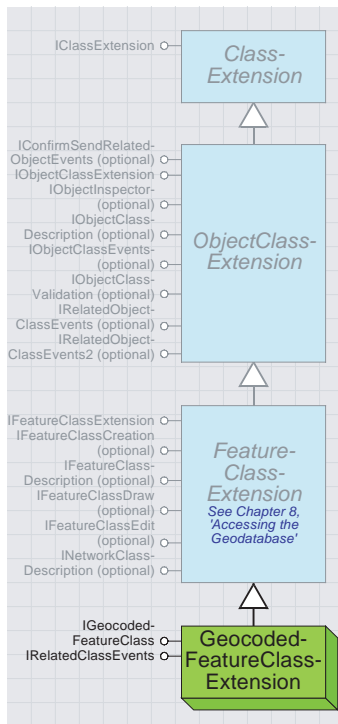
REFERENCE DATA INDEX CLASS

```
Sub BuildGeocodingIndexes(pEnumReferenceDataIndex _
    As IEnumReferenceDataIndex)
    Dim pReferenceDataIndexEdit As IReferenceDataIndexEdit
    Dim i As Long

    pEnumReferenceDataIndex.Reset
    For i = 0 To pEnumReferenceDataIndex.Count - 1
        Set pReferenceDataIndexEdit = pEnumReferenceDataIndex.Next
        If Not pReferenceDataIndexEdit.Exists Then
            pReferenceDataIndexEdit.Build "", Nothing
        End If
    Next i
End Sub
```

Geodatabase extensions





The *GeocodedFeatureClassExtension* is a feature class extension for feature classes you create using an address locator. If you create a relationship class between the address table and the geocoded feature class, the geocoded feature class extension updates the features in the geocoded feature class when you edit the address table.

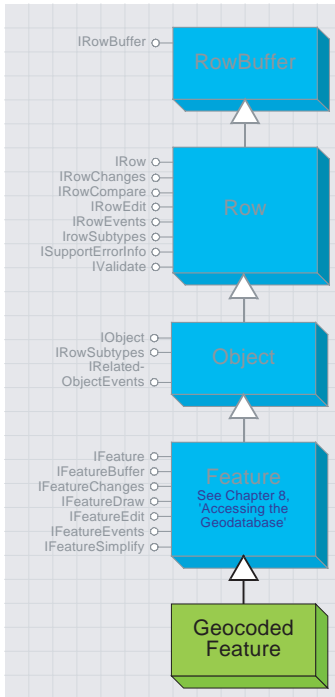
When you geocode a table of addresses in ArcCatalog or ArcMap, you have the choice of either creating a static copy of the attributes from the table of addresses in the *GeocodedFeatureClass* or creating a geocoded feature class that is dynamically related to the table of addresses. If you choose the latter option, a *RelationshipClass* is created between the table of addresses and the geocoded feature class. In addition, the geocoded feature class is registered as a *GeocodedFeature* feature class, and the *ObjectClassExtension* of the feature class is set to the *GeocodedFeatureClassExtension*. These extensions allow you to make edits to the address table and to automatically update the contents of the geocoded feature class to reflect those edits.

The *GeocodedFeatureClassExtension* is a *FeatureClassExtension* that can respond to events from the relationship class and update the *Features* in the geocoded feature class in response to edits that you make to the address table.

When you register a feature class as a geocoded feature class using this extension, use the *IClassSchemaEdit::AlterClassExtensionCLSID* method. Using this method, you specify a *PropertySet* that contains properties for the extension. The geocoded feature class extension is initialized using three properties. The *OriginalAddressFieldNames* property is an array of strings that contain the names of the fields in the address table that contain the address information. The geocoded feature class extension determines if the geocoded feature needs to be updated if these fields are edited in the address table. The *UpdateOnEdit* property is a boolean property that indicates if the extension should update existing addresses if they are edited. If this property has a value of *False*, then addresses are matched only when new records are inserted into the address table. The *UnmatchOnly* property is a boolean property that indicates if the extension should attempt to geocode addresses that are updated in or added to the address table or simply change the status of features to “Unmatched” (or add new unmatched features) in the geocoded feature class.

IGeocodedFeatureClass : IUnknown	Provides access to members for automatically maintaining a geocoded feature class.
← GeocodeAddress (in address: IObject, in result: IFeature)	Updates the geocoded feature.
← NeedsUpdate (in address: IObject, in Shape: IFeature) : Boolean	Indicates if a geocoded feature needs to be updated.

The *IGeocodedFeatureClass* interface provides methods for maintaining a geocoded feature class based on edits to the address table. The *NeedsUpdate* method takes a row from the address table and the corresponding feature in the geocoded feature class as parameters and indicates whether the feature in the geocoded feature class needs to be updated. The *GeocodeAddress* method takes a row from the address table and returns a geocoded feature that can be inserted into the geocoded feature class. In general, you won't need to use the methods on this interface if you're using the *LocatorStyles* provided with ArcGIS. However, if you create a custom locator style, you may need to create a custom feature class extension to maintain feature classes that you geocode with



The geocoded feature is a custom feature for features in a geocoded feature class. If you use the geocoded feature class extension on the geocoded feature class, then the feature class must contain geocoded feature objects.

Locators based on your custom locator style.

The *GeocodedFeature* coclass defines a custom feature. In order to be able to automatically maintain a geocoded feature class using a geocoded feature class extension, the feature class must contain geocoded feature objects.

The following VBA code demonstrates how to set up a geocoded feature class so that it is automatically maintained when you edit the address table. This code assumes that you already have references to the address table, the geocoded feature class, and the workspace that contains them both. It also assumes that the address table has already been registered as an object class and that you know the names of the address fields and the ObjectID field in the address table and the name of the join field in the feature class.

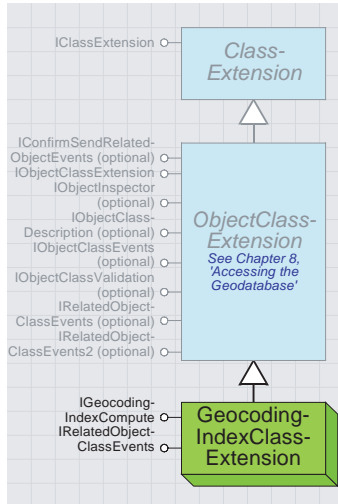
```

Sub MaintainGeocodedFeatureClass(pFeatureWorkspace _
    As IFeatureWorkspace, pAddressTable As ITable, _
    pFeatureClass As IFeatureClass)
    Dim pSchemaLock As ISchemaLock
    Dim pUID As UID
    Dim pClassSchemaEdit As IClassSchemaEdit
    Dim pPropertySet As IPropertySet
    Dim strAddressFields() As String
    Dim pRelationshipClass As IRelationshipClass

    Set pSchemaLock = pFeatureClass
    pSchemaLock.ChangeSchemaLock esriExclusiveSchemaLock
    Set pUID = New UID
    pUID.Value = "esricore.GeocodedFeature"
    Set pClassSchemaEdit = pFeatureClass
    pClassSchemaEdit.AlterInstanceCLSID pUID
    ReDim strAddressFields(0 To 1)
    strAddressFields(0) = "Address"
    strAddressFields(1) = "ZIP"
    Set pUID = New UID
    pUID.Value = "esricore.GeocodedFeatureClassExtension"
    Set pPropertySet = New PropertySet
    With pPropertySet
        .SetProperty "OriginalAddressFieldNames", strAddressFields
        .SetProperty "UpdateOnEdit", True
        .SetProperty "UnmatchOnly", False
    End With
    pClassSchemaEdit.AlterClassExtensionCLSID pUID, pPropertySet
    Set pRelationshipClass = _
        pFeatureWorkspace.CreateRelationshipClass("Geocoding", pTable, _
        pFeatureClass, "is geocoded to", "is geocoded from", _
        esriRelCardinalityOneToOne, esriRelNotificationForward, True, _
        False, Nothing, "ObjectID", "", "Join_OID", "")
    pSchemaLock.ChangeSchemaLock esriSharedSchemaLock
End Sub

```

GEOCODING INDEX CLASSES



The geocoding index class extension is an object class extension for geocoding index tables. If you create a relationship class between the reference data table and the geocoding index table, the geocoding index class extension updates the rows in the geocoding index table when you edit features in the reference data table.

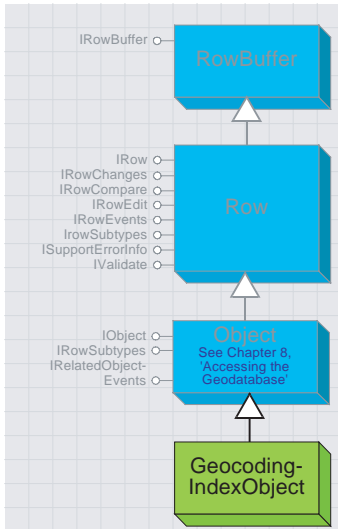
When you create a *Locator* that uses geodatabase *FeatureClasses* and *Tables* as reference data, the locator creates a geocoding index table in the same geodatabase for each feature class and table that it uses as reference data. Ordinarily, if you edit the reference data feature classes or tables, you need to rebuild the corresponding geocoding indexes using the *IReferenceDataIndex::Build* method. However, you can create a *RelationshipClass* between the geocoding reference data and the corresponding geocoding index to automatically update the geocoding index table when you edit a reference data feature class or table. In addition to this, you must register the geocoding index table as a *GeocodingIndexObject* object class and the *ObjectClassExtension* of the geocoding index table must be set to the *GeocodingIndexClassExtension*.

The *GeocodingIndexClassExtension* is an object class extension that can respond to events from the relationship class and update the rows in the geocoding index table in response to edits that you make to the reference data feature class or table. When you register a table as a geocoding index object class, use the *IClassSchemaEdit::AlterClassExtensionCLSID* method. Using this method, you specify a *PropertySet* that contains properties used to initialize the extension. The geocoding index class extension is initialized using different properties depending on the information that the geocoding index table contains. If your reference data table or feature class contains a street name or key field (for single field locators), then the geocoding index table contains a Soundex field. The *SoundexFieldName* property is a string containing the name of the Soundex field in the geocoding index table. The *StreetNameFieldName* property is a string containing the name of the street name or key field in the reference data table or feature class. If your reference data table or feature class contains a single zone field, use the *ZoneFieldName* property to specify the name of the field in the reference data table that contains the zone attribute and the *ZoneSoundexFieldName* property to specify the name of the field in the geocoding index table that contains the Soundex value for the zone. If your reference data table or feature class contains zone information for both the left and right sides of the streets, use the *LeftZoneFieldName* and *RightZoneFieldName* properties to specify the names of these fields and use the *LeftZoneSoundexFieldName* and *RightZoneSoundexFieldName* properties to indicate the names of the fields in the geocoding index table that contain Soundex values for them.

IGeocodingIndexCompute : IUnknown	Provides access to members that calculate geocoding index values.
← ComputeIndexRow (in sourceRow: IObject, in indexRow: IObject)	Generates the index row values for a reference data row.

The *IGeocodingIndexCompute* interface provides a single method, *ComputeIndexRow*, that you can use to generate a row for the geocoding index table from a row in the reference data feature class or table. In general, you won't need to use this interface if you're using the *LocatorStyles* provided with ArcGIS. However, if you develop custom locator styles that use your own indexing method, you may need to create a custom object class extension to maintain geocoding index tables that you create with locators based on your custom locator style.

GEOCODING INDEX CLASSES

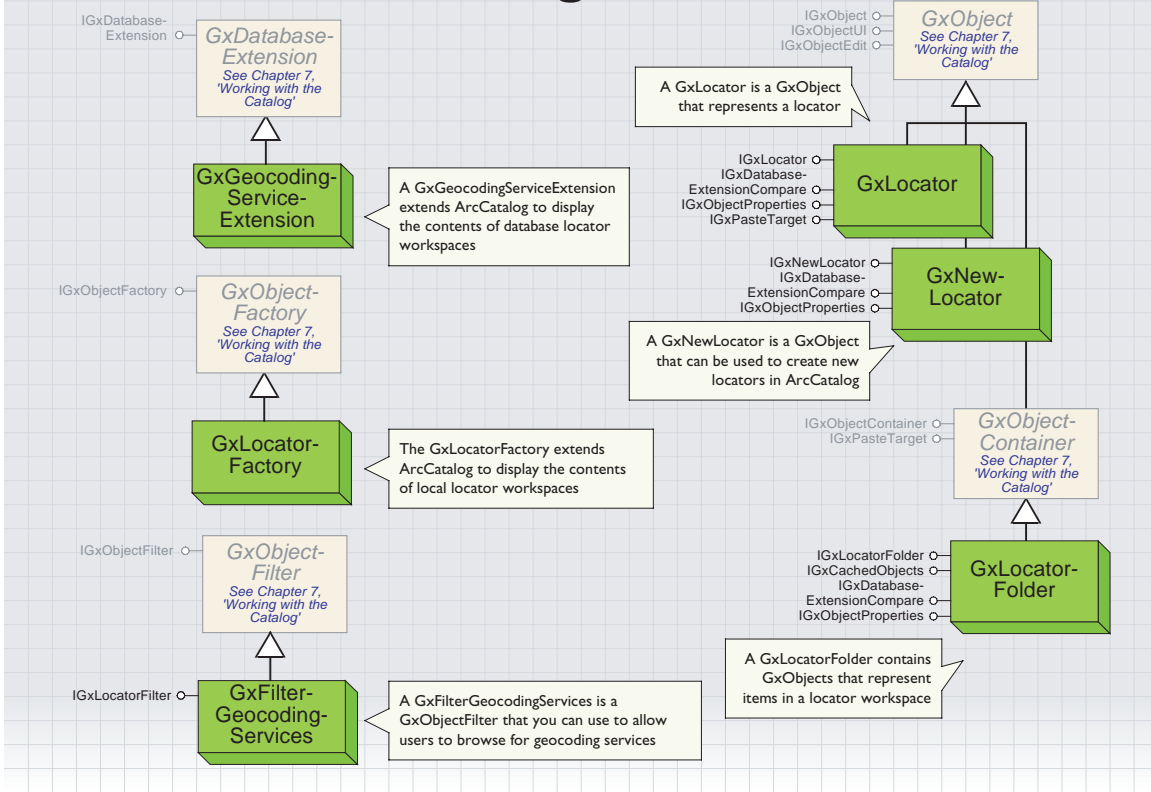


The geocoding index object is a custom object for objects in a geocoding index table. If you use the geocoding index class extension on the geocoding index table, then the geocoding index table must contain geocoding index objects.

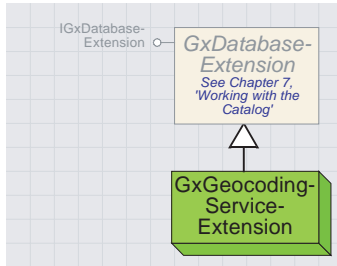
The *GeocodingIndexObject* coclass defines a custom object. In order for the geocoding index object class extension to be able to maintain your geocoding index table, the table must contain geocoding index objects.

For an example of how to set up a geocoding index so that it is maintained automatically, see the Geocoding Index Maintenance sample in the Developer Samples section of the ArcObjects™ Developer Help system.

ArcCatalog extensions



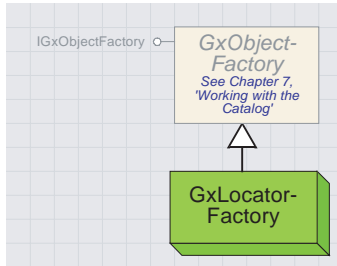
GxGeocodingServiceExtension COCLASS



The `GxGeocodingServiceExtension` extends `ArcCatalog` to display the contents of `ArcSDE` locator workspaces.

The `GxGeocodingServiceExtension` is a `GxDatabaseExtension` that extends `ArcCatalog` to display the contents of `ArcSDE` `LocatorWorkspaces`. Using this extension, `ArcCatalog` can represent an `ArcSDE` locator workspace as a Geocoding Services folder and its constituent geocoding services inside an `ArcSDE` database connection in `ArcCatalog`.

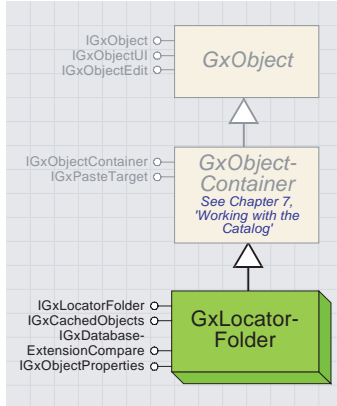
GxLocatorFactory COCLASS



The `GxLocatorFactory` extends `ArcCatalog` to display the contents of locator workspaces on your local file system.

A `GxLocatorFactory` is a `GxObjectFactory` that extends `ArcCatalog` to display the contents of `LocatorWorkspaces` on your local file system. Using the `GxLocatorFactory`, `ArcCatalog` is able to represent the default local locator workspace as the top-level Geocoding Services folder and its constituent geocoding services. This object also allows `ArcCatalog` to create and display `GxLocator` objects that represent `Locators` that are stored in other folders on your local file system.

GxLocatorFolder COCLASS

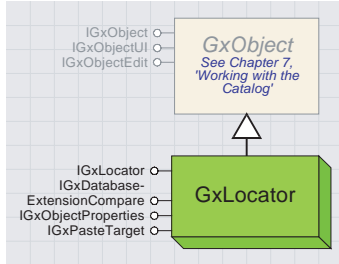


The *GxLocatorFolder* contains *GxObjects* that represent items in a locator workspace. A *GxLocatorFolder* is represented by a Geocoding Services folder in ArcCatalog.

A *GxLocatorFolder* is a *GxObjectContainer* for ArcCatalog that contains *GxObjects* for a *LocatorWorkspace*. A *GxLocatorFolder* is represented in the ArcCatalog user interface by a Geocoding Services folder. The Geocoding Services folder that appears in the top level of the ArcCatalog tree represents the default local locator workspace. ArcSDE database connections also contain a Geocoding Services folder that represents the database locator workspace contained by the ArcSDE database.

IGxLocatorFolder : IUnknown	Provides access to members for editing the properties of an ArcCatalog locator folder.
■-■ LocatorCategory: String	Category of locators in the folder.
■-□ LocatorWorkspace: ILocatorWorkspace	Locator workspace represented by the <i>GxLocatorFolder</i> .

You can use the *IGxLocatorFolder* interface to describe the contents of a *GxLocatorFolder*. The *LocatorWorkspace* property returns a reference to the locator workspace that the *GxLocatorFolder* represents. The *LocatorCategory* property returns the category of *Locators* contained by the *GxLocatorFolder*. For more information on locator categories, see the *LocatorWorkspace Abstract Class* section in this chapter.



The GxLocator is a GxObject that represents a Locator.

A *GxLocator* is a *GxObject* that represents a *Locator*. *GxLocators* appear in the ArcCatalog tree as geocoding services. The *GxGeocodingServiceExtension* creates *GxLocators* that represent server-side locators in ArcSDE databases. The *GxLocatorFactory* creates *GxLocators* that represent client-side locators.

IGxLocator : IUnknown	Provides access to members for retrieving the locator.
<ul style="list-style-type: none"> ■ Locator: ILocator ■ □ LocatorName: ILocatorName 	<ul style="list-style-type: none"> Locator represented by the GxLocator. Name object for the locator.

The *IGxLocator* interface provides properties that describe the locator that the *GxLocator* represents. The *Locator* property returns a reference to the locator that the *GxLocator* object represents. The *LocatorName* property returns a reference to the *LocatorName* object that represents the locator that the *GxLocator* object represents.

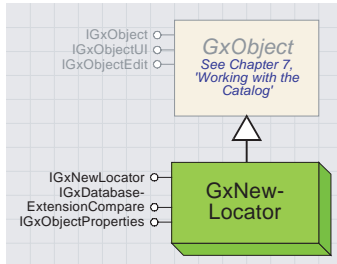
The following VBA code demonstrates how to get a reference to a locator that is selected in the ArcCatalog tree. This code assumes that the selected object in the ArcCatalog tree is a geocoding service.

```

Sub GetSelectedLocator()
    Dim pGxApplication As IGxApplication
    Dim pGxLocator As IGxLocator
    Dim pLocator As ILocator

    Set pGxApplication = ThisDocument.Parent
    Set pGxLocator = pGxApplication.SelectedObject
    Set pLocator = pGxLocator.Locator
    Debug.Print pLocator.Name
End Sub
    
```

GxNewLocator COCLASS



A *GxNewLocator* is a *GxObject* that can be used to create new Locators. A *GxNewLocator* is represented by the *CreateNewGeocodingService* item in a *Geocoding Service* folder.

A *GxNewLocator* object is a *GxObject* that you can use to create a new *Locator*. In ArcCatalog, a *GxNewLocator* object appears as the *Create New Geocoding Service* item inside a *Geocoding Services* folder.

IGxNewLocator : IUnknown	Provides access to members that control which locator styles are displayed when creating new locators in ArcCatalog.
<ul style="list-style-type: none"> ■ LocatorCategory: String ■ LocatorNames: IEnumLocatorName 	<ul style="list-style-type: none"> Category of the locator styles displayed. Names of the locator styles displayed.

The *IGxNewLocator* provides information about the locators that can be created with the *GxNewLocator* object. The *LocatorCategory* property returns the category of locators that can be created using the *GxNewLocator* object. For more information on locator categories, see the *LocatorWorkspace Abstract Class* section in this chapter. The *LocatorNames* property returns a *LocatorNameEnumerator* that contains *LocatorName* objects. The *locator Name* objects contained by this enumerator represent the *LocatorStyles* on which you can base new locators using the *GxNewLocator* object.

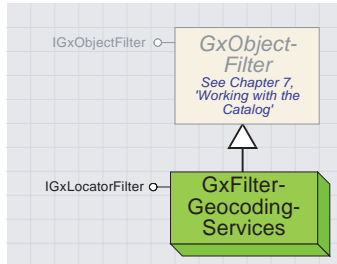
The following VBA code demonstrates how to retrieve a locator style from a *GxNewLocator* object in ArcCatalog in order to create a new locator. This code assumes that you have a *Create New Geocoding Service* item selected in the ArcCatalog tree.

```

Sub GxCreateNewLocator()
    Dim pGxApplication As IGxApplication
    Dim pGxNewLocator As IGxNewLocator
    Dim pEnumLocatorName As IEnumLocatorName
    Dim pLocatorName As ILocatorName, pName As IName
    Dim pLocator As ILocator
    Dim pLocatorWorkspace As ILocatorWorkspace
    Dim pLocatorUI As ILocatorUI
    Dim i As Long

    Set pGxApplication = ThisDocument.Parent
    Set pGxNewLocator = pGxApplication.SelectedObject
    Debug.Print pGxNewLocator.LocatorCategory
    Set pEnumLocatorName = pGxNewLocator.LocatorNames
    pEnumLocatorName.Reset
    For i = 1 To pEnumLocatorName.Count
        Set pLocatorName = pEnumLocatorName.Next
        If pLocatorName.Name = "US Streets (GDB)" Then Exit For
    Next i
    Set pName = pLocatorName
    Set pLocator = pName.Open
    Set pName = pLocatorName.LocatorWorkspaceName
    Set pLocatorWorkspace = pName.Open
    Set pLocatorUI = pLocator.UserInterface
    pLocatorUI.CreateLocator ThisDocument.Parent.hwnd, pLocator, _
        pLocatorWorkspace, ""
End Sub
  
```

GxFilterGeocodingServices COCLASS



A `GxNewFilterGeocodingServices` is a `GxObjectFilter` that you can use in a `GxDialog` to allow users to browse for geocoding services.

The `GxFilterGeocodingServices` coclass describes a `GxObjectFilter` that you can use to browse for geocoding services using a `GxDialog`.

IGxLocatorFilter : IUnknown	Provides access to members that control which locators are displayed.
ShowCreate: Boolean	Indicates whether to display the Create New Locator item.

The `IGxLocatorFilter` interface provides one property, `ShowCreate`, that you can use to indicate whether the `GxDialog` object should display `GxNewLocator` objects.

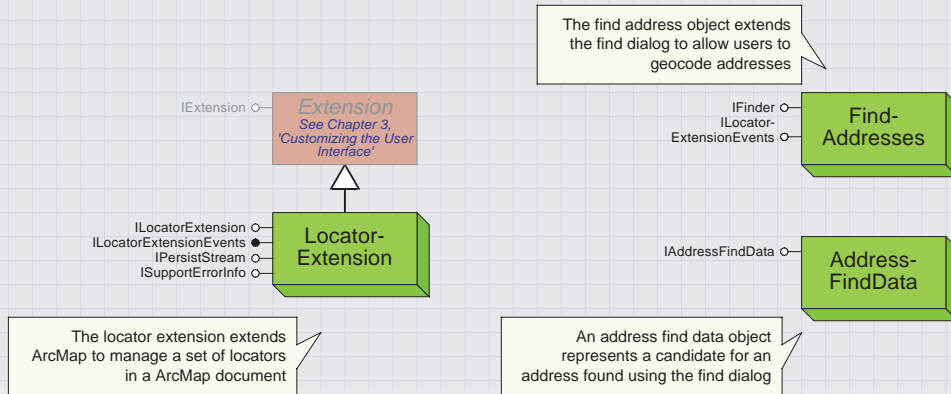
The following VBA code demonstrates how to browse for geocoding services using a `GxDialog`.

```

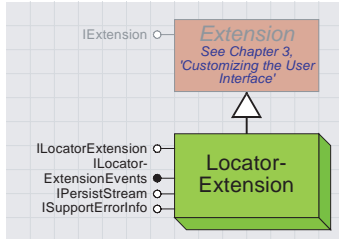
Sub BrowseForGeocodingService()
    Dim pGxObjectFilterCollection As IGxObjectFilterCollection
    Dim pGxLocatorFilter As IGxLocatorFilter
    Dim pGxDialog As IGxDialog
    Dim pGxCatalog As IGxCatalog
    Dim pGxLocator As IGxLocator

    Set pGxObjectFilterCollection = New GxDialog
    Set pGxLocatorFilter = New GxFilterGeocodingServices
    pGxLocatorFilter.ShowCreate = False
    pGxObjectFilterCollection.AddFilter pGxLocatorFilter, True
    Set pGxDialog = pGxObjectFilterCollection
    With pGxDialog
        .AllowMultiSelect = False
        .DoModalOpen ThisDocument.Parent.hwnd, Nothing
        Set pGxCatalog = .InternalCatalog
    End With
    Set pGxLocator = pGxCatalog.SelectedObject
End Sub
  
```

ArcMap extensions



LOCATOREXTENSION COCLASS



The *LocatorExtension* extends *ArcMap* to manage a set of locators within an *ArcMap* document.

The *LocatorExtension* coclass extends *ArcMap* to use *Locators* in an *ArcMap* document. You can use this class to manage the set of locators in an *ArcMap* document and to determine when the set of locators in an *ArcMap* document changes.

ILocatorExtension : IUnknown	Provides access to members that control the set of locators in an <i>ArcMap</i> document.
<ul style="list-style-type: none"> Categories: Variant 	Categories represented by the locators in the <i>ArcMap</i> document.
<ul style="list-style-type: none"> CurrentLocator (in Category: String) : Long 	Current locator in the <i>ArcMap</i> document of a specific category.
<ul style="list-style-type: none"> Locator (in Category: String, in Index: Long) : ILocator 	Locator in the <i>ArcMap</i> document.
<ul style="list-style-type: none"> LocatorCount (in Category: String) : Long 	Number of locators in the <i>ArcMap</i> document in a specific category.
<ul style="list-style-type: none"> ← AddLocator (in Locator: ILocator) : Long 	Adds a locator to the <i>ArcMap</i> document.
<ul style="list-style-type: none"> ← RemoveAllLocators 	Removes all locators from the <i>ArcMap</i> document.
<ul style="list-style-type: none"> ← RemoveCategory (in Category: String) 	Removes a category of locators from the <i>ArcMap</i> document.
<ul style="list-style-type: none"> ← RemoveLocator (in Category: String, in Index: Long) 	Removes a locator from the <i>ArcMap</i> document.

You can use the *ILocatorExtension* interface to manage the set of locators in an *ArcMap* document. The *Categories* property returns an array of strings containing the names of categories represented by the locators in the current *ArcMap* document. For more information on locator categories, see the *LocatorWorkspace Abstract Class* section in this chapter. Using the *ILocatorExtension* interface, you can retrieve a locator by category and index number. The *LocatorCount* property returns the number of locators in the *ArcMap* document for a particular category. The *CurrentLocator* property returns the locator in the specified category at the specified index from the *ArcMap* document. The *CurrentLocator* property returns the index of the current locator for a particular locator category.

The following VBA code can be used to inspect the set of locators in an *ArcMap* document. This code assumes that you have added some locators to an *ArcMap* document.

```

Sub GetCurrentLocators()
    Dim pApplication As IApplication
    Dim pUID As UID
    Dim pLocatorExtension As ILocatorExtension
    Dim strCategories As Variant, strCategory As String
    Dim lngCurrentLocator As Long
    Dim pLocator As ILocator
    Dim i As Long

    Set pApplication = ThisDocument.Parent
    Set pUID = New UID
    pUID.Value = "esricore.LocatorExtension"
    Set pLocatorExtension = pApplication.FindExtensionByCLSID(pUID)
    strCategories = pLocatorExtension.Categories
    For i = LBound(strCategories) To UBound(strCategories)
        strCategory = strCategories(i)
        With pLocatorExtension
            Debug.Print .LocatorCount(strCategory) & " " & strCategory & _
                " locators in the document."
        End With
    Next i
End Sub

```

```

    lngCurrentLocator = .CurrentLocator(strCategory)
    Set pLocator = .Locator(strCategory, lngCurrentLocator)
    Debug.Print "Current " & strCategory & " locator is " & _
    pLocator.Name
End With
Next i
End Sub

```

To add a locator to an ArcMap document, use the *AddLocator* method. This method returns the index of the locator that you added to the ArcMap document. To remove a locator from an ArcMap document, use the *RemoveLocator* method, specifying the category and index of the locator that you want to remove. To remove all locators in a particular category from an ArcMap document, use the *RemoveCategory* method. To remove all of the locators from the document, use the *RemoveAllLocators* method.

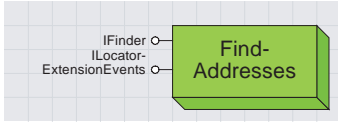
ILocatorExtensionEvents : IUnknown	Provides access to events that occur when locators are added to or removed from an ArcMap document.
← AllLocatorsRemoved	<i>All locators were removed from the ArcMap document.</i>
← CategoryRemoved (in Category: String)	<i>Category of locators was removed from the ArcMap document.</i>
← CurrentLocatorChanged (in Category: String, in Index: Long)	<i>Current locator for the ArcMap document was changed.</i>
← LocatorAdded (in Category: String, in Locator: ILocator, in Index: Long)	<i>Locator was added to the ArcMap document.</i>
← LocatorRemoved (in Category: String, in Index: Long)	<i>Locator was removed from the ArcMap document.</i>

The *ILocatorExtensionEvents* interface provides events that occur when the set of locators in an ArcMap document changes. When a locator is added to the document, the *LocatorAdded* event occurs. When a locator is removed, the *LocatorRemoved* event occurs. The *CurrentLocatorChanged* event occurs when the current locator for a particular category is changed. When all of the locators in a particular category are removed from the document, the *CategoryRemoved* event occurs. When all of the locators are removed from the document, the *AllLocatorsRemoved* event occurs.

To listen for locator extension events using the *ILocatorExtensionEvents* interface, use the following declaration in an object module in VBA. When you add this declaration to an object module, VBA will add methods for these events to your object module. You can then add code to respond to these events.

```
Dim WithEvents m_pLocatorExtensionEvents As LocatorExtension
```

FINDADDRESS COCLASS



The *FindAddress* coclass extends the Find dialog box in ArcMap to allow the user to find addresses using a *Locator*.

ADDRESSFINDDATA COCLASS



The *AddressFindData* coclass represents a candidate for an address that you find using the Find dialog box.

An *AddressFindData* object represents a candidate that you find for an address using the Find dialog box. Each of the items that appears in the Find dialog box results is represented by an *AddressFindData* object.

IAddressFindData : IUnknown	Provides access to members for inspecting address candidates in the Find dialog.
<ul style="list-style-type: none"> ■ □ Geometry: IGeometry ■ ■ Name: String ■ □ Values: IPropertySet 	<ul style="list-style-type: none"> Geometry of the address candidate. Name of the address candidate. Properties of the address candidate.

You can use the *IAddressFindData* interface to inspect address candidates that you find using the Find dialog box in ArcMap. The *Geometry* property returns the shape that the *Locator* creates for the candidate. The *Name* property returns a string containing the concatenated values of the address components that the user typed in the Find dialog box. The *Values* property returns a *PropertySet* containing the candidate values for the candidate. You can inspect this property set using the *AddressCandidates::CandidateFields* property.

The following VBA code can be used to inspect the candidates found using the Find dialog box. This code assumes that the user has right-clicked on one or more candidates in the Find dialog box to display the context menu. You can add this code to the “Address Find Data” context menu to ensure this condition.

```
Sub InspectFindCandidates()
    Dim pApplication As IApplication
    Dim pUID As UID
    Dim pLocatorExtension As ILocatorExtension
    Dim pAddressCandidates As IAddressCandidates
    Dim pCandidateFields As IFields, pCandidateField As IField
    Dim pMxDocument As IMxDocument
    Dim pCandidates As ISet, pCandidate As IPropertySet
    Dim pAddressFindData As IAddressFindData
    Dim pGeometry As IGeometry
    Dim i As Long, j As Long

    Set pApplication = ThisDocument.Parent
    Set pUID = New UID
    pUID.Value = "esricore.LocatorExtension"
    Set pLocatorExtension = pApplication.FindExtensionByCLSID(pUID)
    With pLocatorExtension
        Set pAddressCandidates = .Locator("Address", _
            .CurrentLocator("Address"))
    End With

    Set pCandidateFields = pAddressCandidates.CandidateFields
    Set pMxDocument = ThisDocument
    Set pCandidates = pMxDocument.ContextItem
    pCandidates.Reset
    For i = 1 To pCandidates.Count
        Set pAddressFindData = pCandidates.Next
    Next
End Sub
```

```
Set pGeometry = pAddressFindData.Geometry
Debug.Print
Debug.Print pAddressFindData.Name
Set pCandidate = pAddressFindData.Values
For j = 0 To pCandidateFields.FieldCount - 1
    Set pCandidateField = pCandidateFields.Field(j)
    With pCandidateField
        If Not (.Type = esriFieldTypeOID Or _
            .Type = esriFieldTypeGeometry) Then
            Debug.Print .Name & ": " & pCandidate.GetProperty(.Name)
        End If
    End With
Next j
Next i
End Sub
```

Copyright © 2001 ESRI. All rights reserved. ESRI and ArcView are trademarks of ESRI, registered in the United States and certain other countries; registration is pending in the European Community. StreetMap, ArcCatalog, ArcGIS, ArcObjects, ArcSDE, and ArcMap are trademarks of ESRI. Other companies and products mentioned herein are trademarks or registered trademarks of their respective trademark owners.